ProS: data series progressive k-NN similarity search and classification with probabilistic quality guarantees

Karima Echihabi, Theophanis Tsandilas, Anna Gogolou, Anastasia Bezerianos, Themis Palpanas

▶ **To cite this version:**

Karima Echihabi, Theophanis Tsandilas, Anna Gogolou, Anastasia Bezerianos, Themis Palpanas. ProS: data series progressive k-NN similarity search and classification with probabilistic quality guarantees. The VLDB Journal, 2023, 32, pp.763-789. 10.1007/s00778-022-00771-z . hal-03888664

HAL Id: hal-03888664
https://hal.science/hal-03888664v1

Submitted on 7 Dec 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Noname manuscript No.
(will be inserted by the editor)

ProS: Data Series Progressive k-NN Similarity Search and Classification with Probabilistic Quality Guarantees

Karima Echihabi · Theophanis Tsandilas · Anna Gogolou · Anastasia Bezerianos · Themis Palpanas

This is the author's version of the work. It is posted here for your personal use. Not for redistribution. The definitive version of record was published in the *VLDB Journal (Springer Nature), November 30, 2022*, https://doi.org/10.1007/s00778-022-00771-z.

Abstract Existing systems dealing with the increasing volume of data series cannot guarantee interactive response times, even for fundamental tasks such as similarity search. Therefore, it is necessary to develop analytic approaches that support exploration and decision making by providing progressive results, before the final and exact ones have been computed. Prior works lack both efficiency and accuracy when applied to large-scale data series collections. We present and experimentally evaluate ProS, a new probabilistic learning-based method that provides quality guarantees for progressive Nearest Neighbor (NN) query answering. We develop our method for k-NN queries and demonstrate how it can be applied with the two most popular distance measures, namely, Euclidean and Dynamic Time Warping (DTW). We provide both initial and progressive estimates of the final answer that are getting better during the similarity search, as well suitable stopping criteria for the progressive queries. Moreover, we describe how this method can be used in order to develop a progressive algorithm for data series classification (based on a k-NN classifier), and we additionally propose a method designed specifically for the classification task. Experiments with several and diverse synthetic and real datasets demonstrate that our prediction methods constitute the first practical solutions to the problem, significantly outperforming competing approaches.

Keywords Data Series, Similarity Search, k-NN Classification, Progressive Query Answering

K. Echihabi
Mohammed VI Polytechnic University (UM6P)
E-mail: karima.echihabi@um6p.ma

T. Tsandilas
Université Paris-Saclay, CNRS, Inria, LISN
E-mail: theophanis.tsandilas@lisn.upsaclay.fr

A. Gogolou
Université Paris-Saclay, CNRS, Inria, LISN
E-mail: a.gogolou@gmail.com

A. Bezerianos
Université Paris-Saclay, CNRS, Inria, LISN
E-mail: anastasia.bezerianos@lisn.upsaclay.fr

T. Palpanas
LIPADE, University of Paris & French Univ. Institute (IUF)
E-mail: themis@mi.parisdescartes.fr

1 Introduction

Data Series. Data series are ordered sequences of values measured and recorded from a wide range of human activities and natural processes [98], such as seismic activity, or electroencephalography (EEG) signal recordings. The analysis of such sequences[1] is becoming increasingly challenging as their sizes often grow to multiple terabytes [9, 45, 46, 96].

Data series analysis involves pattern matching [76, 82, 145], anomaly detection [13–15, 17–19, 26, 36, 54, 54, 85, 86, 99, 101, 120, 140], frequent pattern mining [53, 84, 112], clustering [69, 80, 100, 114, 115, 135], and classification [10, 16, 30, 87, 119, 133, 141]. Several algorithms relevant to these tasks rely on *data series similarity*. The data-mining community has proposed several techniques, including many similarity measures (or distance measure algorithms), for calculating the distance between two data series [39, 92, 102], as well as corresponding indexing techniques and algorithms [46, 47, 97], in order to address scalability challenges.

[1] If the dimension that imposes the ordering of the sequence is time then we talk about *time series*. Though, a series can also be defined over other measures (angle in radial profiles, frequency in infrared spectroscopy, etc.). We use the terms *time series*, *data series*, and *sequence* interchangeably.

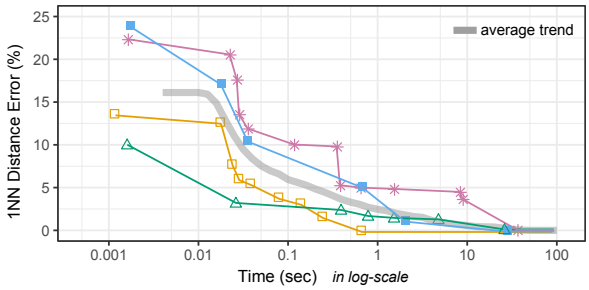


Fig. 1: Progression of 1-NN distance error (Euclidean dist.) for 4 example queries (seismic dataset), using iSAX2+ [23]. The points in each curve represent approximate (intermediate points) or exact answers (last point) given by the algorithm. Lines end when similarity search ends. Thick gray line represents average trend over a random sample of 100 queries.

Data Series Similarity. We observe that data series similarity is often domain- and visualization-dependent [12, 56], and in many situations, analysts depend on time-consuming manual analysis processes. For example, neuroscientists manually inspect the EEG data of their patients, using visual analysis tools, so as to identify patterns of interest [56, 66]. In such cases, it is important to have techniques that operate within interactive response times [95], in order to enable analysts to complete their tasks easily and quickly.

In the past years, several visual analysis tools have combined visualizations with advanced data management and analytics techniques (e.g., [74, 111]), albeit not targeted to data series similarity search. Moreover, we note that even though the data series management community is focusing on scalability issues, the state-of-the-art indexes currently used for scalable data series processing [23, 72, 82, 134, 146] are still far from achieving interactive response times [43, 47].

Progressive Results. To allow for interactive response times when users analyze large data series collections, we need to consider progressive and iterative visual analytics approaches [8, 57, 126, 143]. Such approaches provide progressive answers to users' requests [52, 93, 123], sometimes based on algorithms that return quick approximate answers [38, 50]. Their goal is to support exploration and decision making by providing progressive results. A progressive result is an intermediate answer that iteratively converges to the final, correct solution.

Most of the above techniques consider approximations of aggregate queries on relational databases, with the exception of Ciaccia et al. [31, 32], who provide a probabilistic method for assessing how far an approximate answer is from the exact answer. Nevertheless, these works do not consider data series that are high-dimensional[2]. We note that the framework of Ciaccia et al. [31,32] does not explicitly target progressive similarity search. Furthermore, the approach has only been tested on datasets with up to 275K vectors with dimensionality up to 100, while we are targeting data series vectors in the order of hundreds of millions (in our experiments we provide results with up to 267M series), and with dimensionality that can exceed 1000 (in our experiments we provide results with up to 1280). Our experiments show that the probabilistic estimates that their methods [31, 32] provide are inaccurate and cannot support progressive similarity search.

In this study, we demonstrate the importance of providing progressive similarity search results on large time series collections. Our results show that there is a gap between the time the 1st Nearest Neighbour (1-NN) is found and the time when the search algorithm terminates. In other words, users often wait without any improvement in their answers. We further show that high-quality approximate answers are found very early, e.g., in less than one second, so they can support highly interactive visual analysis tasks.

Figure 1 presents the approximate results of the iSAX2+ index [23] for four example queries on a 100M data series collection with seismic data [122], where we show the evolution of the approximation error as a percentage of the exact 1-NN distance. We observe that the algorithm provides approximate answers within a few milliseconds, and those answers gradually converge to the exact answer, which is the distance of the query from the 1-NN. Interestingly, the 1-NN is often found in less than 1 sec (e.g., see yellow line), but it takes the search algorithm much longer to verify that there is no better answer and terminate. This finding is consistent with previously reported results [32, 57].

Several similarity-search algorithms, such as the iSAX2+ index [23] and the DSTree [134] (the two top performers in terms of data series similarity search [47]), provide very quick approximate answers. In this paper, we argue that such algorithms can be used as the basis for supporting progressive similarity search. Unfortunately, these algorithms do not provide any guarantees about the quality of their approximate answers, while our goal is to provide such guarantees.

Proposed Approach. We develop *ProS*, the first progressive approach for sequence search and classification with probabilistic quality guarantees, which is scalable to very large data series collections. Our goal is to predict how much improvement is expected when the algorithms are still running. Communicating this information to users will allow them to terminate a progressive analysis task early and save time.

[2] The dimensionality of a data series is the length, or number of points in the series [47]. In our context, by high-dimensional, we refer to series with dimensionality in the order of hundreds-thousands.

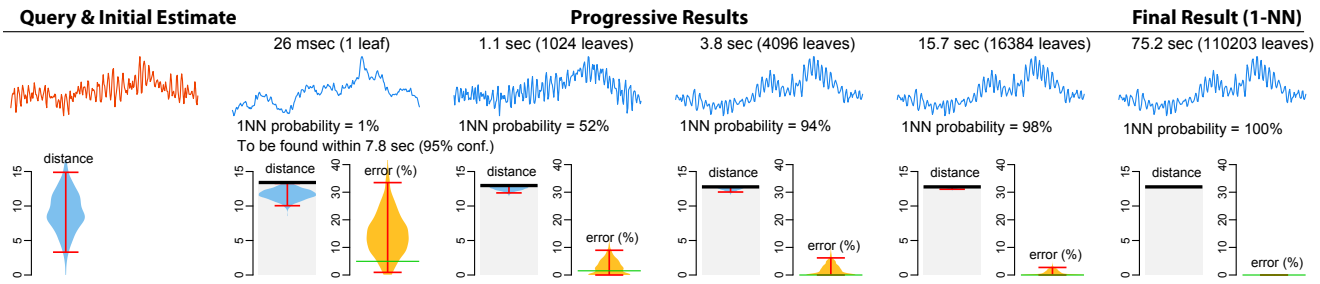


Fig. 2: A query example from the seismic dataset showing the evolution of estimates based on our approach. The thick black lines show the distance of the current approximate answer. The red error bars represent 95% prediction intervals. The green line over the predicted distribution of distance errors shows the real error – it is unknown during the search and is shown here for illustration purposes. Estimates are based on a training set of 100 queries, as well as 100 random witnesses for initial estimates. We use the iSAX2+ index.

Figure 2 showcases our approach with an example on real data. An analyst enters a seismic pattern as a query (in red) and immediately (response times reported at the top of the figure) receives progressive approximations of its 1-NN (in blue). In addition to these progressive answers, the system also provides estimates of the current distance error: the blue distributions [110] estimate the absolute distance error; while the yellow distributions estimate the relative distance error. Observe that the initial distance estimate is rather uncertain, but estimates become precise at the early stages of the search. The system can further communicate a probability of whether the current answer is exact and predict when the exact answer is expected with a certain confidence level. In this example, the query terminates after 75.2sec, but the above predictions can give confidence to the user that the current answer is very close to the 1-NN much earlier (i.e., almost one order of magnitude faster). The user can then decide to stop the query.

The challenge is how to derive such predictions. If we further inspect Figure 1, we see that similarity search answers progressively improve, but improvements are not radical. The error of the first approximate answer (when compared to the final exact answer) is on average 16%, which implies that approximate answers are generally not very far from the 1-NN. We show that this behavior is more general and can be observed across different datasets and different similarity search algorithms [134, 146]. We further show that the distance of approximate answers can help us predict the time that it takes to find the exact answer. Our approach describes these behaviors through statistical models. We then use these models to estimate the error of a progressive answer, assess the probability of an early exact answer, and provide upper bounds for the time needed to find the k-NN. We also explore query-sensitive models that predict a probable range of the k-NN distance before the search algorithm starts, and then is progressively improved as new answers arrive. We further provide reliable stopping criteria for terminating searches with probabilistic guarantees about the distance error or the number of exact answers.

In addition to similarity search, we address the problem of k-NN classification. We show how the early termination of k-NN similarity search can be employed to lead to time savings for k-NN classification. Moreover, we propose probabilistic guarantees for the exact class itself, as well, which allows us to achieve even larger savings.

We note that earlier approaches [31, 32] do not solve the problem, since they support bounds only for *distance* errors, they do not update their estimates during the course of query answering, and they do not scale with the data size.

Contributions. Our key contributions are as follows.

- We formulate the problem of *progressive data series similarity search*, and provide definitions specific to the context of data series.
- We investigate statistical methods, based on regression (linear, quantile, and logistic) and multivariate kernel density estimation, for supporting progressive similarity search based on a small number (50 - 200) of training queries. We show how to apply them to derive estimates for the k-NN distance (distance error), the time to find the k-NN, and the probability that the progressive k-NN answer is correct.
- We further develop stopping criteria that can be used to stop the search long before the normal query execution ends. These criteria make use of distance error estimates, probabilities of exact answers, and time bounds for exact answers. We show how our criteria can be applied with the two most popular data series distance measures, namely, Euclidean and Dynamic Time Warping (DTW).
- Moreover, we describe how our approach extends to data series k-NN classification. In particular, we derive probabilistic guarantees and develop stopping criteria for the exact class of a progressive k-NN classification.
- We perform an extensive experimental evaluation with several and diverse synthetic and real datasets. The re-

sults demonstrate that our solutions dominate the previous approaches, provide accurate probabilistic bounds, and lead to significant time improvements with well-behaved guarantees for errors. Source code and datasets are publicly available [1].

This paper extends our previous work [55] in the following directions:

- We extend the original method that was designed for 1-NN similarity queries to support k-NN queries.
- In addition to the Euclidean distance measure, we now also study in detail the Dynamic Time Warping (DTW) distance in the context of progressive data series similarity search.
- Apart from pattern matching using similarity search, we now propose methods for progressive classification, as well, which is a very popular analysis task for data series collections.
- We expand the empirical evaluation of our methods by adding five new synthetic and real datasets, as well as several new experiments and discussions.
- We expand the discussions of the related work, which helps draw a more complete picture of the research area relevant to our work.

Paper Structure. The rest of this paper is organized as follows. Section 2 summarizes related work on data series similarity search and progressive visual analytics, and Section 3 presents background terminology. In Section 4, we define progressive similarly search and introduce the main concepts for supporting prediction with probabilistic guarantees. Then, in Section 5, we describe our methods for estimating a k-NN distance before and during the execution of a similarity search query, and in Section 6 the corresponding methods for k-NN classification. Section 7 presents an extensive evaluation of all proposed methods. Finally, we conclude in Section 8, where we also propose directions for future work.

2 Related Work

Similarity Search. Several measures have been proposed for computing similarity between data series [39, 92]. Among them, Euclidean Distance (ED) [49], which performs a point-by-point value comparison between two time series, is one of the most popular. ED can be combined with data normalization (often *z-normalization* [59]), in order to consider as similar patterns that may vary in amplitude or value offset. In our work, we focus on ED because it is effective, and leads to efficient solutions for large datasets [39, 47]. We also extend our approach to DTW [113], which is very popular in practice, and more suitable than ED for certain applications, especially in classification tasks [10].

The human-computer interaction community has focused on the interactive visual exploration and querying of data series. These querying approaches are visual, often on top of line chart visualizations [125], and rely either on the interactive selection of part of an existing series (e.g., [21]), or on sketching patterns to search for (e.g., [34, 89]). This line of work is orthogonal to our approach, which considers approximate and progressive results from these queries when interactive search times are not possible.

Optimized and Approximate Similarity Search. The database community has optimized similarity search methods by using index structures [22, 27, 28, 33, 49, 72, 73, 82, 83, 132, 134, 139, 146] or fast sequential scans [112]. Recently, Echihabi et al. [47, 48] compared the efficiency of these methods under a unified experimental framework, showing that there is no single best method that outperforms all the rest. In this study, we focus on the popular centralized solutions, though, our results naturally extend to parallel and distributed solutions [44, 78, 104–108, 138, 139], since these solutions are based on the same principles and mechanisms as their centralized counterparts. Moreover, we focus on (progressive answers for) exact query answering. Given enough time, all answers we produce are exact, which is important for several applications [98]. In this context, progressive answers help to speed-up exact queries by stopping execution early, when it is highly probable that the current progressive answer is the exact one. Note that several data series similarity search methods support approximate query answering that can produce increasingly more accurate answers as time goes by [23, 51, 73, 83, 134, 146], though, none of them provides quality guarantees on the answers. In this work, we focus on the iSAX2+ [23] and DSTree [134] methods, which exhibit superior performance at the similarity search task [47, 48].

In parallel to our work, Li et al. [79] proposed a machine learning method, developed on top of inverted-file (IVF [64] and IMI [7]) and k-NN graph (HNSW [88]) similarity search techniques, that solves the problem of early termination of approximate NN queries, while achieving a target recall. In contrast, our approach employs similarity search techniques based on data series indices [48], and with a very small training set (up to 200 training queries in our experiments), provides guarantees with per-query probabilistic bounds along different dimensions: on the distance error, on whether the current answer is the exact one, and on the time needed to find the exact answer.

k-NN Classification. Similarity-based classification (e.g., k-NN Classifier) is a supervised task consisting of assigning a label to a new item based on the majority vote of its neighbors among the set of labeled training samples. It is used in a

variety of domains, such as bioinformatics for protein classification [4], computer vision for object recognition [30], text mining for web page categorization [75], remote sensing [109], and social media for image classification [40]. We note that, even though lots of work has been dedicated into developing data series classification algorithms, the k-NN classifier remains a strong baseline [10] and the only viable solution in use-cases with (limited hardware resources and) very large amounts of data [109].

To the best of our knowledge, the idea of progressive classification has not been carefully studied before. Previous work has looked at the problem of classifying images at multiple resolutions [24], but does not propose a progressive query answering framework, nor does it provide quality guarantees.

