Fast and Exact Similarity Search in less than a Blink of an Eye

Patrick Schäfer*, Jakob Brand†, Ulf Leser‡
Humboldt-Universität zu Berlin
Berlin, Germany
Email: *patrick.schaefer@hu-berlin.de,
†brandjak@hu-berlin.de,
‡leser@informatik.hu-berlin.de

Botao Peng
Institute of Computing Technology
Chinese Academy of Sciences
Beijing, China
pengbotao@ict.ac.cn

Themis Palpanas
LIPADE
Université de Paris
Paris, France
themis@mi.parisdescartes.fr

***Abstract*—Similarity search is a fundamental operation for analyzing data series (DS), which are ordered sequences of real values. To enhance efficiency, summarization techniques are employed that reduce the dimensionality of DS. SAX-based approaches are the state-of-the-art for exact similarity queries, but their performance degrades for high-frequency signals, such as noisy data, or for high-frequency DS. In this work, we present the SymbOlic Fourier Approximation index (SOFA), which implements fast, exact similarity queries. SOFA is based on two building blocks: a tree index (inspired by MESSI) and the SFA symbolic summarization. It makes use of a learned summarization method called Symbolic Fourier Approximation (SFA), which is based on the Fourier transform and utilizes a data-adaptive quantization of the frequency domain. To better capture relevant information in high-frequency signals, SFA selects the Fourier coefficients by highest variance, resulting in a larger value range, thus larger quantization bins. The tree index solution employed by SOFA makes use of the GEMINI-approach to answer exact similarity search queries using lower bounding distance measures, and an efficient SIMD implementation. We further propose a novel benchmark comprising 17 diverse datasets, encompassing 1 billion DS. Our experimental results demonstrate that SOFA outperforms existing methods on exact similarity queries: it is up to 10 times faster than a parallel sequential scan, 3-4 times faster than FAISS, and 2 times faster on average than MESSI. For high-frequency datasets, we observe a remarkable 38-fold performance improvement.**

***Index Terms*—Data Series, Time Series, Similarity Search, Exact, Euclidean Distance.**

I. Introduction

The advancements and deployments of modern sensors have led to the generation, collection, and analysis of massive datasets of data series (DS) in nearly every scientific field [1], [2]. A DS is an ordered sequence of real values. The most common type of DS is the time series, which is ordered in time. Other frequent orders are wave lengths, angles, pixel offsets, or locations [3].

Important analysis problems for DS are querying [4], classification [5], clustering [6], anomaly detection [7], which all require special algorithms that take the sequential nature of the values into account. At the heart of these techniques is similarity search [4], [8]–[10], which aims to identify the series in a dataset that is closest to a given query series based on a distance measure, such as the widely used Euclidean distance (ED). Similarity search can be split into two main categories: exact search and approximate search [4]. In this work, we focus on exact similarity search using ED.

Indexing is a widely used technique to accelerate similarity searches. Most indices for DS similarity search utilize summarized representations of the DS to map them into lower-dimensional spaces. Symbolic Aggregate approXimation (SAX) [11] is among the most popular techniques [8]. SAX works by computing mean values over segments and then quantizing these values to form a SAX word. Its quantization employs a fixed alphabet of symbols, and the Normal Distribution is divided into equal-depth bins to derive these symbols. Consequently, SAX assumes that the series of a dataset are Normally distributed. The distance between two SAX words from DS A and B provides a lower bound to the ED between A and B, allowing for exact indexing using the GEMINI framework [12], [13], which will be explained in Section II-A. The tighter the lower bound to the ED, the better the indexing performance, leading to reduced retrieval times. Although SAX's simplicity is sufficient for many scenarios, its rigid quantization scheme results in decreased performance when dealing with high-frequency data.