Progressive Visual Analytics. Fekete and Primet [50] provide a summary of the features of a progressive system; three of them are particularly relevant to progressive data series search: (i) progressively improved answers; (ii) feedback about the computation state and costs; and (iii) guarantees of time and error bounds for progressive and final results. Systems that support big data visual exploration include Pangloss [93] that provides quick approximate results of aggregation queries, Falcon [94] that prefetches data for brushing and linking actions, and IncVisage [111] that progressively reveals salient features in heatmap and trendline visualizations.

Systems that provide progressive results are appreciated by users due to their quick feedback [8, 143]. Nevertheless, there are some caveats. Users can be mislead into believing false patterns [93, 126] with early progressive results. It is thus important to communicate the progress of ongoing computations [3, 121], including the uncertainty and convergence of results [3] and guarantees on time and error bounds [50]. Previous work provides such uncertainty and guarantees in relational databases and aggregation type queries [61, 65, 137].

Closer to the context of data series, Ciaccia and Patella [32] studied similarity search queries over general multi-dimensional spaces and proposed a probabilistic approach for computing the uncertainty of partial similarity search results. We discuss their approach in the following section.

3 Preliminaries and Background

A *data series* $S(p_1, p_2, ..., p_\ell)$ is an ordered sequence of real-valued points with length ℓ. A data series of length ℓ can also be represented as a single point in an ℓ-dimensional space. For this reason, the values of a data series are often called *dimensions*, and its length ℓ is called *dimensionality*. We use $\mathbb{S}$ to denote a *data series collection* (or *dataset*). We refer to the size $n = |\mathbb{S}|$ of a data series collection as *cardinality*. In this paper, we focus on datasets with a very large number of regularly sampled data series, with no uncertainty in the values [6, 35, 36, 118, 142], and no missing values [11, 136], which means that we do not need to encode the attribute describing the dimension of the sequence (e.g., the timestamps when the dimension is time). While the techniques used in this paper are designed for series of equal length, our models could be extended to support series of variable length (e.g., following the ideas proposed by the ULISSE index [83]).

Distance Measures. A data series *distance* $d(S_1, S_2)$ is a function that measures the dissimilarity of two data series S_1 and S_2, or alternatively, the dissimilarity of two data series subsequences. As mentioned in Sec 2, we chose Euclidean Distance (ED) as a measure due to its popularity and efficiency [39].

Similarity Search Queries. Given a dataset $\mathbb{S}$, a *query* series Q, and a distance function $d(\cdot, \cdot)$, a *k-Nearest-Neighbor (k-NN) query* identifies the k series in the dataset with the smallest distances to Q. The 1st Nearest Neighbor (1-NN) is the series in the dataset with the smallest distance to Q.

Similarity search can be *exact*, when it produces answers that are always correct, or *approximate*, when there is no such strict guarantee. A *δ-ϵ-approximate algorithm* guarantees that its distance results will have a relative error no more than ϵ with a probability of at least δ [47]. We note that only a couple of approaches [5, 32] provide such guarantees. Yet, their accuracy has never been tested on the range of dimensions and dataset sizes that we examine here.

Similarity Search Methods. Most data series similarity search techniques [22, 33, 49, 72, 82, 97, 107, 134, 139, 146] use an index, which enables scalability. The index can offer quick approximate answers by traversing a single path of the index structure to visit the single most promising leaf, from where we select the *best-so-far (bsf)* answer: this is the candidate answer in the leaf that best matches (has the smallest distance to) the query. The bsf may, or may not be the final, exact answer: in order to verify, we need to either prune, or visit all the other leaves of the index. Having a good first bsf (i.e., close to the exact answer) leads to efficient pruning.

In the general case, approximate data series similarity search algorithms do not provide guarantees about the quality of their answers. In our work, we illustrate how we can efficiently provide such guarantees, with probabilistic bounds.

We focus on index-based approaches that support both quick approximate, and slower but exact, similarity search results. In this work, we adapt the state-of-the-art data series indexes iSAX2+ [23] and DSTree [134], which have been shown to outperform the other data series methods in query answering [47], and we demonstrate that our techniques are

applicable to both indexes. We provide below a succinct description of the iSAX2+ and DSTree approaches.

The iSAX2+ [23] index organizes the data in a tree structure, where the leaf nodes contain the raw data and each internal node summarizes the data series that belong to its subtree using a representation called *Symbolic Aggregate Approximation (SAX)* [81]. SAX transforms a data series using Piecewise Aggregate Approximation (PAA) [68] into equi-length segments, where each segment is associated with the mean value of its points, then represents the mean values using a discrete set of symbols for a smaller footprint.

DSTree [134] is also a tree-based index that stores raw data in the leaves and summaries in internal nodes. Contrary to iSAX2+, DSTree does not support bulkloading, intertwines data segmentation with indexing and uses *Extended Adaptive Piecewise Approximation (EAPCA)* [134] instead of SAX. With EAPCA, a data series is segmented using APCA [25] into varying-length segments, then each segment is represented with its mean and standard deviation values.

Since the query answering time depends on the data distribution [147], and both iSAX2+ and DSTree can produce unbalanced index trees, we provide below an index-invariant asymptotic analysis on the lower/upper bounds of the query runtime. As we consider large on-disk datasets, the query runtime is I/O bound; thus we express complexity in terms of I/O [62,67], using the dataset size n, the index leaf threshold th and the disk block size B. Consider an index over a dataset of size n such that each index leaf contains at most th series ($th \ll n$). We count one disk page access of size B as one I/O operation (for simplicity, we use B to denote the number of series that fit in one disk page). Note that both the iSAX2+ and DSTree indexes fit the entire index tree in-memory; the leaves point to the raw data on-disk.

Best Case. The best case scenario occurs when one of the children of the index root is a leaf, containing one data series. In this case, the approximate search will incur $\Theta(1)$ I/O operation. In the best case, exact search will prune all other nodes of the index and thus will also incur $\Theta(1)$ disk access.

Worst Case. Approximate search always visits one leaf. Therefore, the worst case occurs when the leaf is the largest possible, i.e., it contains th series, in which case approximate search incurs $\Theta(th/B)$ I/O operations. For exact search, the worst case occurs when the algorithm needs to visit every single leaf of the index. This can happen when the index tree has $n - th + 1$ leaves (i.e., each leaf contains only one series, except for one leaf with th series), as a result of each new series insertion causing a leaf split where only one series ends up in one of the children. Therefore, the exact search algorithm will access all the leaves, and will incur $\Theta(N)$ I/O operations. (Note that this is a pathological case that would happen when all series are almost identical: in this case, indexing and similarity search are not useful anyways.)

Baseline Approach. We briefly describe here the probabilistic approach of Ciaccia et al. [31–33]. Based on Ciaccia et al. [33], a dataset $\mathbb{S}$ (a data series collection in our case) can be seen as a random sample drawn from a large population $\mathcal{U}$ of points in a high-dimensional space. Being a random sample, a large dataset is expected to be representative of the original population. Given a query Q, let $f_Q(x)$ be the probability density function that gives the relative likelihood that Q's distance from a random series drawn from $\mathcal{U}$ is equal to x. Likewise, let $F_Q(\cdot)$ be its cumulative probability function. Based on $F_Q(\cdot)$, we can derive the cumulative probability function $G_{Q,n}(\cdot)$ for Q's k-NN distances in a dataset of size $n = |\mathbb{S}|$. For 1-NN similarity search, we have:

$$G_{Q,n}(x) = 1 - (1 - F_Q(x))^n \quad (1)$$

We now have a way to construct estimates for 1-NN distances. Unfortunately, $f_Q(\cdot)$, and thus $F_Q(\cdot)$, are not known. The challenge is how to approximate them from a given dataset. We discuss two approximation methods:

1. *Query-Agnostic Approximation.* For high-dimensional spaces, a large enough sample from the overall distribution $f(\cdot)$ of pairwise distances in a dataset provides a reasonable approximation for $f_Q(\cdot)$ [33]. This approximation can then be used to evaluate probabilistic stopping-conditions by taking sampling sizes between 10% and 1% (for larger datasets) [32].

2. *Query-Sensitive Approximation.* The previous method does not take the query into account. A query-sensitive approach is based on a training set of reference queries, called *witnesses*. Witnesses can be randomly drawn from the dataset, or selected with the GNAT algorithm [20], which identifies the n_w points that best cover a multidimensional (metric) space based on an initial random sample of $3n_w$ points. Given that close objects have similar distance distributions, Ciaccia et al. [31] approximate $f_Q(\cdot)$ by using a weighted average of the distance distributions of all the witnesses.

The above methods have major limitations. First, since their 1-NN distance estimates are static, they are less appropriate for progressive similarity search. Second, a good approximation of $F_Q(\cdot)$ does not necessarily lead to a good approximation of $G_{Q,n}(\cdot)$. This is especially true for large datasets, as the exponent term n in Equation 1 will inflate even tiny approximation errors. Note that $G_{Q,n}(\cdot)$ can be thought of as a scaled version of $F_Q(\cdot)$ that zooms in on the range of the lowest distance values. If this narrow range of distances is not accurately approximated, the approximation of $G_{Q,n}(\cdot)$ will also fail. Our own evaluation demonstrates

Table 1: Table of symbols

Symbol	Description
S, Q	data series, query series
ℓ	length of a data series
$\mathbb{S}$	data series collection (or dataset)
$n = \|\mathbb{S}\|$	number of series in $\mathbb{S}$
$R(t)$	progressive answer at time t
$c_Q(t)$	class of k-NN classification at time t
k-NN, $knn(Q)$	k^{th} Nearest Neighbor of Q
$d_{Q,R}(t)$, $d(Q, R(t))$	distance between Q and $R(t)$
$d_{Q,knn}$, $d(Q, knn(Q))$	distance between Q and its k-NN
$\epsilon_Q(t)$	relative distance error of $R(t)$ from k-NN
$\epsilon_Q^f(t)$	relative family-wise distance error
$p_Q(t)$	probability that $R(t)$ is exact (i.e., the k-NN)
$p_{c_Q}(t)$	probability that the class $c_Q(t)$ is exact
t_Q	time to find the k-NN
$\tau_{Q,\phi}$	time to find the k-NN with probability $1-\phi$
$\tau_{Q,\theta,\epsilon}$	time for which $\epsilon_Q(t) < \epsilon$ with confidence $1-\theta$
$\hat{\bullet}$	estimate of $\bullet$
$I_Q(t)$	information at time t
$h_{Q,t}(x)$	probability density function of Q's distance from its k-NN, given information $I_Q(t)$
$H_{Q,t}(x)$	cumulative distribution function of Q's distance from its k-NN, given $I_Q(t)$
$f_Q(x)$	probability density function of Q's distance from a random series in $\mathbb{S}$
$F_Q(x)$	cumulative distribution function of Q's distance from a random series in $\mathbb{S}$
$G_{Q,n}(x)$	cumulative distribution function of Q's distance from its k-NN
$\mathcal{W}$	set of witness series
$n_w = \|\mathcal{W}\|$	number of witnesses in $\|\mathcal{W}\|$

this problem. Third, they require the calculation of a large number of distances. Since the approximation of $G_{Q,n}(\cdot)$ is sensitive to errors in large datasets (see above), a rather large number of samples is required in order to capture the frequency of the very small distances.

k-NN Classification. Given a training dataset $\mathbb{S}$ with individual data series allocated to a class in $C = \{c_1, c_2..., c_L\}$ and a new data series Q, a k-NN classifier assigns to Q the most common class $c_Q \in C$ among its k nearest neighbors in the training dataset. As a consequence, k-NN classification fully relies on k-NN similarity search, and therefore, there exists a direct link with all the methods that we describe below.

4 Progressive Similarity Search

We define progressive similarity search for k-NN queries[3]. (Table 1 summarizes the symbols we use in this paper.)

Definition 1 Given a k-NN query Q, a data series collection $\mathbb{S}$, and a time *quantum* q, a ***progressive similarity-search algorithm*** produces results $R(t_1), R(t_2), ..., R(t_z)$ at time points $t_1, t_2, ..., t_z$, where $t_{i+1} - t_i \leq q$, such that $d(Q, R(t_{i+1})) \leq d(Q, R(t_i))$.

We borrow the quantum q parameter from Fekete and Primet [50]. It is a user-defined parameter that determines how frequently users require updates about the progress of their search. Although there is no guarantee that distance results will improve at every step of the progressive algorithm, the above definition states that a progressive distance will *never* deteriorate. This is an important difference of progressive similarity search compared to other progressive computation mechanisms, where results may fluctuate before they eventually converge, which may lead users to making wrong decisions based on intermediate results [29, 50, 60].

Clearly, progressive similarity search can be based on approximate similarity search algorithms – a progressive result is simply an approximate (best-so-far) answer that is updated over time. A progressive similarity search algorithm is also exact if the following condition holds:

$$\lim_{t \to \infty} d(Q, R(t)) = d(Q, knn(Q)) \quad (2)$$

where $knn(Q)$ represents the k-NN of the query series Q.

According to the above condition, the progressive algorithm will always find an exact answer. However, there are generally no strong guarantees about how long this can take. Ideally, a progressive similarity search algorithm will find good answers very fast, e.g., within interactive times, and will also converge to the exact answer without long delays. Even so, in the absence of information, users may not be able to trust a progressive result, no matter how close it is to the exact answer.

In this paper, we investigate exactly this problem. Specifically, we seek to provide guarantees about: (i) How close is the progressive answer to the exact answer? (ii) What is the probability that the current progressive answer is the exact answer? (iii) When is the search algorithm expected to find the exact answer?

4.1 Progressive Distance Estimates

Given a progressive answer $R(t)$ to a k-NN query at time t, we are interested in knowing how far from the k-NN this answer is. For simplicity, we will denote the k-NN distance to the query as $d_{Q,knn} \equiv d(Q, knn(Q))$ and the distance between $R(t)$ and the query as $d_{Q,R}(t) \equiv d(Q, R(t))$. Then, the relative distance error is $\epsilon_Q(t) = \frac{d_{Q,R}(t)}{d_{Q,knn}(t)} - 1$. Given that this error is not known, our goal is to find an estimate $\hat{\epsilon}_Q(t)$. However, finding an estimate for the relative error is not any simpler than finding an estimate $\hat{d}_{Q,knn}(t)$ of the actual k-NN distance. We concentrate on this latter quantity for our analysis below. Though, since $d_{Q,R}(t)$ is known, deriving the distance error estimate $\hat{\epsilon}_Q(t)$ from the k-NN distance estimate $\hat{d}_{Q,knn}(t)$ is straightforward:

$$\hat{\epsilon}_Q(t) = \frac{d_{Q,R}(t)}{\hat{d}_{Q,knn}(t)} - 1 \quad (3)$$

We represent progressive similarity-search estimates as probability distribution functions.

[3] We define the problem using k-NN, but for simplicity use $k = 1$ in the rest of this paper. We defer the discussion of the general case to future work.

Definition 2 Given a k-NN query Q, a data series collection $\mathbb{S}$, and a progressive similarity-search algorithm, a ***progressive k-NN distance estimate*** $\hat{d}_{Q,knn}(t)$ of the k-NN distance at time t is a probability density function:

$$h_{Q,t}(x) = Pr\{d_{Q,knn} = x \mid I_Q(t)\} \quad (4)$$

This equation gives the conditional probability that $d_{Q,knn}$ is equal to x, given information $I_Q(t)$.

We expect that progressive estimates will converge to $d_{Q,knn}$ (i.e., $\hat{\epsilon}_Q(t)$ will converge to zero). Evidently, the quality of an estimate at time t depends on the information $I_Q(t)$ that is available at this moment. In Section 5, we investigate different types of information we can use for this.

Given the probability density function in Equation 4, we can derive a point estimate that gives the *expected* k-NN distance, or an interval estimate in the form of a *prediction interval* (PI). Like a confidence interval, a prediction interval is associated with a confidence level. Given a confidence level $1-\theta$, we expect that approximately $(1-\theta)\times 100\%$ of the prediction intervals we construct will include the true k-NN distance. Note that although a confidence level can be informally assumed as a probability (i.e., what is the likelihood that the interval contains the true k-NN distance?), this assumption may or may not be strictly correct. Our experiments evaluate the frequentist behavior of such intervals.

To construct a prediction interval with confidence level $1-\theta$ over a density distribution function $h_{Q,t}(\cdot)$, we derive the cumulative distribution function:

$$H_{Q,t}(x) = Pr\{d_{Q,knn} \leq x \mid I_Q(t)\} \quad (5)$$

From this, we take the $\theta/2$ and $(1-\theta/2)$ quantiles that define the limits of the interval.

4.2 Guarantees for Exact Answers

Users may also need guarantees about the exact k-NN. We investigate two types of probabilistic guarantees for exact answers. First, at any moment t of the progressive search, we assess the probability $p_Q(t)$ that the exact answer has been found, given information $I_Q(t)$:

$$p_Q(t) = Pr\{d_{Q,R}(t) = d_{Q,knn} \mid I_Q(t)\} \quad (6)$$

Second, we estimate the time t_Q it takes to find the exact k-NN. As we already discussed, this time can be significantly faster than the time needed to complete the search. Let $\hat{t}_Q$ be its estimate. We express it as a probability density function:

$$r_{Q,t}(x) = Pr\{t_Q = x \mid I_Q(t)\} \quad (7)$$

which expresses the conditional probability that t_Q is equal to x, given information $I_Q(t)$. From this, we derive its cumulative distribution function $R_Q(\cdot)$. Then, given a confidence level $1-\phi$, we can find a probabilistic upper bound $\tau_{Q,\phi}$ such that $R_Q(\tau_{Q,\phi}) = 1-\phi$; ϕ represents the probability that the progressive answer at time $\tau_{Q,\phi}$ is not the exact, i.e., the proportion of bounds that fail to include the exact answer.

4.3 Stopping Criteria

Based on the provided estimates, users may decide to trust the current progressive result and possibly stop their search. Which *stopping criterion* to use is not straightforward and depends on whether users prioritize guarantees about the k-NN distance, about the relative error of the current progressive result, or about the exact answer itself.

An analyst may choose to stop query execution as soon as the prediction interval of the k-NN distance lies above a low threshold value. Unfortunately, this strategy raises some concerns. Previous work on progressive visualization [90] discusses the problem of confirmation bias, where an analyst may use incomplete results to confirm a "preferred hypothesis". This is a well-studied problem in sequential analysis [129]. It relates to the multiple-comparisons problem [144] and is known to increase the probability of a Type I error (false positives). We evaluate how such multiple sequential tests affect the accuracy of our methods, but discourage their use as stopping criteria, and instead propose the following three.

A first approach is to make use of the relative distance error estimate $\hat{\epsilon}_Q(t)$ (see Eq. 3). For instance, the analyst may decide to stop the search when the upper bound of the error's interval is below a threshold $\epsilon = 1\%$. An error-based stopping criterion offers several benefits: (i) the choice of a threshold does not depend on the dataset, so its interpretation is easier; (ii) this criterion does not inflate Type I errors as long as the threshold ϵ is fixed in advance; (iii) the error $\epsilon_Q(t)$ monotonically converges to zero (the same holds for the bounds of its estimates), thus there is a unique point in time $\tau_{Q,\theta,\epsilon}$ at which the bound of the estimated error reaches ϵ, where $1-\theta$ is our confidence level (here, θ determines the proportion of times for which the relative distance error of our result will be greater than ϵ).

A second approach is to use the $\tau_{Q,\phi}$ bound (see Section 4.2) to stop the search, which provides guarantees about the proportion of exact answers, rather than the distance error. It also depends on a single parameter, rather than two. To avoid the multiple-comparisons problem, we provide a single estimate of this bound at the very beginning of the search, allowing users to plan ahead their stopping strategy.

A third approach is to bound the probability $p_Q(t)$. Specifically, we stop the search when this probability ex-

ceeds a level $1-\phi$, where ϕ here represents the probability that the current progressive answer is not the exact. We experimentally assess the tradeoffs of these three stopping criteria.

4.4 Family-wise Error in Progressive k-NN Queries

A k-NN similarity search query aims to identify k data series as answers (not simply the k-th NN of Q). In most practical scenarios, an analyst is thus interested in stopping criteria that apply to all the progressive answers of a k-NN query.

We observe that similarity search algorithms always find the exact i-NN to a query before its exact $(i+1)$-NN. Therefore, our second ($\tau_{Q,\phi}$) and third ($p_Q(t)$) stopping criteria naturally apply to all k answers of a k-NN query. If the k-th answer is exact when the search stops, then we also know that answers of a rank lower are also exact.

In contrast, our first criterion on the relative distance error is optimistic. Stopping when the relative distance error $\epsilon_Q(t)$ of the k-th answer is lower than ϵ does not provide any guarantee about the relative distance error of lower-rank answers. To deal with this problem, we focus instead on the relative *family-wise* distance error, defined as follows:

$$\epsilon^f_Q(t) = \frac{d_{Q,R}(t)}{d^f_{Q,knn}(t)} - 1 \tag{8}$$

where the distance term $d^f_{Q,knn}(t) \leq d_{Q,knn}$ represents a k-NN distance that is corrected for the family-wide error at time t, such that:

$$d^f_{Q,knn}(t) = \frac{d_{Q,knn}}{\max\limits_{1\leq i\leq k}\{d_{Q,R_i}(t)/d_{Q,inn}\}} \tag{9}$$

Our goal now is to find an estimate $\hat{d}^f_{Q,knn}(t)$.

5 Prediction Methods

We now present our approach, called *ProS*. We use 1 synthetic and 3 real datasets (i.e., seismic, SALD, and deep1B) from past studies [47, 146] to showcase our methods. We further explain and use these datasets in Section 7 (see Table 2 for a summary of their characteristics) to evaluate our methods.

Our goal is to support reliable prediction with small training sets of queries. We are also interested in expressing the uncertainty of our predictions with well-controlled bounds, as discussed in the previous section. We thus focus on statistical models that capture a small number of generic relationships in the data. We first examine methods that assume constant information ($I_Q(t) = I_Q$). They are useful for providing an initial estimate *before* the search starts. We distinguish between query-sensitive methods, which take into account the query series Q, and query-agnostic methods, which provide a common estimate irrespective of Q ($I_Q = I$). Inspired by Ciacca et al. [31, 33], these methods serve as baselines to compare to a new set of progressive methods. Our progressive methods update information during the execution of a search, resulting in considerably improved predictions.

To simplify our analysis, we focus on 1-NN similarity search. At the end of the section, we explain how our analysis naturally extends to k-NN search.

5.1 Initial 1-NN Distance Estimates

We first concentrate on how to approximate the distribution function $h_{Q,0}(x)$ (see Equation 4), thus provide estimates before similarity search starts.

As Ciaccia et al. [31], we rely on witnesses, which are "training" query series that are randomly sampled from a dataset. Unlike their approach, however, we do not use the distribution of raw pairwise distances $F_Q(\cdot)$. Instead, for each witness, we execute 1-NN similarity queries with a fast state-of-the-art algorithm, such as iSAX2+ [23], or DSTree [134]. This allows us to derive directly the distribution of 1-NN distances and predict the 1-NN distance of new queries.

This approach has two main benefits. First, we use the tree structure of the above algorithms to prune the search space and reduce pre-calculation costs. Rather than calculating a large number of pairwise distances, we focus on the distribution of 1-NN distances with fewer distance calculations. Second, we achieve reliable and high-quality approximation with a relatively small number of training queries ($\approx 100-200$) independently of the dataset size (we report and discuss these results in Section 7).

Query-Agnostic Model (Baseline). Let $\mathcal{W} = \{W_j | j = 1..n_w\}$ be a set of $n_w = |\mathcal{W}|$ witnesses randomly drawn from the dataset. We execute a 1-NN similarity search for each witness and build their 1-NN distance distribution. We then use this distribution to approximate the overall (query-independent) distribution of 1-NN distances $g_n(\cdot)$ and its cumulative probability function $G_n(\cdot)$. This method is comparable to Ciaccia et al. [33] query-agnostic approximation method and serves as a baseline.

Query-Sensitive Model. Intuitively, the smaller the distance between the query and a witness, the better the 1-NN of this witness predicts the 1-NN of the query. We capture this relationship through a random variable that expresses the weighted sum of the 1-NN distance of all n_w witnesses:

$$dw_Q = \sum_{j=1}^{n_w}(a_{Q,j} \cdot d_{W_i,1nn}) \tag{10}$$

Similar to Ciacca et al. [31], we use weights $a_{Q,j}$ that decrease exponentially to the distance between the query Q

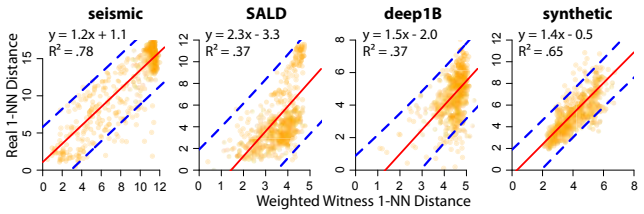


Fig. 3: Linear models (red lines) predicting the real 1-NN distance $d_{Q,1nn}$ based on the weighted witness 1-NN distance dw_Q for $exp = 5$. All models are based on $n_w = 200$ random witnesses and $n_r = 100$ queries, and tested on 500 queries (in orange). The blue dashed lines show the range of their 95% prediction intervals.

and the j^{th} witness:

$$a_{Q,j} = \frac{d(Q, W_j)^{-exp}}{\sum_{i=1}^{n_w} d(Q, W_i)^{-exp}} \quad (11)$$

Our tests have shown optimal results for exponents exp that are close to 5. For simplicity, we use $exp = 5$ for all our analyses. Additional tests have shown that the fit of the model becomes consistently worse if witnesses are selected with the GNAT algorithm [20, 31] (we omit these results for brevity). Therefore, we only examine random witnesses here.

We use dw_Q as predictor of the query's real 1-NN distance $d_{Q,1nn}$ and base our analysis on the following linear model:

$$d_{Q,1nn} = \beta \cdot dw_Q + c \quad (12)$$

Figure 3 shows the parameters of instances of this model for the four datasets of Table 2. We conduct linear regressions by assuming that the distribution of residuals is normal (Gaussian) and has equal variance.

Since the model parameters (β and c) and the variance are dataset specific, they have to be trained for each individual dataset. To train the model, we use an additional random sample of n_r *training queries* that is different from the sample of witnesses. Based on the distance of each training query Q_i from the witnesses, we calculate dw_{Q_i} (see Equation 10). We also run similarity search to find its 1-NN distance $d_{Q_i,1nn}$. We then use all pairs $(dw_{Q_i}, d_{Q_i,1nn})$, where $i = 1..n_r$, to build the model. The approach allows us to construct both point estimates (see Equation 10) and prediction intervals (see Figure 3) that provide probabilistic guarantees about the range of the 1-NN distance.

5.2 Progressive 1-NN Distance Estimates

So far, we have focused on initial 1-NN distance estimates. Those do not consider any information about the

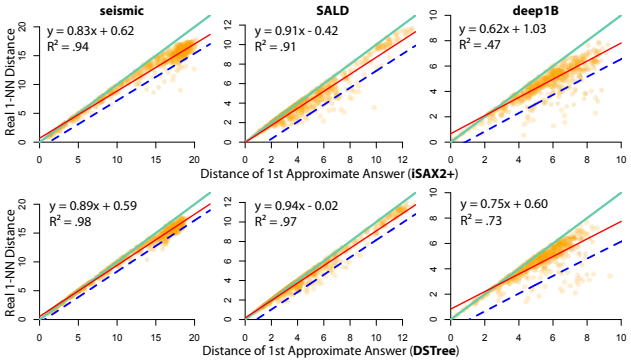


Fig. 4: Linear models (red solid lines) predicting the real 1-NN distance $d_{Q,1nn}$ based on the *first* approximate answer distance of iSAX2+ and DSTree. All models are trained with 200 queries. The 500 (orange) points in each plot are queries out of the training set. Green (solid) lines ($y = x$) are hard upper bounds, set by the approximate answer. Blue lines show the range of one-sided 95% prediction intervals that form probabilistic lower bounds.

partial results of a progressive similarity-search algorithm. Now, given Definition 1, the distance of a progressive result $d_{Q,R}(t_i)$ will never deteriorate and thus can act as an upper bound for the real 1-NN distance. The challenge is how to provide a probabilistic lower bound that is larger than zero.

Our approach relies on the observation that the approximate answers of index-based algorithms are generally close to the exact answers. Figure 4 illustrates the relationship between the true 1-NN distance and the distance of the first progressive (approximate) answer returned by iSAX2+ [23]. (The results for the DSTree index [134] that follows a completely different design from iSAX2+ are very similar; we omit them for brevity). We observe a strong linear relationship for both algorithms, especially for the DSTree index. We can express it with a linear model and then derive probabilistic bounds in the form of prediction intervals. As shown in Figure 4, the approach is particularly useful for constructing lower bounds. Those are clearly greater than zero and provide valuable information about the extent to which a progressive answer can be improved or not.

Since progressive answers improve over time and tend to converge to the 1-NN distance, we could take such information into account to provide tighter estimates as similarity search progresses. To this end, we examine different progressive prediction methods. They are all based on the use of a dataset of n_r training queries that includes information about all progressive answers of a similarity search algorithm to each query, including a timestamp and its distance.

Linear Regression. Let $t_1, t_2, ..., t_m$ be specific moments of interest (e.g., 100ms, 1s, 3s, and 5s). Given t_i, we can build

a time-specific linear model:

$$d_{Q,1nn} = \beta_{t_i} \cdot d_{Q,R}(t_i) + c_{t_i} \quad (13)$$

where $d_{Q,R}(t_i)$ is Q's distance from the progressive answer at time t_i. As an advantage, this method produces models that are well adapted to each time of interest. On the downside, it requires the pre-specification of a discrete set of time points, which may not be desirable for certain application scenarios. However, building such models from an existing training dataset is inexpensive, so reconfiguring the moments of interest at use time is not a problem.

The above model can be enhanced with an additional term $\beta \cdot dw_Q$ (see Equation 10) that takes witness information into account. However, this term results in no measurable improvements in practice, so we do not discuss it further.

Kernel Density Estimation. A main strength of the previous method is its simplicity. However, linearity is a strong assumption that may not always hold. Other assumption violations, such as heteroscedasticity, can limit the accuracy of linear regression models. As alternatives, we investigate non-parametric methods that approximate the density distribution function $h_{Q,t}(\cdot)$ based on multivariate kernel density estimation [41, 130].

As for linear models, we rely on the functional relationship between progressive and final answers. We represent this relationship as a 3-dimensional density probability function $k_Q(x, y, t)$ that expresses the probability that the 1-NN distance from Q is x, given that Q's distance from the progressive answer at time t is y. From this function, we derive $h_{Q,t}(x)$ by setting $y = d_{Q,R}(t)$.

We examine two approaches for constructing the function $k_Q(\cdot, \cdot, \cdot)$. As for linear models, we specify discrete moments of interest t_i and then use bivariate kernel density estimation [131] to construct an individual density probability function $k_Q(\cdot, \cdot, t_i)$. Alternatively, we construct a common density probability function by using 3-variate kernel density estimation. The advantage of this method is that it can predict the 1-NN distance at any point in time. Nevertheless, this comes with a cost in terms of precision (see Section 7).

The accuracy of kernel density estimation highly depends on the method that one uses to smooth out the contribution of points (2D or 3D) in a training sample. We use gaussian kernels, but for each estimation approach, we select bandwidths with a different technique. We found that the plug-in selector of Wand and Jones [131] works best for our bivariate approach, while the smoothed cross-validation technique [41] works best for our 3-variate approach.

Measuring Time. So far, we have based our analysis on time. Nevertheless, time (expressed in seconds) is not a reliable measure for training and applying models in practice. The reason is that time largely depends on the available computation power, which can vary greatly across different hardware settings. Our solution is to use alternative measures that capture the progress of computation without being affected by hardware and computation loads. One can use either the number of series comparisons (i.e., the number of distance calculations), or the number of visited leaves. Both measures can be easily extracted from the iSAX2+ [23], the DSTree [134], or other tree-based similarity-search algorithms. Our analyses in this paper are based on the number of visited leaves (*Leaves Visited*). We should note that for a given number of visited leaves, we only consider a single approximate answer, which is the best-so-far answer after traversing the last leaf.

5.3 Estimates for Exact Answers

We investigate two types of estimates for exact answers (see Section 4.2): (i) progressive estimates of the probability $p_Q(t)$ that the 1-NN has been found; and (ii) query-sensitive estimates of the time t_Q that it takes to find the exact answer. We base our estimations on the observation that queries with larger 1-NN distances tend to be harder, i.e., it takes longer to find their 1-NN. Now, since approximate distances are good predictors of their exact answers (see previous subsection), we can also use them as predictors of $p_Q(t)$ and t_Q.

Probability Estimation. Let $t_1, t_2, ..., t_m$ be moments of interest, and let $d_{Q,R}(t_i)$ be the distance of the progressive result at time t_i. We use logistic regression to model the probability $p_Q(t_i)$ as follows:

$$ln\frac{p_Q(t_i)}{1 - p_Q(t_i)} = \beta_{t_i} \cdot d_{Q,R}(t_i) + c_{t_i} \quad (14)$$

Again, we can use the number of visited leaves to represent time. Figure 5 presents an example for the seismic dataset, where we model the probability of exact answers for four points in time (when 64, 256, 1024, and 4096 leaves are visited). We observe that over time, the curve moves to the right range of distances, and thus, probabilities increase.

Note that we have considered other predictors as well (such as the time passed since the last progressive answer), but they did not offer any predictive value.

Time Bound Estimation. As we explained in Section 4.3, we provide a single estimate for t_Q at the very beginning of the search. Figure 6 (top) illustrates the relationship between the distance of the first approximate answer and the number of leaves (in logarithmic scale) at which the 1-NN is found. We observe that the correlation between the two variables is rather weak. However, we can still extract meaningful query-sensitive upper bounds, shown as green lines. To construct such bounds, we use quantile regression [71]. This method allows us to directly estimate the $1-\phi$ quantile

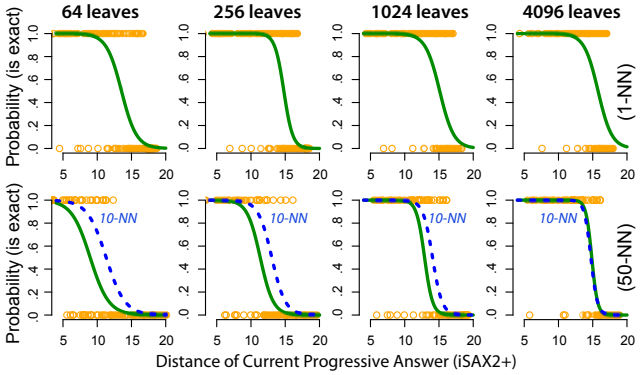


Fig. 5: Estimating the probability of exact answers with 100 training queries based on the current 1-NN (top) and 50-NN (bottom) progressive answer (seismic dataset). We show the logistic curves (in green) at different points in time (64, 256, 1024, and 4096 leaves) and 200 test queries (in orange). For comparison, we also show the logistic curve for the 10-NN (blue dashed line).