Figure 1 illustrates some cases where the SAX transformation is failing on our benchmark datasets (Section V). The original signal is illustrated in gray, and the mean values over 16 segments in orange. For high-frequency signals, using mean values (orange) fails to capture the signal's semantics, resulting in a flat line summarization (Figure 1 TOP). Additionally, the actual data distribution is often non-Gaussian, leading to poor lower bounding distances (LBDs) and degraded indexing performance (Figure 1 BOTTOM). Please note that Figure 1 (top) showcases a single high-frequency series from each dataset where SAX fails, providing a simplified perspective, as each dataset comprises multiple series with varying frequencies. Overall, our datasets cover a wide range of frequencies (Figure 14), with OBS and ISC-EHB having lowest frequencies.

We introduce the SOFA *(SymbOlic Fourier Approximation)* index, built on two core components: the Symbolic Fourier Approximation (SFA) [14] and a tree index based on MESSI, adapted to our approach. SOFA leverages SFA for efficient and accurate similarity search in large datasets, addressing

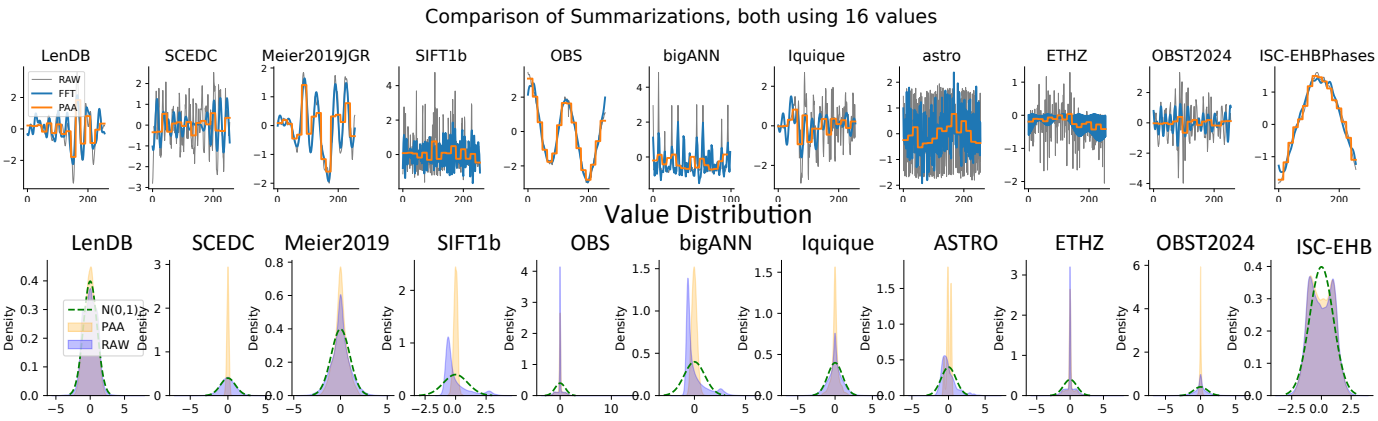

Fig. 1. TOP: PAA with 16 values (orange) fails to approximate a data series (in gray) with high frequency, resulting in a flat line. Meanwhile FFT with 16 values (in blue) closely mimics the data. BOTTOM: Distribution of values for each dataset. SAX is built upon the assumption that the data follows Normal N(0,1) distribution (dotted in green). This is neither the case case for the PAA approximations nor the raw data.

challenges associated with high-frequency data. SFA transforms a DS into the frequency domain and applies a learned quantization of the Fourier coefficients. This quantization is derived from the frequency distribution of all series within a dataset, ensuring that the representation adapts to the dataset's characteristics, rather than a single series. Figure 1 highlights SFA's ability to approximate high-frequency series. The SFA approximation using the 8 dominant Fourier coefficients (8 real & 8 imag values), learned from the dataset, is depicted in blue. As evident from the figure, SFA provides a closer approximation to the actual DS compared to SAX, especially for signals with high variability.