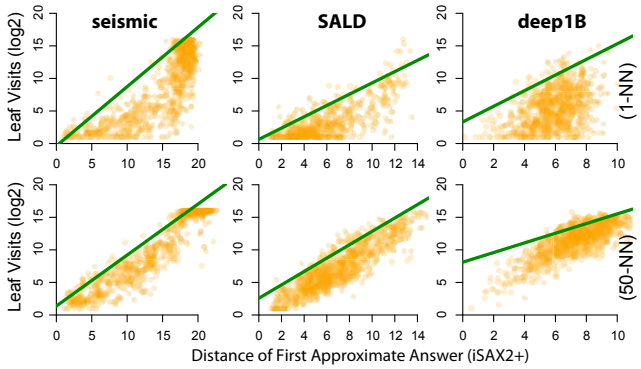


Fig. 6: Upper time bounds ($\phi = .05$) for 1-NN (top) and 50-NN (bottom) answers. Bounds (in green) are constructed from 100 random queries through quantile regression, where we estimate the 95% quantile of the logarithm of leaf visits as a function of the distance of the 1st approximate answer.

of the time needed to find the exact answer, i.e., derive the upper bound $\tau_{Q,\phi}$. As a shortcoming, the accuracy of quantile regression is sensitive in small samples. Nevertheless, we show that 100 training queries are generally enough to ensure high-quality results.

Example. Figure 2 presents an example that illustrates how the above methods can help users assess how far from the 1-NN their current answers are. We use a variation of pirate plots [110] to visualize the 1-NN distance estimate $\hat{d}_{Q,1nn}(t)$ and the relative error estimate $\hat{\epsilon}_Q(t)$ by showing their probability density distribution and their 95% prediction interval. We also communicate the probability $p_Q(t_i)$ and a probabilistic bound $\tau_{Q,\phi}$ ($\phi = .05$) after the first visited leaf. The initial distance estimate based on witnesses is rather wide. However, prediction intervals become tighter as soon as search starts. In particular, the upper bound of the error estimate drops below 10% within 1.1sec, while the probability that the current answer is exact is estimated as 98% after 15.7sec (total query execution time for this query is 75.2sec). In this example, the 1-NN is found in 3.8sec.

5.4 Progressive Estimates for k-NN Similarity Search

The predictions methods presented above naturally extend to the general case of k-NN search. As Figure 5 shows, exact answers for larger k ranks are found later in time. Still, distance is a good predictor of whether a progressive answer is exact. We observe that at earlier steps, uncertainty is higher for large k ranks, but as more leaves are visited, the prediction quality of the logistic model improves.

Figure 5 (bottom) presents how upper time-bound estimation extends to 50-NN. We can still derive useful bounds based on the distance of the very first approximate answer. Interestingly, the correlation between this distance and the logarithm of visited leaves is stronger now. We could eventually use this behavior to construct meaningful lower time bounds for k-NN search.

For k-NN search, we evaluate the family-wise error of distance estimates $\hat{\epsilon}^f_Q(t)$ based on Equation 8. We apply the same prediction methods (see Section 5.2) but our dependent variable is now $d^f_{Q,knn}$ (rather than $d_{Q,1nn}$). We use Equation 9 to calculate $d^f_{Q,knn}$ for our training dataset.

5.5 Dynamic Time Warping (DTW)

The data series indexes we employ, i.e., iSAX2+ and DSTree, originally supported only the Euclidean distance. We modified their query answering algorithms to provide support for DTW based on the ideas proposed in [70].

First, we find U and L, the upper and lower envelopes that bound the query Q according to the Sakoe-Shiba constraint [117] using the algorithm proposed in [77]. Then, for each index, we calculate $\hat{U}$ and $\hat{L}$ the summarizations of U and L, and we derive $MinDist(Q, N)$, the lower bounding distance between the summarized envelopes $\hat{U}$ and $\hat{L}$ of the query Q and an index node N. Note that we probe the index using the summarized envelopes $\hat{U}$ and $\hat{L}$ rather than the query Q itself. The distance $MinDist(Q, N)$ is guaranteed to lower bound LB_{Keogh}[4], which itself lower bounds DTW.

$$LB_{Keogh}(Q, C) = \sqrt{\sum_{i=1}^{n} \begin{cases} (c_i - U_i)^2 & \text{if } c_i > U_i \\ (L_i - c_i)^2 & \text{if } c_i < L_i \\ 0 & otherwise \end{cases}} \quad (15)$$

[4] We note that other lower bounds for DTW can be used as well, such as LB_Improved [77]. Even though LB_Improved can produce tighter bounds, previous experiments have resulted in higher query answering times due to the additional computations it involves [105].

We do not calculate LB_{Keogh} directly because it is also computationally expensive. As Equation 15 and Figure 7a indicate, it requires calculating distances between the individual n high-dimensional points of the candidate C and the envelopes U and L (if some points of the candidate fall inside the query envelope, then their distance is zero). On the other hand, $MinDist(Q, N)$ is faster to compute because it consists of the distances between the segments of a node N and the segments of the summarized envelopes. We use $\bar{C}$, $\bar{Q}$, $\hat{U}$ and $\hat{L}$ to refer to the summaries of C, Q, L and U respectively. The specific representation used for the summaries can be inferred from the context.

iSAX2+. Since iSAX2+ is based on PAA, we use the same formulas as those in [70] to derive $\hat{U}$ and $\hat{L}$ (Equations 16-17), the special piecewise aggregate approximations of U and L for the i^{th} segment of $\bar{Q}$, and calculate $MinDist_{PAA}(Q, N)$ (Equation 18). We consider a PAA summarization using M segments.

$$\hat{U}_i = max(U_{(\frac{n}{M})(i-1)+1}, \ \dots \ , U_{(\frac{n}{M})(i)}) \tag{16}$$

$$\hat{L}_i = max(L_{(\frac{n}{M})(i-1)+1}, \ \dots \ , L_{(\frac{n}{M})(i)}) \tag{17}$$

$$LB_{PAA}(Q, \bar{C}) = \sqrt{\frac{n}{M}} \sqrt{\sum_{i=1}^{M} \begin{cases} (\bar{c}_i - \hat{U}_i)^2 & \text{if } \bar{c}_i > \hat{U}_i \\ (\hat{L}_i - \bar{c}_i)^2 & \text{if } \bar{c}_i < \hat{L}_i \\ 0 & otherwise \end{cases}} \tag{18}$$

$$MinDist_{PAA}(Q, N) = \sqrt{\frac{n}{M}} \sqrt{\sum_{i=1}^{M} \begin{cases} (l_i - \hat{U}_i)^2 & \text{if } l_i > \hat{U}_i \\ (\hat{L}_i - h_i)^2 & \text{if } h_i < \hat{L}_i \\ 0 & otherwise \end{cases}} \tag{19}$$

Such that l_i and h_i are the lower and higher endpoints of the major diagonal of R, the smallest rectangle that spatially contains the ith segment of all data series in N. Figure 7b illustrates the LB_{PAA} distance between Q and $\bar{C}$.

DSTree. For DSTree, we propose new upper and lower bounding envelopes, $\hat{U}$ and $\hat{L}$, and a new lower bounding distance $MinDist_{EAPCA}(Q, N)$. Given an EAPCA representation with M segments and m_i is the right endpoint of segment i ($m_1 < ... < m_i < ... < m_M = n$), the upper and lower EAPCA envelopes of segment i of $\bar{Q}$ are defined as follows:

$$\hat{U}_i = max(U_{m_{i-1}+1}, U_{m_{i-1}+2}, \ \dots \ , U_{m_i}) \tag{20}$$

$$\hat{L}_i = min(L_{m_{i-1}+1}, L_{m_{i-1}+2}, \ \dots \ , L_{m_i}) \tag{21}$$

The EAPCA lower bounding distance between Q and $\bar{C}$ is defined as:

$$LB_{EAPCA}(Q, \bar{C}) = \sqrt{\sum_{i=1}^{M} (m_i - m_{i-1}) \, a_i} \tag{22}$$

Where:

$$a_i = \begin{cases} (\bar{c}_i - \hat{U}_i)^2 & \text{if } \bar{c}_i > \hat{U}_i \\ (\hat{L}_i - \bar{c}_i)^2 & \text{if } \bar{c}_i < \hat{L}_i \\ 0 & otherwise \end{cases} \tag{23}$$

Figure 7c shows the LB_{EAPCA} distance between Q and $\bar{C}$. The proof that $LB_{EAPCA}(Q, \bar{C}) \leq LB\_Keogh(Q, C)$ is a straightforward extension of the proof in Proposition 1 in [70].

Consider an EAPCA index node N containing a set of data series $Y_1, \dots, Y_l$ with synopsis $\mathbf{Z} = (z_1, z_2, \dots, z_M)$ where $z_i = (\mu_i^{min}, \mu_i^{max})$ where $\mu_i^{min} = min(\mu_i^{Y_1}, \dots, \mu_i^{Y_l})$ and $\mu_i^{max} = max(\mu_i^{Y_1}, \dots, \mu_i^{Y_l})$. Then, the lower bounding distance between Q and node N is defined as:

$$MinDist_{EAPCA}(Q, N) = \sqrt{\sum_{i=1}^{M} (r_i - r_{i-1})(LB_i)} \tag{24}$$

Such that

$$LB_i = \begin{cases} (\mu_i^{min} - \hat{U}_i)^2 & \text{if } \mu_i^{min} > \hat{U}_i \\ (\hat{L}_i - \mu_i^{max})^2 & \text{if } \mu_i^{max} < \hat{L}_i \\ 0 & otherwise \end{cases} \tag{25}$$

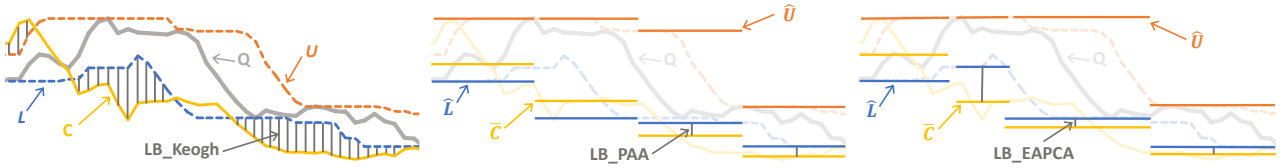
The proof that $MinDist_{EAPCA}(Q, N) \leq LB_{EAPCA}(Q, \bar{Y_j}) \ \forall \ 1 \leq j \leq l$ is a straightforward extension of the proof in Theorem 2 in [134].

Note that although the DSTree exploits the standard deviation of the points in each segment to produce a tighter lower bound between a query Q and a node N, we only use the mean. The reason is that the standard deviations of the points belonging to each segment of the EAPCA upper and lower envelopes are zero, and thus cannot contribute to the lower bound.

6 Progressive k-NN Classification

For k-NN classification, we can use again a progressive similarity search algorithm. At any given time, we take its progressive answer and use it to infer the *progressive class.* However, since we are now interested in the class of the data series that serves as query, the notion of "approximation" is not relevant anymore – the class can be either correct or wrong. A progressive answer in this case is only interesting if it returns the correct class, or alternatively, if it returns the same class (correct or not) as the non-progressive algorithm, i.e., the final, answer. In this case, we say that the class is the *exact class*.

Our goal is now to provide guarantees for this class, rather than for the distance of the progressive answer. More formally, given a data series Q as query, we run again a progressive k-NN search. At each time t, we take the most common class $c_Q(t)$ among the progressive k nearest neighbors



(a) LB_{Keogh} (b) LB_{PAA} (c) LB_{EAPCA}

Fig. 7: Envelopes U and L of query Q with warping size 10%. Each dark gray vertical line contributes to the lower bounding distance between Q and a candidate answer C. Distances are calculated by taking the square root of the sum of the squares of the lengths of the gray lines. In the case of LB_{PAA} and LB_{EAPCA}, the squares of the lengths are scaled by the number of points in each segment.

of Q. We then assess the probability $p_{c_Q}(t)$ that the class c_Q of the exact answer is found, where as for k-NN similarity search, we use information $I_Q(t)$:

$$p_{c_Q}(t) = Pr\{c_Q(t) \equiv c_Q \mid I_Q(t)\} \quad (26)$$

Extending our ProS approach, we describe two solutions on how to either bound, or directly estimate this quantity.

6.1 Bounding the Probability of Exact Class

We can easily infer that $p_{c_Q}(t) \geq p_Q(t)$ (see Equation 14), i.e., the probability that the current progressive class is exact is at least as high as the probability that the current progressive k-NN is exact. In other words, although the similarity search algorithm may have not yet found the exact answer to the k-NN similarity search query, the class can be the exact.

A direct implication of the above is that the exact-answer probabilistic guarantees that we presented in Section 4.2 can be also considered as guarantees for the exact class. Likewise, the probability and time-bound stopping criteria presented in Section 4.3 can also apply as stopping criteria for k-NN classification. Nevertheless, they are stricter, more conservative and result in reduced time savings. Instead, we update the stopping criteria by simply replacing the parameter ϕ by ϕ_c, where ϕ_c represents the probability that the current progressive class is not the exact.

6.2 Estimating the Probability of Exact Class

We consider again m moments of interest $t_1, t_2, ..., t_m$. At each moment t_i, we estimate $p_{c_Q}(t_i)$ by using three predictors: (i) the distance $d_{Q,R}(t_i)$ of the k-NN, (ii) the current class $c_Q(t_i)$, and (iii) the extent to which the current k answers agree on this class. The latter is quantified as follows:

$$a(t_i) = \frac{n_{c_Q(t_i)} - 1}{k - 1} \quad (27)$$

where $n_{c_Q(t_i)}$ is the number of occurrences of $c_Q(t_i)$ among the k nearest neighbors returned by the progressive search ($k > 1$). We can the use these predictors to build a linear logistic regression model as in Equation 14.

Note that not all three predictors are always relevant. For example, if the number of available classes is large, information about the current class has no predictive value unless we use a much larger set of training queries. We have tested additional variables, such as the ones that evaluate the stability of $c_Q(t_i)$ over time, but we did not find them to be good predictors.

Table 2: Experimental datasets for similarity search.

Name	Description	Num of series	Series length
1. synthetic	random walks	100M/10M	256
2. seismic [122]	seismic records	100M/10M	256
3. SALD [127]	MRI data	200M/20M	128
4. deep1B [128]	image descriptors	267M/27M	96
5. PhysioNet [58]	ECG recordings	20M/10M	256

7 Experimental Evaluation

Environment. All experiments were run on a Dell T630 rack server with two Intel Xeon E5-2643 v4 3.4Ghz CPUs, 512GB of RAM, and 3.6TB (2 x 1.8TB) HDD in RAID0.

Implementation. Our estimation methods were implemented in R. We use R's *lm* function to carry our linear regression, the *ks* library [42] for multivariate kernel density estimation, and the *quantreg* library [2] for quantile regression. We use a grid of 200×200 points to approximate a 2D density distribution and a grid of $60 \times 180 \times 180$ points to approximate a 3D density distribution. Source code and datasets are in [1].

Datasets. For the evaluation of the progressive similarity search techniques, we used 1 synthetic and 3 real datasets from past studies [47, 146], as well as an additional real dataset, PhysioNet [58]. All datasets are *100GB* in size with cardinalities and lengths reported in Table 2. For our experiments with DTW, however, we used a smaller subset of these datasets (*10GB* in size), since running them on the original datasets was extremely expensive.

Synthetic data series were generated as random walks (cumulative sums) of steps that follow a Gaussian distribution (0,1). This type of data has been extensively used in the past [23, 49, 147] and models the distribution of stock market prices [49]. The IRIS seismic dataset [122] is a collection of seismic instrument recordings from several stations worldwide (100M series of length 256). The SALD neuroscience dataset [127] contains MRI data (200M series of length 128). The image processing dataset, deep1B [128], contains vectors extracted from the last layers of a convolutional neural network (267M series of length 96). The PhysioNet dataset [58] contains ECG data (20 million series of length 256).

The above datasets are not annotated. In order to evaluate the progressive k-NN classification techniques, we use the datasets in Table 3.

The Cylinder-Bell-Funnel (CBF) dataset [116] is a synthetic dataset that has been used extensively in the data series classification community, consisting of data series belonging to one of three classes: cylinders, bells and funnels. Instances of each class are generated randomly with Gaussian noise added, such that each series has a fixed length of 128, but the onset and duration of each pattern varies randomly. An amplitude parameter is also used to control the difficulty of the dataset, where the smaller the amplitude, the less distinct the data series in different classes, thus the harder the classification task. We used the amplitude values 1 and 3 to generate the CBF1 and CBF3 datasets, respectively. We used subsets of the CBF1 dataset ranging from 2M to 200M series and two CBF3 subsets of 20M and 200M series each.

The SITS dataset [103] is a remote sensing dataset (i.e., derived from sensor measurements by satellites orbiting Earth) containing 1M series of size 46 points each, and 24 classes. Each series corresponds to 1 pixel of satellite images of the earth, taken at 46 time instances. We drop the last data point for every time series so that we have series of length 45, which can be more efficiently indexed by iSAX2+ with 9 segments of length 5 (remember that all SAX segments should have equal length).

For ImageNet [37], image embeddings were generated using a pre-trained EfficientNetB1 [124] neural network. We applied a global average pooling to the last layers of the network to produce a single vector of 1280 real values per image. The dataset contains a total of 1361 distinct classes. For our experiments, we use the vectors of ImageNet's training images ($\approx$ 1.3M images) as the series dataset and the vectors of its testing images (50K images from 1000 classes) to sample our queries. To also test our methods on a smaller number of classes, we used WordNet's [91] hierarchical structure and grouped the original classes ("synsets") to 30 larger classes that correspond to the leaf nodes of the hierarchy used by Huang et al. [63].

Table 3: Experimental datasets for k-NN classification

Name	Description	Num of series	Length	Classes
CBF 1	synthetic	2M-200M	128	3
CBF 3	synthetic	20M / 200M	128	3
SITS	satellite images	1M	45	24
ImageNet	image embeddings	1.3M	1280	1361 / 30

Measures. We use the following measures to assess the estimation quality of each method and compare their results:

Coverage Probability: It measures the proportion of the time that the prediction intervals contain the true 1-NN distance. If the confidence level of the intervals is $1-\theta$, the coverage probability should be close to $1-\theta$. A low coverage probability is problematic. In contrast, a coverage probability that is higher than its nominal value (i.e., its confidence level) is acceptable but can hurt the intervals' precision. In particular, a very wide interval that always includes the true 1-NN distance (100% coverage) can be useless.

Prediction Intervals Width: It measures the size of prediction intervals that a method constructs. Tighter intervals are better. However, this is only true if the coverage probability of the tighter intervals is close to or higher than their nominal confidence level. Note that for progressive distance estimates, we construct one-sided intervals. Their width is defined with respect to the upper distance bound $d_{Q,R}(t)$.

Root-Mean-Squared Error (RMSE): It evaluates the quality of point (rather than interval) estimates by measuring the standard deviation of the true 1-NN distance values from their expected (mean) values.

To evaluate the performance of our stopping criteria, we further report on the following measures:

Time Savings: Given a load of queries and a stopping criterion, it measures the time saved as a percentage of the total time needed to complete the search without early stopping.

Exact Answers: It measures the number of exact answers as a percentage of the total number of queries. For k-NN classification, we report on *exact classes*, where we assess the percentage of queries for which the progressive class is the final one.

Accuracy Ratio: We also measure the ratio of the accuracy of k-NN classification with early stopping to the accuracy of exact k-NN classification.

Validation Methodology. To evaluate the different methods, we use a Monte Carlo cross-validation approach that consists of the following steps. For each dataset, we randomly draw two disjoint sets of data series $\mathcal{W}_{pool}$ and $\mathcal{T}_{pool}$ and pre-calculate all distances between the series of these two sets. The first set serves as a pool for drawing random sets of witnesses (if applicable), while the second set serves as a pool for randomly drawing training (if applicable) and

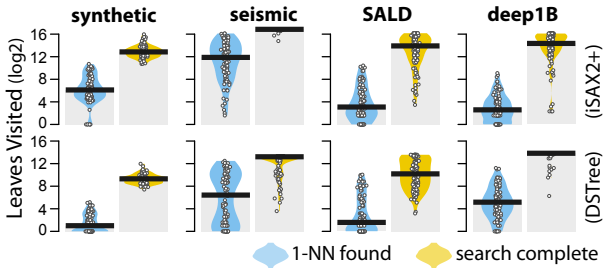


Fig. 8: Distribution (over 100 queries) of the number of leaves visited (in log_2 scale) until finding the 1-NN (light blue) and completing the search (yellow). The thick black lines represent medians.

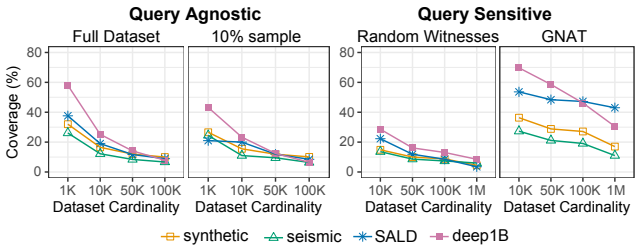


Fig. 9: Coverage probabilities of query-agnostic (left) and query-sensitive (right) methods of Ciaccia et al. [31, 32] for 95% confidence level. We use 500 witnesses for the query-sensitive methods. We show best-case results (with the best exp: 3, 5, 12, or adaptive).

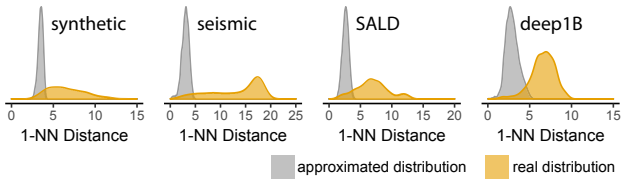


Fig. 10: Real distribution of 1-NN distances and its query-agnostic approximation based on Ciaccia and Patella [32]. All datasets contain 100K series.

testing queries. At each iteration, we draw n_w witnesses ($n_w = 50$, 100, 200, or 500) and/or n_r training queries ($n_r = 50$, 100, or 200) from $\mathcal{W}_{pool}$ and $\mathcal{T}_{pool}$, respectively. We also draw $n_t = 200$ testing queries from $\mathcal{T}_{pool}$ such that they do not overlap with the training queries. We train and test the evaluated methods and then repeat the same procedure $N = 100$ times, where each time, we draw a new set of witnesses, training, and testing queries. Thus, for each method and condition, our results are based on a total of $N \times n_t = 20K$ measurements.

For all progressive methods, we test the accuracy of their estimates after the similarity search algorithm has visited 1 (2^0), 4 (2^2), 16 (2^4), 64 (2^6), 256 (2^8), and 1024 (2^{10}) leaves. Figure 8 shows the distributions of visited leaves for 100 random queries for all four datasets.

7.1 Results on Prediction Quality

Previous Approaches. We first evaluate the query-agnostic and query-sensitive approximation methods of Ciaccia et al. [31, 32]. To assess how the two methods scale with and without sampling, we examine smaller datasets with cardinalities of up to 1M data series (up to 100K for the query-agnostic approach). Those datasets are derived from the initial datasets presented in Table 2 through random sampling. Such smaller dataset sizes allow us to derive the full distribution of distances without sampling errors, while they are sufficient for demonstrating the behavior of the approximation methods as datasets grow.

Figure 9 presents the coverage probabilities of the methods. The behavior of query-agnostic approximation is especially poor. Even when the full dataset is used to derive the distribution of distances, the coverage tends to drop below 10% for larger datasets (95% confidence level). This demonstrates that the approximated distribution of 1-NN distances completely fails to capture the real one. Figure 10 compares the real to the approximated distributions for datasets of 100K series. We observe that the method largely underestimates the 1-NN distances for all four datasets.

Results for the query-sensitive method are better, but coverage is still below acceptable levels. Figure 9 presents results for $n_w = 500$ witnesses. Note that our further tests have shown that larger numbers of witnesses result in no or very little improvement, while Ciacca et al. [31] had tested a maximum of 200 witnesses. To weight distances (see Equation 11), we tested the exponent values $exp = 3$, 5, and 12, where the first two were also tested by Ciacca et al. [31], while we found that the third one gave better results for some datasets. We also tested the authors' adaptive technique. Figure 9 presents the best result for each dataset, most often given by the adaptive technique.

We observe that the GNAT method results in clearly higher coverage probabilities than the fully random method. This result is somehow surprising because Ciacca et al. [31] report that the GNAT method tends to become less accurate than the random method in high-dimensional spaces with more than eight dimensions. Even so, the coverage probability of the GNAT method is largely below its nominal level. In all cases, it tends to become less than 50% as the cardinality of the datasets increases beyond 100K, while in some cases, it drops below 20% (synthetic and seismic).

For much larger datasets (e.g., 100M data series), we expect the accuracy of the above methods to become even worse. We conclude that they are not appropriate for our purposes, thus we do not study them further.

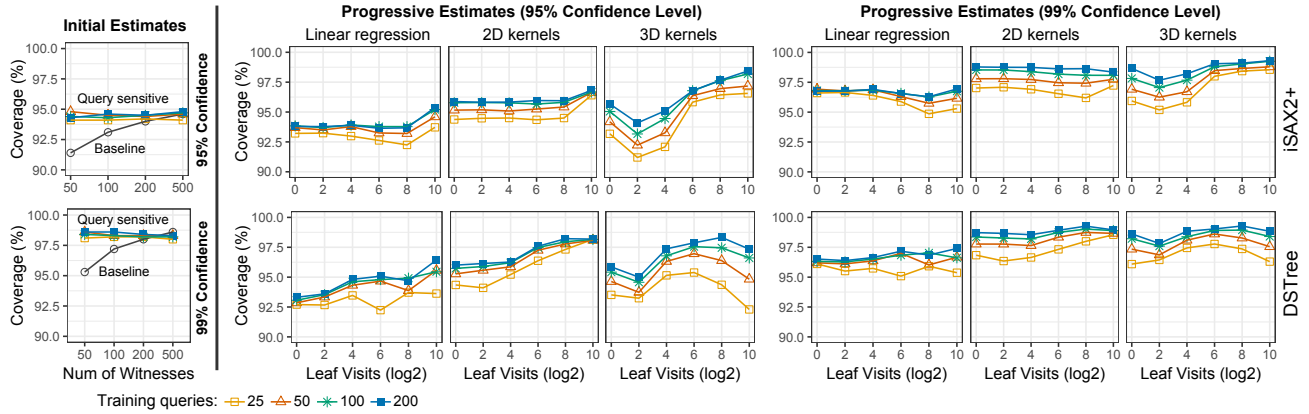


Fig. 11: Coverage probabilities of our estimation methods for 95% and 99% confidence levels. We show averages for the four datasets (synthetic, seismic, SALD, deep1B) and for 25, 50, 100, and 200 training queries.

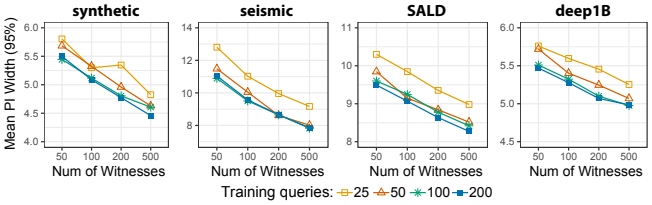


Fig. 12: The mean width of the 95% PI for the witness-based query-sensitive method in relation to the number of witnesses and training queries.

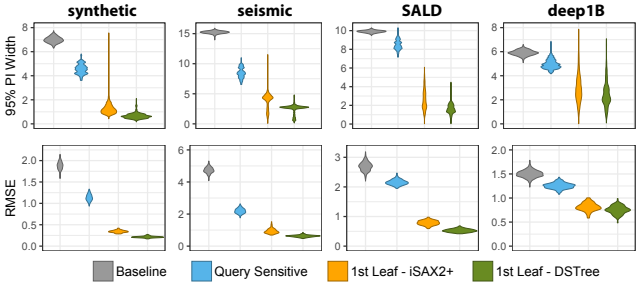


Fig. 13: Violin plots showing the distribution of the width of 95% prediction intervals (top) and the distribution of the RMSE of expected 1-NN distances (bottom). We use $n_w = 500$ (baseline and query-sensitive method) and $n_r = 100$ (query-sensitive method and 2D kernel model for the 1st approximate answer).

Quality of Distance Estimates. We evaluate the coverage probability of 1-NN distance estimation methods for confidence levels 95% ($\theta = .05$) and 99% ($\theta = .01$). Figure 11 presents our results. The coverage of the *Baseline* method reaches its nominal confidence level for $n_w = 200$ to 500 witnesses. In contrast, the *Query-Sensitive* method demonstrates a very good coverage even for small numbers of witnesses ($n_w = 50$) and training queries ($n_r = 25$). However, as Figure 12 shows, more witnesses increase the precision of prediction intervals, i.e., intervals become tighter while they still cover the same proportion of true 1-NN distances. Larger numbers of training queries also help.

The coverage probabilities of progressive estimates (Figure 11-Right) are best for the 2D kernel density approach, very close to their nominal levels. Linear regression leads to lower coverage, while the coverage of the 3D kernel density approach is more unstable. We observe that although the accuracy of the models drops in smaller training sets, coverage levels can still be considered as acceptable even if the number of training queries is as low as $n_r = 25$.

Figure 13 compares the quality of initial and early (i.e., based on first approximate answer) estimates provided by different techniques: (i) Baseline, (ii) Query-Sensitive method, (iii) 2D kernel density estimate for iSAX2+, and (iii) 2D kernel density estimate for DSTree. For all comparisons, we set $n_w = 500$ and $n_r = 100$. For these parameters, the coverage probability of all methods is close to 95%. We evaluate the width of their 95% prediction intervals and RMSE. We observe similar trends for both measures, where the query-sensitive method outperforms the baseline. We also observe that estimation based on the first approximate answer (at the first leaf) leads to radical improvements for all datasets. Overall, the DSTree index gives better estimates than iSAX2+.

As shown in Figure 14, progressive answers lead to further improvements. The RMSE is very similar for all three estimation methods, which means that their point estimates are equally good. Linear regression results in the narrowest intervals, which explains the lower coverage probability of this method. Overall, 2D kernel density estimation provides the best balance between coverage and interval width.

Sequential Tests. We assess how multiple sequential tests (refer to Section 4) affect the coverage probability of 1-NN distance prediction intervals. We focus on 2D kernel density estimation ($n_r = 100$), which gives the best coverage (see Figure 11). We examine the effect of (i) three sequential tests

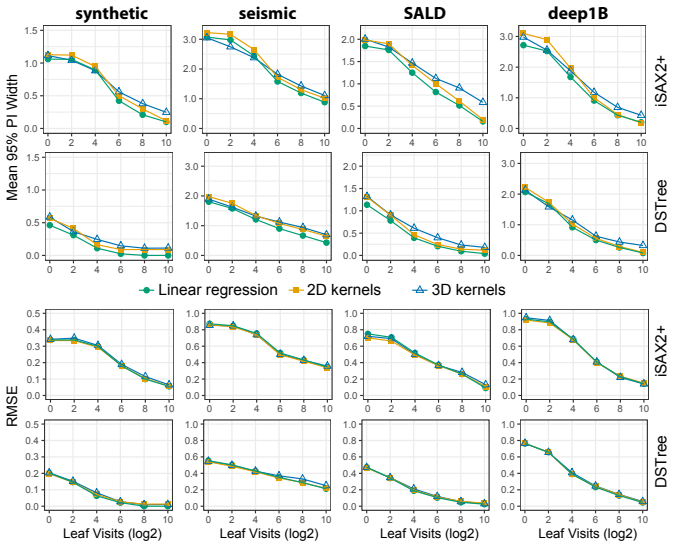


Fig. 14: Progressive models: Mean width of 95% prediction intervals of 1-NN distance estimates and RMSE. Results are based on $n_r = 100$ training queries.

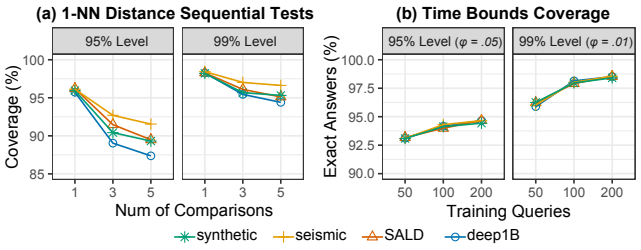


Fig. 15: (a) Effect of 3 and 5 sequential tests on the coverage of 95% and 99% prediction intervals. We use 2D kernels with $n_r = 100$. (b) Coverage of exact answers for time upper bounds (95% and 99% conf. levels).

when visiting 1, 512, and 1024 leaves, and (ii) five sequential tests when visiting 1, 256, 512, 768, and 1014 leaves. We count an error if at least one of the three, or five progressive prediction intervals do not include the true 1-NN distance.

As results for DSTree and iSAX2+ were very close, we report on their means (see Figure 15(a)). The coverage of 95% prediction intervals drops from over 95% to about 90% for five tests (higher for seismic and lower for deep1B). Likewise, the coverage of their 99% prediction intervals drops to around 95%. These results provide rules of thumb on how to correct for multiple sequential tests, e.g., use a 95% level in order to guarantee a 90% coverage in 5 sequential tests. Notice, however, that such rules may depend on the estimation method and the time steps at which comparisons are made. An in-depth study of this topic is part of our future work.

Time Bounds for Exact Answers. We are also interested in the quality of time guarantees for exact answers (refer to

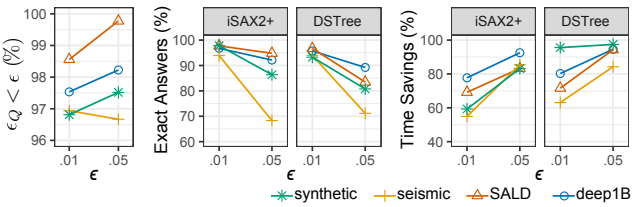


Fig. 16: Evaluation of the stopping criterion that bounds the distance error ($\epsilon_Q < \epsilon$). We use 95% prediction intervals ($\theta = .05$) and $n_r = 100$ training queries.

Section 4.2). We evaluate the coverage of our time bounds for 50, 100, and 200 training queries for confidence levels 95% ($\phi = .05$) and 99% ($\phi = .01$). Figure 15(b) summarizes our results. We observe that coverage is good for training samples of $n_r \geq 100$, but drops for $n_r = 50$.

7.2 Results on Time Savings

We compare our stopping criteria (see Section 4.3) and assess the time savings they offer. Figure 16 shows results for our first criterion that bounds the distance error. We consider 16 discrete and uniform moments t_i, where t_{16} is chosen to be equal to the maximum time it takes to find an exact answer in the training sample. For each t_i, we train an individual 2D kernel density and use 95% prediction intervals ($\theta = .05$) for estimation. The coverage (ratio of queries for which $\epsilon_Q < \epsilon$) exceeds its nominal level (95%) for all datasets, which suggests that results might be conservative. The reason is that stopping could only occur at certain moments. For higher granularity, one can use a larger number of discrete moments. The ratio of exact answers is close to 95% for $\epsilon = .01$ but becomes unstable for $\epsilon = .05$, dropping to as low as 70% for the seismic dataset. On the other hand, this results in considerable time savings, especially for DSTree: higher than 90% for the synthetic, SALD, and deep1B datasets.

Figure 17 compares the two stopping criteria that control the ratio of exact answers. For the probability criterion, we consider again 16 discrete moments to stop the search, as above. The time-bound criterion results in mean exact answer ratios that are very close to nominal levels, while the probability criterion is rather conservative. However, the time gains of the two techniques are comparable. For iSAX2+, the probability criterion achieves both a higher accuracy and higher time savings than the probability criterion. In contrast, both criteria lead to similar time savings for DSTree, reducing query times by up to 95%.