The MESSI index [15], [16] represents the state-of-the-art in-memory DS index. It makes use of the GEMINI-approach to answer exact similarity search queries. It was specifically developed for concurrent multi-threaded environments, modern hardware (SIMD), and for in-memory DS management. Our contributions can be summarized as follows:

1) We propose utilizing the learned symbolic summarization technique, Symbolic Fourier Approximation (SFA), for efficient DS similarity search. SFA demonstrates superior capability in capturing the latent semantics of high-frequency DS (Section V-C).
2) We introduce SOFA, an advanced indexing structure that synergistically combines SFA with a parallel tree index adapted from MESSI, a state-of-the-art DS index optimized for modern hardware architectures (Section IV).
3) In Section IV-G, we present an efficient SIMD-based implementation of the SFA lower-bounding distance, which solves conditional branching and lower bounding.
4) We present the most extensive benchmark to date for similarity search, encompassing 17 diverse datasets. It totals $\sim$ 1 TB of data, comprising 1,017,586,504 (1 billion) DS (Table I).
5) Our extensive experimental evaluation (Section V) demonstrates that SOFA consistently outperforms existing methods. It is up to 38 times faster than MESSI [15], and on average 2.5 times faster. It is up to 10 times faster than the a parallel sequential scan [17], and $3-4$ times faster than FAISS [18].

The remainder of this paper is structures as follows: In Section II we review the background on similarity search. Section III discussed the related work. Section IV introduces the SOFA framework based on the MESSI index IV-A, and the SFA summarization technique IV-C. Section V presents our experimental results and Section VI concludes.

II. Background and Definitions

We define DS and the z-normalized Euclidean distance (z-ED), which we will use throughout this work.

Definition 1: **Data Series (DS)**: A data series $A = (p_1, \ldots, p_n)$ is an ordered sequence of n points, where each point $p_i = (t_i, a_i)$ consists of a real value $a_i \in \mathbb{R}$ and a position $t_i \in \mathbb{N}$.

In the subsequent discussion, we will disregard the positions and focus solely on the values. The order of values differentiate DS from vector data, such as embeddings learned from images or text. DS typically show low frequency, which allows for efficient summarization techniques to be applied. The later type of data is characterized by high variance in high frequency, which makes summarization more challenging. In the context of similarity search, the similarity of two subsequences is measured using the *z-normalized ED*.

Definition 2: **z-normalized Euclidean distance (z-ED)**: Given two univariate DS $A = (a_1, \ldots, a_l)$ with mean μ_A and standard-deviation σ_A and $B = (b_1, \ldots, b_l)$ with μ_B and σ_B, both of length l, their (squared) z-ED is defined as:

$$d(A,B) = \sqrt{\sum_{t=1}^{l} \left(\frac{a_t - \mu_A}{\sigma_A} - \frac{b_t - \mu_B}{\sigma_B} \right)^2}$$

Given a query series S_q of length n, a collection $\mathcal{S}$ of size N consisting of DS of length n, similarity search aims to identify the nearest neighbor series $S_c \in \mathcal{S}$ for which the distance to S_q is the smallest: $\forall S_o \in \mathcal{S}, S_o \neq S_c : d(S_c, S_q) \leq d(S_o, S_c)$

Similarity search can be accelerated using summarization techniques, which map DS into lower-dimensional space.

Definition 3: A **summarization** is a mapping $\mathcal{E} : \mathbb{R}^n \to \mathbb{R}^l$ that transforms a DS S of length n into a lower-dimensional series of length l, where $l \ll n$.

The lower bounding distance is a crucial property for similarity search, as it enables the exact retrieval of the nearest neighbors in a reduced l-dimensional space using the GEMINI-framework.

Definition 4: A **lower bounding distance (LBD)** $d'(\cdot, \cdot)$ for a distance measure $d(\cdot, \cdot)$ is a measure that satisfies the lower bounding property for DS A and B:

$$d'(A', B') \leq d(A, B)$$

In this context, $A' = \mathcal{E}(A)$ and $B' = \mathcal{E}(B)$ represent the summarizations of A B, respectively. (z-normalized) Euclidean LBD s have been defined for summarization techniques including iSAX [8], PAA [19], SFA [14], or DFT [20].