Training Costs vs. Gains. Training linear models with 100 queries is instantaneous, while learning $16 - 20$ density functions with 2D kernel density estimation takes no more than 4-6 seconds on a regular laptop. Of course, our ap-

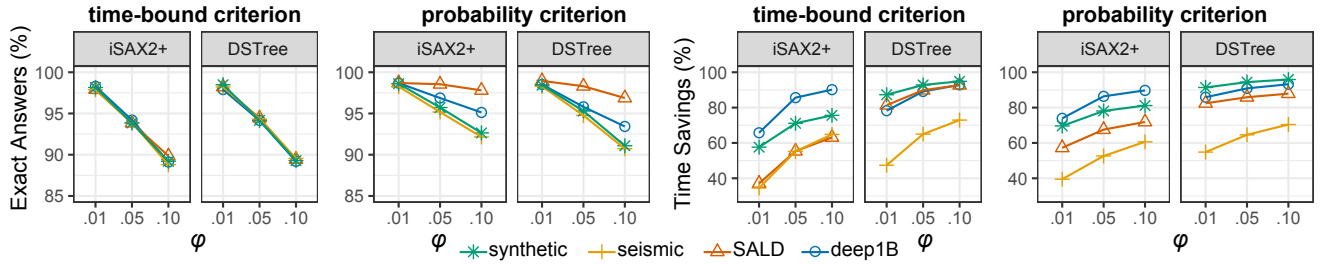


Fig. 17: Evaluation of stopping criteria that bound (ϕ) the probability/ratio of non-exact answers. We measure their ratio of exact answers and their time savings (%). For all conditions, we use $n_r = 100$ training queries.

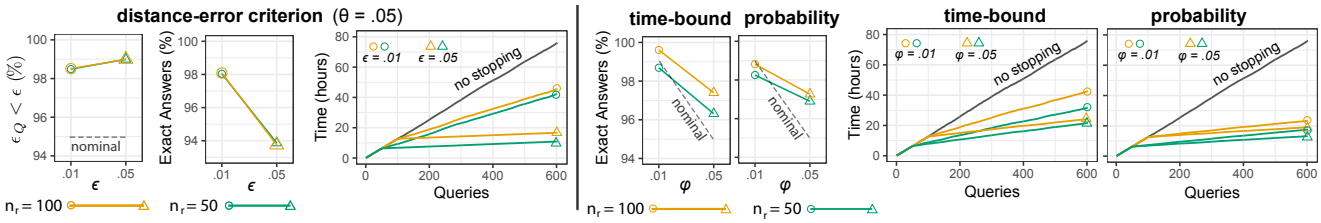


Fig. 18: Performance of our stopping criteria for a real workload of 600 queries (deep1B dataset and DSTree). We draw $n_r = 50$ or 100 random queries for training. We then apply a criterion to the remaining queries. Answers with $\epsilon_Q < \epsilon$ and exact ones (%) are measured for those "testing" queries. (We report means over 100 repetitions.)

proach requires the full execution of the training queries. For a detailed analysis of the costs of exact similarity search with iSAX2+ and DSTree, we refer the reader to the results of Echihabi et al. [47]. Depending on the size and type of the dataset, processing 100 queries can take some dozens of minutes (50 GB datasets), or several hours (250 GB datasets). Nevertheless, the higher this initial training cost, the higher the benefit is when users later execute their queries.

Figure 18 shows the results for the first 600 queries extracted from a real query workload that comes with the deep1B dataset. (Experiment conducted on a server with two Intel Xeon E5-2650 v4 2.2GHz CPUs, 75GB of RAM.) Results are based on 100 repetitions; each time we draw at random 50, or 100 queries for training. We then apply our stopping criteria to accelerate the remaining queries.

The results show that our approach leads to significant performance improvements, while coverage (exact answers, or answers with $\epsilon_Q < \epsilon$) is very close to, or higher than the nominal levels, even with training sizes of only 50 queries. For example, this workload of 600 queries would normally take 76 hours to execute with the DSTree index, but we can execute it in less than 20 hours (probability criterion; including training time), while achieving an average coverage of more than 95% exact answers. Finally, we note that, as the trends in the graphs show, the time savings and speedup offered by our progressive similarity search techniques will increase as the size of the query workload increases.

7.3 Results for k-NN Similarity Search

We evaluate how well our approach generalizes to k-NN similarity search. We follow the same experimental method but focus on the performance of our stopping criteria. In addition to the four datasets that we used earlier, we also test our approach on the 20M-series PhysioNet dataset (reported in Table 2). Figure 19 summarizes our evaluation results. We first show the average savings in time if stopping was optimally performed by an oracle that knows when a k-NN is found (Figure 19a). We observe that optimal time savings deteriorate as k increases. This is especially the case for the seismic dataset, e.g., savings reduce from 87% ($k = 1$) to 45% ($k = 100$) when iSAX2+ is used.

For the distance-error criterion, we report results for $\theta = .05$ and $\epsilon = .05$ (Figure 19b), where we now control and evaluate the relative family-wise distance error ϵ_Q^f (see Equation 8). For iSAX2+, time savings are stable and high (above 70%) across the full range of k values. This comes at the cost of a decreasing ratio of exact answers. For DSTree, time savings drop for $k = 5$, but stay constant after, with the exception of the seismic dataset, where time savings stay constant but the ratio of exact answers significantly drops. Overall, the distance-error stopping criterion provides a way for quickly finding low-error answers that may not be exact.

Figure 19c presents results for the time-bound and probability criteria. The probability criterion has a better ratio of exact answers but time savings for both criteria are very similar and drop as k increases. Time savings become as low as 10% for the seismic dataset and $k \geq 50$ (iSAX2+).

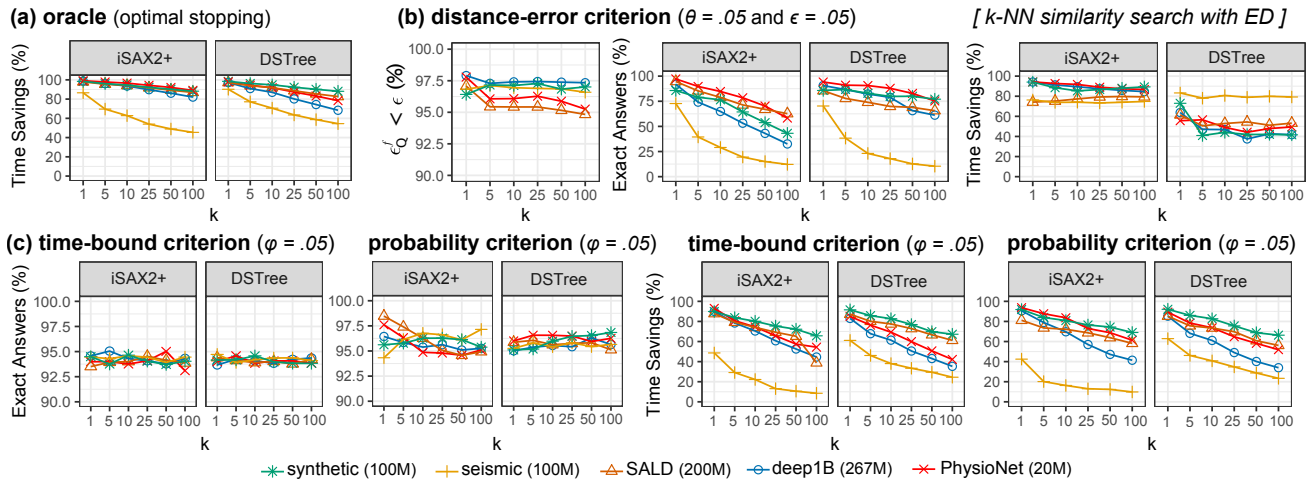


Fig. 19: Evaluation of our stopping criteria for k-NN similarity search ($n_r = 100$ training queries). (a) Optimal time savings if an oracle stopped the search as soon as the exact k-NN was found. (b) Results for the distance-error criterion, where ϵ_Q^f refers to the maximum, i.e., family-wise, distance error among all k nearest neighbors. (c) Results for the time-bound criterion and the probability criterion. For our experiments with ED, we used the large versions of the datasets in Table 2.

7.4 Results for k-NN Similarity Search with DTW

We now repeat the same experiments for DTW, where we apply the bounding envelopes we presented in Section 5.5. In this case, we use the smaller versions of the datasets in Table 2, because similarity search with DTW is extremely expensive for larger datasets. For similar performance reasons, we also study similarity search for $k \leq 25$. We present our results in Figure 20.

Compared to the results for ED (refer to Figure 19), we observe that time savings are now lower for all stopping criteria. This loss of performance is due to the wider lower bound distances that we need to use during query answering. The exact answer is also found much later. This is especially the case for the seismic and deep1B datasets (see Figure 20a). For these two datasets, the time-bound and probability stopping criteria result in marginal savings (around 10% for $k = 1$), which tend to disappear as k increases (see Figure 20c). For the distance-error criterion, which does not require exact answers, time savings are more pronounced. For seismic, synthetic, SALD, and PhysioNet, we observe savings in the range of $30 - 60\%$, independently of k (see Figure 20b). Thus, we conclude that our approach is still valuable when using DTW, but with less impressive results, due to the fact that the use of DTW renders the problem harder.

7.5 Results for k-NN Classification

As mentioned earlier, we use a different set of annotated datasets to evaluate our prediction methods for k-NN classification. Table 4 presents the $\%$ accuracy of exact k-NN classification for the datasets and conditions that we evaluate. It is worth noting that for some datasets, higher accuracy is achieved with a small number of nearest neighbors. Experiments on DTW are conducted with smaller synthetic datasets (up to 20M series)[5]. However, we investigate the role of the dataset size in detail. We note also that the accuracy of k-NN classification for ImageNet is considerably lower than the accuracy ($> 80\%$) of state-of-the-art neural network architectures [124]. DTW is not meaningful for ImageNet image embeddings while it is very expensive, thus we do not include it in our evaluation.

Table 4: Ground truth ($\%$ accuracy) of the exact k-NN classification for the main datasets that we evaluate in our experiments.

	Dataset	1-NN	3-NN	5-NN	10-NN	20-NN
Euclidean	CBF1 (200M)	67.0	70.2	69.7	70.7	70.8
	CBF3 (200M)	91.0	91.2	89.8	90.8	90.8
	SITS	85.0	85.0	84.0	83.9	82.5
	ImageNet	31.0	32.1	33.0	32.4	33.3
	ImageNet (30 cl.)	57.0	58.6	58.3	56.9	56.9
DTW	CBF1 (20M)	66.3	70.7	72.3	74.1	75.4
	CBF3 (20M)	96.7	97.1	97.2	97.5	97.4
	SITS	84.3	83.9	83.3	82.6	81.5

We present an overview of how k-NN similarity search and classification ($k = 10$) evolve in the case of Euclidean distance in Figure 21a. The top graphs show the percentage of queries for which the 10-NN is found, while the the bottom graphs show the percentage of queries for which the current class is the exact. We observe that for CBF3 and SITS, the exact class is generally found very early, e.g., at the very first leaf. ImageNet turns to be significantly harder

[5] When using DTW, k-NN search becomes computationally very expensive, and the time required to run all experiments with the original, large dataset sizes was prohibitive.

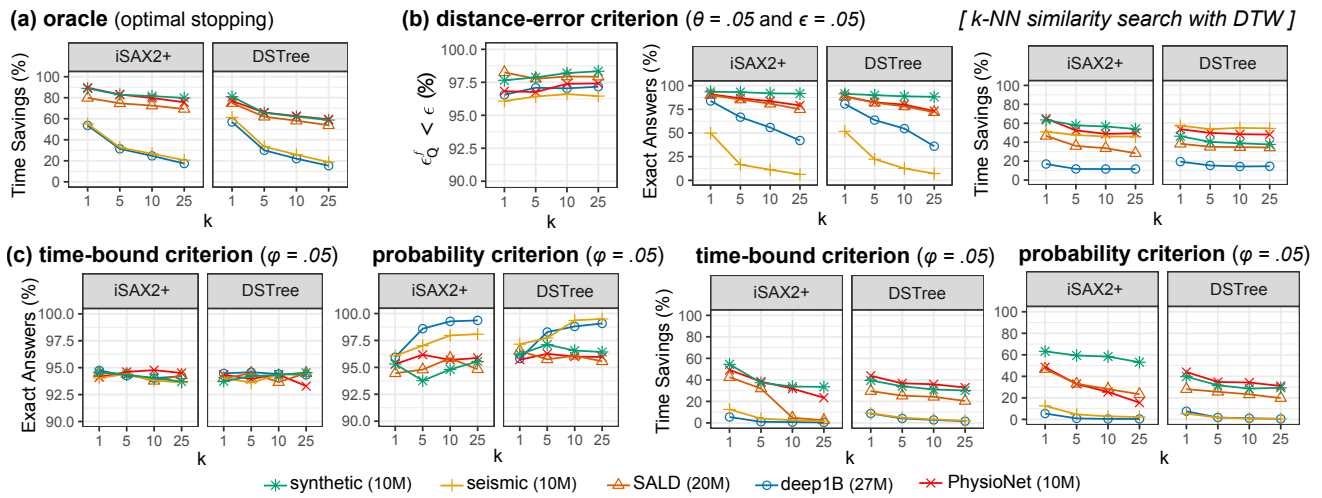


Fig. 20: Evaluation of our stopping criteria for k-NN similarity search and DTW ($n_r = 100$ training queries). (a) Optimal time savings if an oracle stopped the search as soon as the exact k-NN was found. (b) Results for the distance-error criterion, where ϵ_Q^f refers to the maximum, i.e., family-wise, distance error among all k nearest neighbors. (c) Results for the time-bound criterion and the probability criterion. For our experiments with DTW, we used the small versions of the datasets in Table 2.

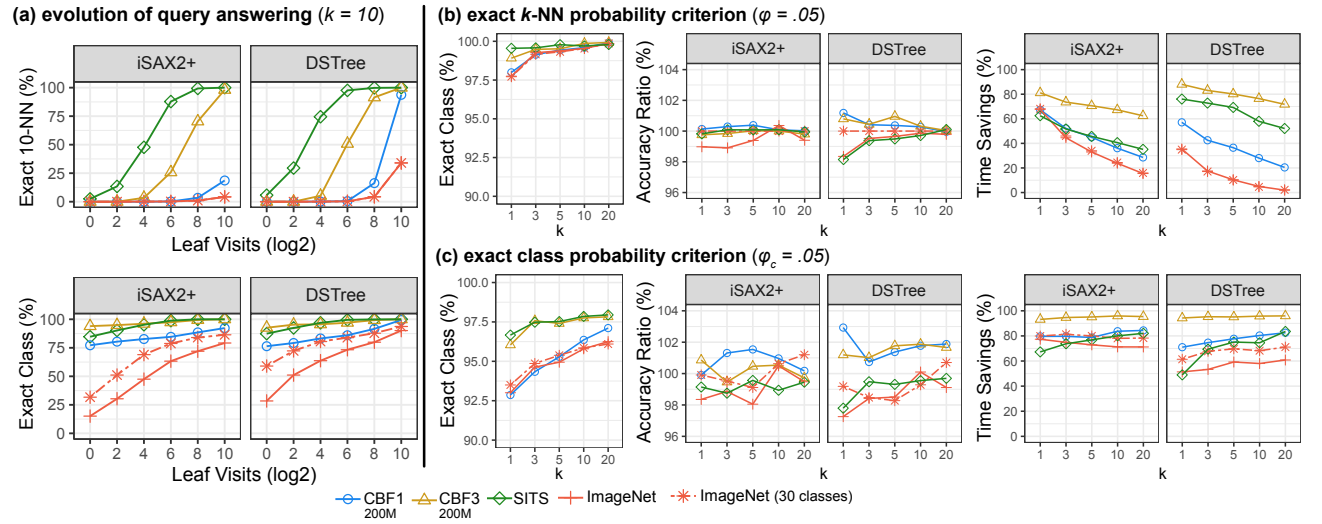


Fig. 21: Results for k-NN classification (Euclidean distance). (a) Evolution of answers to 10-NN classification queries for random sample of 1000 queries per dataset. We show the percentage of queries for which the current 10-NN is exact (top) and the percentage of queries for which the current class is exact (bottom). (b) Results for our conservative stopping criterion that assesses the probability that the current k-NN is exact. (c) Results for our class-level criterion that assesses the probability that the current class is exact. We use $n_r = 100$ training queries.

– exact answers arrive much later during the k-NN search. In summary, there is no guarantee that a quick approximate answer will return the exact class.

Figure 21b presents results for the naive probability criterion that controls for the ratio of the exact k-NNs ($\phi = .05$). As expected, this criterion is extremely conservative: when $k \geq 3$, the ratio of exact answers becomes higher than 99% for all datasets. As a consequence, time savings drop as k increases and become especially low for the harder datasets. Observe that the accuracy ratio is strictly higher than ratio of exact classes and is often higher than 100%. Although this result may seem counter-intuitive, we note that an exact answer (i.e., the answer of the non-progressive k-NN classifier) is not necessarily correct. In this case, it may happen that the non-exact progressive (approximate) answer is the correct one, and this is more likely to occur when the number of alternative classes is small, as it is the case for the two CBF datasets. Still, one might expect that the correct class would coincide more frequently with the final exact answer. Interestingly, this is not always the case, which

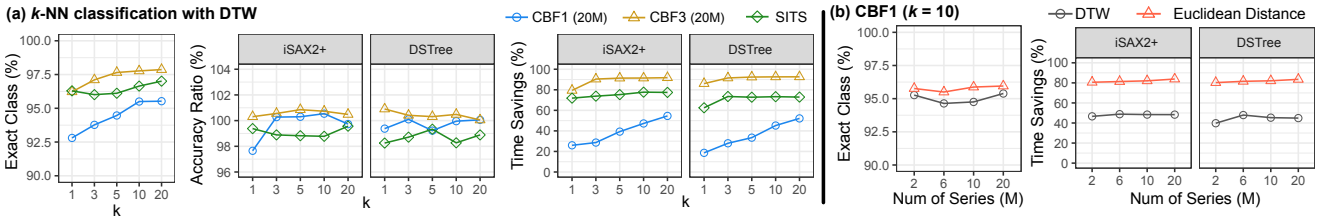


Fig. 22: Evaluation of the exact class probability criterion ($\phi_c = .05$) for k-NN classification: (a) with DTW and (b) with DTW and Euclidean distance when varying the number of data series for CBF1. We use $n_r = 100$ training queries.

suggests that our model predictors can often help to optimally stop, e.g., when consensus among the class of nearest neighbors is high.

Figure 21c presents results for our exact class probability criterion. The average ratio of exact answers is close to or higher than its nominal level of 95% ($\phi_c = .05$) for most cases but deteriorates when $k \leq 3$. The accuracy ratio is again high, ranging between 97% and 103%. Time savings are especially high for iSAX2+, greater than 65% and up to 95% for CBF3. Overall, these results demonstrate that our approach can achieve huge time improvement with no or with minimal cost in terms of classification accuracy.

Finally, we evaluate our exact class probability criterion ($\phi_c = .05$) with DTW. Results are presented in Figure 22a. For CBF3 and SITS, time savings are again at similar levels as with Euclidean distance (see Figure 21c). In contrast, savings are now more modest for CBF1 but they grow as k increases. Of course, the size of CBF1 is smaller now. However, as we see in Figure 22b, the size of the dataset (i.e., the number of series) does not seem to have any clear effect on savings. It also becomes clear that for this harder dataset, our approach leads to relatively larger benefits with Euclidean distance than with DTW.

8 Conclusions

In this work, we argue that two important research questions are how to provide progressive answers for similarity search queries on very large data series collections, and how to couple these answers with probabilistic quality guarantees. Providing progressive answers for data series similarity search queries along with probabilistic quality guarantees is an important research problem. It eliminates wasted time and reduces user waiting times, in cases where improvement in the final answer is not possible.

In this context, we studied the problems of k-NN similarity search and classification for the Euclidean and DTW distance measures. We described our approach, ProS, which comprises the first scalable and effective solutions to these problems, and demonstrated its applicability, effectiveness and significant time savings using several synthetic and real datasets from diverse domains.

As part of our future work, we are going to study in detail how and when such probabilistic measures help humans to effectively complete their visual analysis tasks, as well as which visualization and human-computer interaction approaches are the most suitable in this context. Given the increasing popularity of data series analysis tasks, these research directions are both relevant and important, offering exciting research opportunities.

Acknowledgments. We would like to thank Siddharth Grover for his contributions in the implementation of some of the algorithms in this paper. Work partially supported by program Investir l'Avenir and Univ. of Paris IDEX Emergence en Recherche ANR-18-IDEX-0001, EU project NESTOR (MSCA #748945), and FMJH Program PGMO with EDF-THALES.

References

1. Supplementary material (2022). URL https://helios2.mi.parisdescartes.fr/~themisp/pros/
2. et al., R.K.: quantreg: Quantile regression. https://cran.r-project.org/web/packages/quantreg (2019)
3. Angelini, M., Santucci, G., Schumann, H., Schulz, H.J.: A review and characterization of progressive visual analytics. Informatics **5**, 31 (2018)
4. Ankerst, M., Kastenmüller, G., Kriegel, H.P., Seidl, T.: Nearest neighbor classification in 3d protein databases. ISMB (1999)
5. Arya, S., Mount, D.M., Netanyahu, N.S., Silverman, R., Wu, A.Y.: An optimal algorithm for approximate nearest neighbor searching fixed dimensions. J. ACM **45**(6), 891–923 (1998). DOI 10.1145/293347.293348. URL http://doi.acm.org/10.1145/293347.293348
6. Aßfalg, J., Kriegel, H., Kröger, P., Renz, M.: Probabilistic similarity search for uncertain time series. In: Scientific and Statistical Database Management, 21st International Conference, SSDBM 2009, New Orleans, LA, USA, June 2-4, 2009, Proceedings, pp. 435–443 (2009). DOI 10.1007/978-3-642-02279-1_31. URL https://doi.org/10.1007/978-3-642-02279-1_31
7. Babenko, A., Lempitsky, V.S.: The inverted multi-index. IEEE Trans. Pattern Anal. Mach. Intell. **37**(6), 1247–1260 (2015)
8. Badam, S.K., Elmqvist, N., Fekete, J.D.: Steering the craft: Ui elements and visualizations for supporting progressive visual analytics. Comput. Graph. Forum **36**(3), 491–502 (2017). DOI 10.1111/cgf.13205. URL https://doi.org/10.1111/cgf.13205
9. Bagnall, A.J., Cole, R.L., Palpanas, T., Zoumpatianos, K.: Data series management (dagstuhl seminar 19282). Dagstuhl Reports **9**(7) (2019)

10. Bagnall, A.J., Lines, J., Bostrom, A., Large, J., Keogh, E.J.: The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. Data Min. Knowl. Discov. **31**(3), 606–660 (2017)
11. Bansal, P., Deshpande, P., Sarawagi, S.: Missing value imputation on multidimensional time series. Proc. VLDB Endow. **14**(11), 2533–2545 (2021). DOI 10.14778/3476249.3476300. URL http://www.vldb.org/pvldb/vol14/p2533-bansal.pdf
12. Batista, G.E., Keogh, E.J., Tataw, O.M., Souza, V.M.: Cid: An efficient complexity-invariant distance for time series. Data Min. Knowl. Discov. **28**(3) (2014)
13. Blázquez-García, A., Conde, A., Mori, U., Lozano, J.A.: A review on outlier/anomaly detection in time series data. ACM Comput. Surv. **54**(3), 56:1–56:33 (2021). DOI 10.1145/3444690. URL https://doi.org/10.1145/3444690
14. Boniol, P., Linardi, M., Roncallo, F., Palpanas, T.: Automated Anomaly Detection in Large Sequences. In: ICDE (2020)
15. Boniol, P., Linardi, M., Roncallo, F., Palpanas, T., Meftah, M., Remy, E.: Unsupervised and Scalable Subsequence Anomaly Detectionin Large Data Series. VLDBJ (2021)
16. Boniol, P., Meftah, M., Remy, E., Palpanas, T.: dcam: Dimension-wise class activation map for explaining multivariate data series classification. In: SIGMOD '22: International Conference on Management of Data, Philadelphia, PA, USA, June 12 - 17, 2022, pp. 1175–1189 (2022)
17. Boniol, P., Palpanas, T.: Series2Graph: Graph-based Subsequence Anomaly Detection for Time Series. PVLDB (2020)
18. Boniol, P., Paparrizos, J., Kang, Y., Palpanas, T., Tsay, R., Elmore, A.J., Franklin, M.J.: Theseus: Navigating the Labyrinth of Subsequence Anomaly Detection. Proc. VLDB Endow. (2022)
19. Boniol, P., Paparrizos, J., Palpanas, T., Franklin, M.J.: SAND: Streaming Subsequence Anomaly Detection. PVLDB (2021)
20. Brin, S.: Near neighbor search in large metric spaces. In: Proceedings of the 21th International Conference on Very Large Data Bases, VLDB '95, pp. 574–584. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1995). URL http://dl.acm.org/citation.cfm?id=645921.673006
21. Buono, P., Simeone, A.L.: Interactive shape specification for pattern search in time series. In: AVI (2008)
22. Camerra, A., Palpanas, T., Shieh, J., Keogh, E.J.: isax 2.0: Indexing and mining one billion time series. In: ICDM, pp. 58–67. IEEE Computer Society (2010)
23. Camerra, A., Shieh, J., Palpanas, T., Rakthanmanon, T., Keogh, E.J.: Beyond one billion time series: Indexing and mining very large time series collections with isax2+. Knowl. Inf. Syst. **39**(1), 123–151 (2014)
24. Castelli, V., Li, C., Turek, J., Kontoyiannis, I.: Progressive classification in the compressed domain for large EOS satellite databases. In: 1996 IEEE International Conference on Acoustics, Speech, and Signal Processing Conference Proceedings, ICASSP '96, Atlanta, Georgia, USA, May 7-10, 1996, pp. 2199–2202 (1996)
25. Chakrabarti, K., Keogh, E., Mehrotra, S., Pazzani, M.: Locally adaptive dimensionality reduction for indexing large time series databases. ACM Trans. Database Syst. **27**(2), 188–228 (2002). DOI 10.1145/568518.568520. URL http://doi.acm.org/10.1145/568518.568520
26. Chandola, V., Banerjee, A., Kumar, V.: Anomaly detection: A survey. ACM Computing Surveys (CSUR) **41**(3), 15 (2009)
27. Chatzigeorgakidis, G., Skoutas, D., Patroumpas, K., Palpanas, T., Athanasiou, S., Skiadopoulos, S.: Local similarity search on geolocated time series using hybrid indexing. In: Proceedings of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, SIGSPATIAL 2019, Chicago, IL, USA, November 5-8, 2019, pp. 179–188 (2019)
28. Chatzigeorgakidis, G., Skoutas, D., Patroumpas, K., Palpanas, T., Athanasiou, S., Skiadopoulos, S.: Twin subsequence search in time series. In: Proceedings of the 24th International Conference on Extending Database Technology, EDBT 2021, Nicosia, Cyprus, March 23 - 26, 2021, pp. 475–480 (2021)
29. Chaudhuri, S., Ding, B., Kandula, S.: Approximate query processing: No silver bullet. In: SIGMOD (2017)
30. Chen, Y., Garcia, E.K., Gupta, M.R., Rahimi, A., Cazzanti, L.: Similarity-based classification: Concepts and algorithms. J. Mach. Learn. Res. **10**, 747–776 (2009). URL http://dl.acm.org/citation.cfm?id=1577069.1577096
31. Ciaccia, P., Nanni, A., Patella, M.: A query-sensitive cost model for similarity queries with m-tree. In: In Proc. of the 10th ADC, pp. 65–76. Springer Verlag (1999)
32. Ciaccia, P., Patella, M.: Pac nearest neighbor queries: Approximate and controlled search in high-dimensional and metric spaces. In: ICDE, pp. 244–255 (2000)
33. Ciaccia, P., Patella, M., Zezula, P.: A cost model for similarity queries in metric spaces. In: Proceedings of the Seventeenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, PODS '98, pp. 59–68. ACM, New York, NY, USA (1998). DOI 10.1145/275487.275495. URL http://doi.acm.org/10.1145/275487.275495
34. Correll, M., Gleicher, M.: The semantics of sketch: Flexibility in visual query systems for time series data. In: VAST (2016)
35. Dallachiesa, M., Nushi, B., Mirylenka, K., Palpanas, T.: Uncertain time-series similarity: Return to the basics. PVLDB **5**(11), 1662–1673 (2012). URL http://vldb.org/pvldb/vol5/p1662_micheledallachiesa_vldb2012.pdf
36. Dallachiesa, M., Palpanas, T., Ilyas, I.F.: Top-k nearest neighbor search in uncertain data series. Proc. VLDB Endow. **8**(1), 13–24 (2014). DOI 10.14778/2735461.2735463. URL http://dx.doi.org/10.14778/2735461.2735463
37. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: ImageNet: A Large-Scale Hierarchical Image Database. In: CVPR09 (2009)
38. Ding, B., Huang, S., Chaudhuri, S., Chakrabarti, K., Wang, C.: Sample + seek: Approximating aggregates with distribution precision guarantee. In: SIGMOD (2016)
39. Ding, H., Trajcevski, G., Scheuermann, P., Wang, X., Keogh, E.: Querying and mining of time series data: Experimental comparison of representations and distance measures. Proceedings of the VLDB Endowment **1**(2), 1542–1552 (2008)
40. Douze, M., Tolias, G., Pizzi, E., Papakipos, Z., Chanussot, L., Radenovic, F., Jeníček, T., Maximov, M., Leal-Taixé, L., Elezi, I., Chum, O., Canton-Ferrer, C.: The 2021 image similarity dataset and challenge. CoRR **abs/2106.09672** (2021)
41. Duong, T., Hazelton, M.L.: Cross-validation bandwidth matrices for multivariate kernel density estimation. Scandinavian Journal of Statistics **32**(3), 485–506 (2005). DOI 10.1111/j.1467-9469.2005.00445.x
42. Duong, T., Wand, M., Chacon, J., Gramacki, A.: ks: Kernel smoothing. https://cran.r-project.org/web/packages/ks/ (2019)
43. Echihabi, K.: Truly Scalable Data Series Similarity Search. In: VLDB PhD Workshop (2019)
44. Echihabi, K., Fatourou, P., Zoumpatianos, K., Palpanas, T., Benbrahim, H.: Hercules Against Data Series Similarity Search. PVLDB **15**(10) (2022)
45. Echihabi, K., Palpanas, T., Zoumpatianos, K.: New trends in high-d vector similarity search: Ai-driven, progressive, and distributed. Proc. VLDB Endow. **14**(12), 3198–3201 (2021)
46. Echihabi, K., Zoumpatianos, K., Palpanas, T.: Big sequence management: Scaling up and out. In: Y. Velegrakis, D. Zeinalipour-Yazti, P.K. Chrysanthis, F. Guerra (eds.) Proceedings of the 24th International Conference on Extending Database Technology, EDBT 2021, Nicosia, Cyprus, March 23 - 26, 2021, pp.

714–717. OpenProceedings.org (2021). DOI 10.5441/002/edbt.2021.91. URL https://doi.org/10.5441/002/edbt.2021.91

47. Echihabi, K., Zoumpatianos, K., Palpanas, T., Benbrahim, H.: The lernaean hydra of data series similarity search: An experimental evaluation of the state of the art. PVLDB **12**(2), 112–127 (2018)
48. Echihabi, K., Zoumpatianos, K., Palpanas, T., Benbrahim, H.: Return of the Lernaean Hydra: Experimental Evaluation of Data Series Approximate Similarity Search. PVLDB **13**(3), 402–419 (2019)
49. Faloutsos, C., Ranganathan, M., Manolopoulos, Y.: Fast subsequence matching in time-series databases. In: SIGMOD, pp. 419–429. ACM, New York, NY, USA (1994). DOI 10.1145/191839.191925. URL http://doi.acm.org/10.1145/191839.191925
50. Fekete, J.D., Primet, R.: Progressive analytics: A computation paradigm for exploratory data analysis. CoRR **abs/1607.05162** (2016). URL http://arxiv.org/abs/1607.05162
51. Ferhatosmanoglu, H., Tuncel, E., Agrawal, D., Abbadi, A.E.: High dimensional nearest neighbor searching. Inf. Syst. **31**(6), 512–540 (2006)
52. Fisher, D., Drucker, S.M., König, A.C.: Exploratory visualization involving incremental, approximate database queries and uncertainty. IEEE CG&A **32** (2012)
53. Gao, Y., Lin, J.: HIME: discovering variable-length motifs in large-scale time series. Knowl. Inf. Syst. **61**(1), 513–542 (2019)
54. Gao, Y., Lin, J., Brif, C.: Ensemble grammar induction for detecting anomalies in time series. In: Proceedings of the 23rd International Conference on Extending Database Technology, EDBT, pp. 85–96 (2020)
55. Gogolou, A., Tsandilas, T., Echihabi, K., Bezerianos, A., Palpanas, T.: Data series progressive similarity search with probabilistic quality guarantees. In: Proceedings of the 2020 International Conference on Management of Data, SIGMOD (2020)
56. Gogolou, A., Tsandilas, T., Palpanas, T., Bezerianos, A.: Comparing similarity perception in time series visualizations. IEEE TVCG **25** (2018)
57. Gogolou, A., Tsandilas, T., Palpanas, T., Bezerianos, A.: Progressive similarity search on time series data. In: Proceedings of the Workshops of the EDBT/ICDT 2019 Joint Conference, EDBT/ICDT 2019, Lisbon, Portugal, March 26, 2019 (2019). URL http://ceur-ws.org/Vol-2322/BigVis_5.pdf
58. Goldberger, A.L., Amaral, L.A.N., Glass, L., Hausdorff, J.M., Ivanov, P.C., Mark, R.G., Mietus, J.E., Moody, G.B., Peng, C.K., Stanley, H.E.: PhysioBank, PhysioToolkit, and PhysioNet: Components of a new research resource for complex physiologic signals. Circulation **101**(23), e215–e220 (2000 (June 13)). Circulation Electronic Pages: http://circ.ahajournals.org/content/101/23/e215.full PMID:1085218; doi: 10.1161/01.CIR.101.23.e215
59. Goldin, D.Q., Kanellakis, P.C.: On similarity queries for time-series data: Constraint specification and implementation. In: CP (1995)
60. Guo, Y., Binnig, C., Kraska, T.: What you see is not what you get!: Detecting simpson's paradoxes during data exploration. In: Proceedings of the 2nd Workshop on Human-In-the-Loop Data Analytics, HILDA@SIGMOD (2017)
61. Hellerstein, J.M., Haas, P.J., Wang, H.J.: Online aggregation. In: SIGMOD (1997)
62. Hellerstein, J.M., Koutsoupias, E., Papadimitriou, C.H.: On the analysis of indexing schemes. In: Proceedings of the Sixteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, PODS '97, p. 249–256. Association for Computing Machinery, New York, NY, USA (1997). DOI 10.1145/263661.263688. URL https://doi.org/10.1145/263661.263688
63. Huang, T., Zhen, Z., Liu, J.: Semantic relatedness emerges in deep convolutional neural networks designed for object recognition. bioRxiv (2020). DOI 10.1101/2020.07.04.188169. URL https://www.biorxiv.org/content/early/2020/07/06/2020.07.04.188169.1
64. Jégou, H., Douze, M., Schmid, C.: Product quantization for nearest neighbor search. IEEE Trans. Pattern Anal. Mach. Intell. **33**(1), 117–128 (2011)
65. Jermaine, C., Arumugam, S., Pol, A., Dobra, A.: Scalable approximate query processing with the DBO engine. ACM Trans. Database Syst. **33**(4), 23:1–23:54 (2008)
66. Jing, J., Dauwels, J., Rakthanmanon, T., Keogh, E., Cash, S., Westover, M.: Rapid annotation of interictal epileptiform discharges via template matching under dynamic time warping. Journal of Neuroscience Methods **274** (2016)
67. Kanellakis, P.C., Ramaswamy, S., Vengroff, D.E., Vitter, J.S.: Indexing for data models with constraints and classes (extended abstract). In: Proceedings of the Twelfth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, PODS '93, p. 233–243. Association for Computing Machinery, New York, NY, USA (1993). DOI 10.1145/153850.153884. URL https://doi.org/10.1145/153850.153884
68. Keogh, E., Chakrabarti, K., Pazzani, M., Mehrotra, S.: Dimensionality reduction for fast similarity search in large time series databases. Knowledge and Information Systems **3**(3), 263–286 (2001). DOI 10.1007/PL00011669
69. Keogh, E., Pazzani, M.: An enhanced representation of time series which allows fast and accurate classification, clustering and relevance feedback. In: Fourth International Conference on Knowledge Discovery and Data Mining (KDD'98), pp. 239–241. ACM Press, New York City, NY (1998)
70. Keogh, E., Ratanamahatana, C.A.: Exact indexing of dynamic time warping. Knowledge and information systems (2005)
71. Koenker, R.: Quantile Regression. Econometric Society Monographs. Cambridge University Press (2005). DOI 10.1017/CBO9780511754098
72. Kondylakis, H., Dayan, N., Zoumpatianos, K., Palpanas, T.: Coconut: A scalable bottom-up approach for building data series indexes. PVLDB **11**(6), 677–690 (2018). DOI 10.14778/3184470.3184472
73. Kondylakis, H., Dayan, N., Zoumpatianos, K., Palpanas, T.: Coconut: sortable summarizations for scalable indexes over static and streaming data series. VLDB J. **28**(6), 847–869 (2019)
74. Kraska, T.: Northstar: An interactive data science system. PVLDB **11**(12), 2150–2164 (2018)
75. Kwon, O.W., Lee, J.H.: Web page classification based on k-nearest neighbor approach. In: Proceedings of the Fifth International Workshop on on Information Retrieval with Asian Languages (2000)
76. Laviron, P., Dai, X., Huquet, B., Palpanas, T.: Electricity demand activation extraction: From known to uknown signatures, using similarity search. In: Proceedings of the ACM International Conference on Future Energy Systems, e-Energy (2021)
77. Lemire, D.: Faster retrieval with a two-pass dynamic-time-warping lower bound. Pattern Recognit. **42**(9), 2169–2180 (2009)
78. Levchenko, O., Kolev, B., Yagoubi, D.E., Akbarinia, R., Masseglia, F., Palpanas, T., Shasha, D.E., Valduriez, P.: Best-neighbor: efficient evaluation of knn queries on large time series databases. Knowl. Inf. Syst. **63**(2), 349–378 (2021). DOI 10.1007/s10115-020-01518-4. URL https://doi.org/10.1007/s10115-020-01518-4
79. Li, C., Zhang, M., Andersen, D.G., He, Y.: Improving Approximate Nearest Neighbor Search through Learned Adaptive Early Termination. In: SIGMOD (2020)