A. Exact Similarity Search using GEMINI

Indexing high-dimensional DS is a considerable challenge due to the *Curse of Dimensionality* [21]. As the dimensionality of the search space increases, the number of DS that must be examined typically grows exponentially. Consequently, executing an exact similarity query using spatial access methods (SAMs) can take longer than performing a sequential scan of all data [22], [23]. SAMs traditionally become ineffective at dimensionalities ranging from $n = 10$ to 20 dimensions [12].

Research in [12], [13] was the first to introduce the concept of dimensionality reduction of DS (summarization) prior to indexing. This approach, known as GEMINI, demonstrated that by using a lower-bounding distance measure on the summarization, queries in the reduced-dimensional search space return a super-set of the exact result as in the original search space. These false alarms are then pruned using the actual distance measure.

The rationale behind GEMINI is to compute the lower-bounding distances for each DS A relative to the query Q. The exact distance between A and Q is only calculated if the lower-bounding distance exceeds the current best-so-far exact distance computation. By providing a LBD, the Curse of Dimensionality is mitigated, reducing the effective dimensionality to $10 - 20$ dimensions of the summarization.

Since the introduction of GEMINI, numerous summarization techniques with lower bounding techniques have been proposed, which can be categorized into real-valued and discrete-valued (symbolic) methods. Research on symbolic series is scarce. Symbolic methods combine dimensionality reduction with quantizing. SAX [24] has established itself as the de-facto standard for exact similarity search on DS.

B. Distance calculations using SIMD

Single Instruction, Multiple Data (SIMD) is a parallel computing architecture that performs the same operation simultaneously on multiple data points, efficiently utilizing data-level parallelism [25]. By fetching instructions once and executing them in parallel, SIMD reduces latency and achieves speedups of up to 16x with modern CPUs supporting vector widths of up to 512 bits. In data series, SIMD has been used for Euclidean distance calculations [26] and integrated into indexing techniques like ParIS+ and MESSI to optimize lower-bound distance computations, improving similarity search performance [15], [27], [28].

III. Related Work

Since the introduction of the GEMINI framework [12] for exact similarity search on DS using lower bounding, a vast array of summarization techniques has emerged. The majority of research in this domain has focused on numeric techniques, which can be broadly categorized as follows: Piecewise Aggregate Approximation (PAA) [19] uses mean values over fixed-length segments. Adaptive Piecewise Constant Approximation (APCA) [29] employs mean values over adaptive segments. Piecewise Linear Approximation (PLA) [30] uses line segments to represent DS. Chebyshev Polynomials [31] are another method for DS summarization. The Fourier Transform [12], [20] utilizes the frequency domain representation. Wavelets [32] are also used for DS summarization. Despite advancements in lower bounding strategies, there remains a notable gap in research focused on leveraging numerical techniques for indexing massive datasets. This is partly due to the substantial memory requirements of these methods, but also as these show no clear improvement over symbolic representations. In their study, [14] compared several techniques based on pruning power, namely, APCA, PAA, PLA, CHEBY, and DFT. They conclude that none outperformed DFT. Moreover, SFA consistently matched or exceeded the performance of all but DFT across nearly all scenarios. SFA is outperformed by DFT due to SFA's additional quantization step applied on top of DFT. The results that there is no significant difference were independently confirmed by [11].

SAX [11] is a symbolic summarization technique that is commonly used for DS similarity search, as it has a much lower memory footprint requiring only a few bits (rather than a double/8 byte), and allows for lower bounding the ED, which is essential in finding the exact nearest neighbors [12]. The iSAX-family of indices is large [8], including iSAX [33], iSAX2.0 and iSAX2+ [34], ParIS+ [27], MESSI [16], ULISSE [35], Hercules [36], DET-LSH [37], and LeaFi [38]. MESSI [16] achieves SotA performance for in-memory DS similarity search in a single machine. Moreover, since all SAX-based indices [8] use the same summarization technique, they will all benefit from the improvements introduced here.