80. Li, X., Lin, J., Zhao, L.: Time series clustering in linear time complexity. Data Min. Knowl. Discov. **35**(6), 2369–2388 (2021)
81. Lin, J., Keogh, E.J., Lonardi, S., Chiu, B.Y.: A symbolic representation of time series, with implications for streaming algorithms. In: Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery, DMKD 2003, San Diego, California, USA, June 13, 2003, pp. 2–11 (2003). DOI 10.1145/882082.882086
82. Linardi, M., Palpanas, T.: Scalable, variable-length similarity search in data series: The ulisse approach. PVLDB (2019)
83. Linardi, M., Palpanas, T.: Scalable data series subsequence matching with ulisse. VLDBJ (2020)
84. Linardi, M., Zhu, Y., Palpanas, T., Keogh, E.J.: Matrix profile X: Valmod - scalable discovery of variable-length motifs in data series (2018)
85. Linardi, M., Zhu, Y., Palpanas, T., Keogh, E.J.: Matrix profile goes MAD: variable-length motif and discord discovery in data series. Data Min. Knowl. Discov. **34**(4) (2020)
86. Lu, Y., Wu, R., Mueen, A., Zuluaga, M.A., Keogh, E.J.: Matrix profile XXIV: scaling time series anomaly detection to trillions of datapoints and ultra-fast arriving data streams. In: KDD '22: The 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 14 - 18, 2022, pp. 1173–1182 (2022)
87. Lucas, B., Shifaz, A., Pelletier, C., O'Neill, L., Zaidi, N.A., Goethals, B., Petitjean, F., Webb, G.I.: Proximity forest: an effective and scalable distance-based classifier for time series. Data Min. Knowl. Discov. **33**(3), 607–635 (2019)
88. Malkov, Y.A., Yashunin, D.A.: Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. IEEE Trans. Pattern Anal. Mach. Intell. **42**(4), 824–836 (2020)
89. Mannino, M., Abouzied, A.: Expressive time series querying with hand-drawn scale-free sketches. In: CHI (2018)
90. Micallef, L., Schulz, H.J., Angelini, M., Aupetit, M., Chang, R., Kohlhammer, J., Perer, A., Santucci, G.: The human user in progressive visual analytics. In: Short Paper Proceedings of EuroVis'19, pp. 19–23. Eurographics Association (2019). DOI 10.2312/evs.20191164
91. Miller, G.A.: Wordnet: A lexical database for english. Communications of the ACM **38**(11), 39–41 (1995). DOI 10.1145/219717.219748. URL https://doi.org/10.1145/219717.219748
92. Mirylenka, K., Dallachiesa, M., Palpanas, T.: Data series similarity using correlation-aware measures. In: SSDBM (2017)
93. Moritz, D., Fisher, D., Ding, B., Wang, C.: Trust, but verify: Optimistic visualizations of approximate queries for exploring big data. In: CHI (2017)
94. Moritz, D., Howe, B., Heer, J.: Falcon: Balancing interactive latency and resolution sensitivity for scalable linked visualizations. In: Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems, CHI '19, pp. 694:1–694:11. ACM, New York, NY, USA (2019). DOI 10.1145/3290605.3300924. URL http://doi.acm.org/10.1145/3290605.3300924
95. Nielsen, J.: Response times: The 3 important limits. https://www.nngroup.com/articles/response-times-3-important-limits/
96. Palpanas, T.: Data series management: The road to big sequence analytics. SIGMOD Record **44**(2), 47–52 (2015). DOI 10.1145/2814710.2814719. URL http://doi.acm.org/10.1145/2814710.2814719
97. Palpanas, T.: Evolution of a Data Series Index - The iSAX Family of Data Series Indexes. Communications in Computer and Information Science (CCIS) (2020)
98. Palpanas, T., Beckmann, V.: Report on the First and Second Interdisciplinary Time Series Analysis Workshop (ITISA). SIGMOD Rec. **48**(3) (2019)
99. Paparrizos, J., Boniol, P., Palpanas, T., Tsay, R.S., Elmore, A., Franklin, M.J.: Volume Under the Surface: A New Accuracy Evaluation Measure for Time-Series Anomaly Detection. PVLDB (2022)
100. Paparrizos, J., Gravano, L.: Fast and accurate time-series clustering. ACM Trans. Database Syst. **42**(2), 8:1–8:49 (2017)
101. Paparrizos, J., Kang, Y., Boniol, P., Tsay, R., Palpanas, T., Franklin, M.J.: TSB-UAD: an end-to-end benchmark suite for univariate time-series anomaly detection. Proc. VLDB Endow. **15**(8), 1697–1711 (2022). URL https://www.vldb.org/pvldb/vol15/p1697-paparrizos.pdf
102. Paparrizos, J., Liu, C., Elmore, A.J., Franklin, M.J.: Debunking four long-standing misconceptions of time-series distance measures. In: Proceedings of the 2020 International Conference on Management of Data, SIGMOD Conference 2020, online conference [Portland, OR, USA], June 14-19, 2020, pp. 1887–1905. ACM (2020). DOI 10.1145/3318464.3389760. URL https://doi.org/10.1145/3318464.3389760
103. Pelletier, C., Webb, G.I., Petitjean, F.: Temporal convolutional neural network for the classification of satellite image time series. Remote Sensing **11**(5) (2019). DOI 10.3390/rs11050523. URL https://www.mdpi.com/2072-4292/11/5/523
104. Peng, B., Fatourou, P., Palpanas, T.: MESSI: In-Memory Data Series Indexing. In: ICDE (2020)
105. Peng, B., Fatourou, P., Palpanas, T.: Fast data series indexing for in-memory data. VLDBJ (2021)
106. Peng, B., Fatourou, P., Palpanas, T.: SING: Sequence Indexing Using GPUs. In: ICDE (2021)
107. Peng, B., Palpanas, T., Fatourou, P.: Paris: The next destination for fast data series indexing and query answering. IEEE BigData (2018)
108. Peng, B., Palpanas, T., Fatourou, P.: Paris+: Data series indexing on multi-core architectures. TKDE (2020)
109. Petitjean, F., Forestier, G., Webb, G.I., Nicholson, A.E., Chen, Y., Keogh, E.J.: Dynamic time warping averaging of time series allows faster and more accurate classification. In: ICDM (2014)
110. Phillips, N.: A companion to the e-book "yarrr!: The pirate's guide to r". https://github.com/ndphillips/yarrr (2017)
111. Rahman, S., Aliakbarpour, M., Kong, H.K., Blais, E., Karahalios, K., Parameswaran, A., Rubinfield, R.: I've seen "enough": Incrementally improving visualizations to support rapid decision making. Proc. VLDB Endow. **10**(11), 1262–1273 (2017). DOI 10.14778/3137628.3137637. URL https://doi.org/10.14778/3137628.3137637
112. Rakthanmanon, T., Campana, B.J.L., Mueen, A., Batista, G.E.A.P.A., Westover, M.B., Zhu, Q., Zakaria, J., Keogh, E.J.: Searching and mining trillions of time series subsequences under dynamic time warping. In: KDD, pp. 262–270. ACM (2012)
113. Rakthanmanon, T., Campana, B.J.L., Mueen, A., Batista, G.E.A.P.A., Westover, M.B., Zhu, Q., Zakaria, J., Keogh, E.J.: Searching and mining trillions of time series subsequences under dynamic time warping. In: The 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD, pp. 262–270. ACM (2012)
114. Rakthanmanon, T., Keogh, E.J., Lonardi, S., Evans, S.: Time series epenthesis: Clustering time series streams requires ignoring some data. In: Data Mining (ICDM), 2011 IEEE 11th International Conference on, pp. 547–556. IEEE (2011)
115. Rodrigues, P.P., Gama, J., Pedroso, J.P.: Odac: Hierarchical clustering of time series data streams. In: SDM, pp. 499–503. SIAM (2006)
116. Saito, N.: Local Feature Extraction and its Applications using a Library of Bases, pp. 269–451 (2000). DOI 10.1142/9789812813305_0005. URL https://www.worldscientific.com/doi/abs/10.1142/9789812813305_0005

117. Sakoe, H., Chiba, S.: Dynamic Programming Algorithm Optimization for Spoken Word Recognition, p. 159–165. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1990)
118. Sarangi, S.R., Murthy, K.: Dust: A generalized notion of similarity between uncertain time series. In: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, July 25-28, 2010, pp. 383–392 (2010). DOI 10.1145/1835804.1835854. URL http://doi.acm.org/10.1145/1835804.1835854
119. Schäfer, P., Leser, U.: TEASER: early and accurate time series classification. Data Min. Knowl. Discov. **34**(5), 1336–1362 (2020)
120. Schneider, J., Wenig, P., Papenbrock, T.: Distributed detection of sequential anomalies in univariate time series. VLDBJ **30** (2021)
121. Schulz, H.J., Angelini, M., Santucci, G., Schumann, H.: An enhanced visualization process model for incremental visualization. IEEE Transactions on Visualization and Computer Graphics **22**, 1830–1842 (2016). DOI 10.1109/TVCG.2015.2462356
122. for Seismology, I.R.I.: Iris seismic data access (2014). http://ds.iris.edu/data/access/
123. Stolper, C.D., Perer, A., Gotz, D.: Progressive visual analytics: User-driven visual exploration of in-progress analytics. IEEE TVCG **20** (2014)
124. Tan, M., Le, Q.: EfficientNet: Rethinking model scaling for convolutional neural networks. In: K. Chaudhuri, R. Salakhutdinov (eds.) Proceedings of the 36th International Conference on Machine Learning, *Proceedings of Machine Learning Research*, vol. 97, pp. 6105–6114. PMLR (2019). URL http://proceedings.mlr.press/v97/tan19a.html
125. Tufte, E.R.: The Visual Display of Quantitative Information (1986)
126. Turkay, C., Kaya, E., Balcisoy, S., Hauser, H.: Designing progressive and interactive analytics processes for high-dimensional data analysis. IEEE Transactions on Visualization and Computer Graphics **23**(1), 131–140 (2017). DOI 10.1109/TVCG.2016.2598470. URL https://doi.org/10.1109/TVCG.2016.2598470
127. University, S.: Southwest university adult lifespan dataset (sald) (2017). http://fcon_1000.projects.nitrc.org/indi/retro/sald.html
128. Vision, S.C.: Deep billion-scale indexing. http://sites.skoltech.ru/compvision/noimi (2018)
129. Wald, A.: Sequential tests of statistical hypotheses. The Annals of Mathematical Statistics **16**(2), 117–186 (1945). DOI 10.1214/aoms/1177731118. URL https://doi.org/10.1214/aoms/1177731118
130. Wand, M.P., Jones, M.C.: Comparison of smoothing parameterizations in bivariate kernel density estimation. Journal of the American Statistical Association **88**(422), 520–528 (1993). DOI 10.1080/01621459.1993.10476303
131. Wand, M.P., Jones, M.C.: Multivariate plug-in bandwidth selection. Computational Statistics **9**(2), 97–116 (1994). URL http://oro.open.ac.uk/28244/
132. Wang, Q., Palpanas, T.: Deep Learning Embeddings for Data Series Similarity Search. In: SIGKDD (2021)
133. Wang, Q., Whitmarsh, S., Navarro, V., Palpanas, T.: iEDeaL: A Deep Learning Framework for Detecting Highly Imbalanced Interictal Epileptiform Discharges. PVLDB **16**(2) (2023)
134. Wang, Y., Wang, P., Pei, J., Wang, W., Huang, S.: A data-adaptive and dynamic segmentation index for whole matching on time series. PVLDB **6**(10), 793–804 (2013)
135. Warren Liao, T.: Clustering of time series data — a survey. Pattern Recognition **38**(11), 1857–1874 (2005)
136. Wellenzohn, K., Böhlen, M.H., Dignös, A., Gamper, J., Mitterer, H.: Continuous imputation of missing values in streams of pattern-determining time series. In: Proceedings of the 20th International Conference on Extending Database Technology, EDBT, pp. 330–341. OpenProceedings.org (2017)
137. Wu, S., Ooi, B.C., Tan, K.: Online aggregation. In: Advanced Query Processing, Volume 1: Issues and Trends, pp. 187–210 (2013)
138. Yagoubi, D.E., Akbarinia, R., Masseglia, F., Palpanas, T.: Dpisax: Massively distributed partitioned isax (2017)
139. Yagoubi, D.E., Akbarinia, R., Masseglia, F., Palpanas, T.: Massively distributed time series indexing and querying. TKDE **32**(1) (2020)
140. Yankov, D., Keogh, E.J., Rebbapragada, U.: Disk aware discord discovery: finding unusual time series in terabyte sized datasets. Knowl. Inf. Syst. **17**(2), 241–262 (2008)
141. Yeh, C.C.M., Zhu, Y., Ulanova, L., Begum, N., Ding, Y., Dau, H.A., Zimmerman, Z., Silva, D.F., Mueen, A., Keogh, E.: Time series joins, motifs, discords and shapelets: A unifying view that exploits the matrix profile. Data Mining and Knowledge Discovery pp. 1–41 (2017)
142. Yeh, M., Wu, K., Yu, P.S., Chen, M.: Proud: A probabilistic approach to processing similarity queries over uncertain data streams. In: EDBT 2009, 12th International Conference on Extending Database Technology, Saint Petersburg, Russia, March 24-26, 2009, Proceedings, pp. 684–695 (2009). DOI 10.1145/1516360.1516439. URL http://doi.acm.org/10.1145/1516360.1516439
143. Zgraggen, E., Galakatos, A., Crotty, A., Fekete, J., Kraska, T.: How progressive visualizations affect exploratory analysis. IEEE Transactions on Visualization and Computer Graphics **23**(8), 1977–1987 (2017). DOI 10.1109/TVCG.2016.2607714
144. Zgraggen, E., Zhao, Z., Zeleznik, R.C., Kraska, T.: Investigating the effect of the multiple comparisons problem in visual analysis. In: CHI (2018)
145. Zoumpatianos, K., Idreos, S., Palpanas, T.: Rinse: Interactive data series exploration with ads+. PVLDB **8**(12), 1912–1915 (2015). DOI 10.14778/2824032.2824099
146. Zoumpatianos, K., Idreos, S., Palpanas, T.: Ads: The adaptive data series index. VLDB J. **25**(6), 843–866 (2016). DOI 10.1007/s00778-016-0442-5
147. Zoumpatianos, K., Lou, Y., Palpanas, T., Gehrke, J.: Query workloads for data series indexes. In: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Sydney, NSW, Australia, August 10-13, 2015, pp. 1603–1612 (2015). DOI 10.1145/2783258.2783382. URL http://doi.acm.org/10.1145/2783258.2783382