The research on symbolic transformations, which show that their proposed distance lower bounds the ED can be divided into SAX-based [8], [39]–[43] and SFA-based [14], [44]. SFA [14] is a data-adaptive transformation based on the frequency domain. It is most common in dictionary-based classification approaches [5], [45], [46]. An index was proposed based on SFA, named the SFA trie [14], inspired by building a prefix tree on the SFA word. However, it showed to be non-competitive to the SotA [4]. SFFA [44] extends on SFA using the Fractional Fourier transform, and a

supervised feature selection mechanism. We cannot use SFFA directly, as our data has no labels. ASAX [43] introduces fixed variable-length segments, which remain constant across series. SAX-CP [39] extracts variable-length segments using change points. Many SAX-based variants add trend features, increasing the total number of features. For example, SAX-TD [47], SAX-CP [39], TFSAX [40], TSAX [48], SAX-VFD [42], and IEPF-TSR [41] require $2w + 1$, $2w$, $2w$, $3w$, $4w$, and $3w$ features, respectively, for w segments. It remains unknown if these approaches outperform SAX when the number of features is kept constant by reducing the number of mean values proportionally [49]. Our requests for code to the corresponding authors of five papers published after 2019, received no response. Thus, we re-implemented SAX-TD [47] and ASAX [43] for the experiments. Surprisingly, both performed significantly worse than iSAX.

While other distance measures may be suitable in specific contexts, our focus is on Euclidean Distance (ED). Elastic distances, for instance, can be advantageous when working with limited data, such as in classification [5], [50] or clustering tasks [6]. However, [51] demonstrated that the error rate of ED is approaching that of Dynamic Time Warping (DTW) as the dataset size increases, rendering the difference negligible with a few thousand objects. Consequently, large-scale dataset indexing often favors ED due to its computational efficiency [8]. The Scalable Compound Infrastructure [52] introduces the concept of shift-insensitive SAX histograms to more effectively handle DS with gaps. It also incorporates (sub-)vector quantization for improved representation. Unlike both iSAX and SOFA, however, the resulting representation does not provide a lower bound for the Euclidean Distance (ED).

A similar, yet different area of research is on vector datasets [53]. Other than DS, vector data has no ordering. I.e. the values may be re-ordered in any arbitrary order. This can result in very high variance in high frequencies, which makes the application of summarization techniques hard. Thus, most research in vector data focuses on approximate retrieval [54]. FAISS [18] is a SotA approach for exact similarity search on vector datasets. In this work, we compare to the FAISS approach. Approximate similarity search [18] may not always yield exact results, but results that are very close. MASS [55] and the UCR suite [3] were originally designed for subsequence search, targeting the problem of finding the lowest distance subsequence in a long DS. Our paper, however, focuses on whole-series matching. MASS is less effective and up to 5 times slower than the UCR suite for this task (see Fig 3 in [56]). Thus, we did not include MASS in our experiments, but compare to the UCR suite.

Distributed systems have been developed to accelerate DS similarity retrieval. TARDIS [9] is an iSAX-based distributed index combining a centralized global index with local indices on cluster worker nodes, using word-level variable cardinality and K-ary trees (sigTree) for local indexing. It supports exact and approximate k-NN queries, and SOFA's improvements could be applied to it. CLIMBER [57] employs a novel feature extraction mechanism, recursively selecting random pivots to create a recursive Voronoi partition. Its two-level hybrid index, CLIMBER-INX, organizes data into groups using trie-based partitioning. PARROT [58] is a correlation-based framework for approximate similarity search, identifying patterns and exceptions within partitions. Its two-layer index includes a centralized global index (PARROT-G) and local indices (PARROT-L) for significant P-patterns in each partition.

IV. SOFA - Fast, Exact Sim. Search

SOFA (SymbOlic Fourier Approximation) is based on two building blocks: a tree index adapted from MESSI and the SFA symbolic summarization. In the following section, we first introduce MESSI for answering exact queries using the GEMINI approach for exact similarity search. From Section IV-C, we present the SAX summarization, and compare it to SFA, and finally sketch how to implement it within MESSI (Section IV-F).

A. MESSI

MESSI [15], [16] represents the state-of-the-art in-memory DS indexing [8]. It was specifically designed to exploit concurrent multi-threaded environments and for in-memory DS management for exact similarity search using the GEMINI framework. It utilizes variable cardinalities for iSAX summaries, varying the number of bits used for each segment's symbols, to build a hierarchical index.

B. Structure

This tree consists of three types of nodes. **Root Node:** Points to multiple child nodes (up to 2w in the worst case, where w is the number of bits), representing all possible iSAX summaries. **Inner Nodes:** Each has two children and holds an iSAX summary representing all series in its subtree. **Leaf Nodes:** Store iSAX summaries of several DS and pointers to these series. When the number of series in a leaf node exceeds its capacity, the leaf splits into two new leaves, becoming an inner node. This split is achieved by increasing the cardinality of one segment's iSAX summary to ensure a balanced distribution of series between the new leaves [34]). The iSAX summaries for the new leaves are then adjusted by setting the new bit to 0 and 1, respectively.

a) Exact Similarity Search: For query answering, MESSI first performs an Approximate Search by traversing the index tree to find the best candidate series. Then, it computes an approximate answer by calculating the real distance, called Best-So-Far (*BSF*), between the query series and the candidate series. The leaf with the smallest lower-bound distance to the query is identified, and *BSF* is used to prune as many candidate series as possible from the dataset during the tree parsing.

MESSI employs multiple *index workers* to traverse index subtrees in parallel, using *BSF* to determine which subtrees can be pruned. Each subtree is assigned to a single worker to minimize synchronization needs. Each worker only needs to synchronize during subtree selection. The leaves of unpruned subtrees are placed into a fixed number of priority queues,

ordered by the lower-bound distance between the PAA of the query and the iSAX summary of the leaf node.

The workers access these priority queues using locks. Each index worker processes a priority queue by repeatedly calling *DeleteMin()* to retrieve a leaf node. The worker checks if the lower-bound distance between the query and the leaf exceeds the current *BSF*. If it does, the leaf is pruned. Otherwise, the worker examines the series within the leaf by computing the lower-bound distance between the PAA of the query and the iSAX summaries of each series. If this lower-bound distance is less than the *BSF*, the worker calculates the real distance using the raw values further. If a series with a smaller distance than the current *BSF* is found, the *BSF* is updated.

When a worker encounters a node with a distance greater than the *BSF*, it abandons the current priority queue and selects another, as all remaining elements in the abandoned queue have higher distances and can be pruned. This process continues until all priority queues are processed, and the *BSF* is updated accordingly. The final value of the *BSF* is returned as the query result. Note that the above process is trivially extended to return the k-NN. For further details, readers are referred to [15], [16].

C. iSAX - A Static Symbolic Representation

The indexable SAX (iSAX) summarization technique combines (a) the Piecewise Aggregate Approximation (PAA) with (b) a fixed equal-depth binning quantization derived from the Normal distribution N(0,1). iSAX transforms a DS into a word as follows:

1) **Segmentation:** The DS is divided into l segments of equal length n/l.
2) **Aggregation:** Each segment is represented by its mean value, also referred to as Piecewise Aggregate Approximation (PAA) [19]. This step acts as a low pass filter, as high variance segments, like high-frequency segments, dropouts or noise, of a series are smoothed.
3) **Fixed Quantization:** The l segment (PAA) means are mapped to symbols using breakpoints that assume a Normal distribution of the data.

Quantization in iSAX is build upon the assumption that values sampled from DS follow a Normal distribution $N(0,1)$, with mean 0 and std 1. iSAX ignores the actual data distribution and applies equal-depth binning applied to the Normal distribution, given the fixed-size alphabet Σ. These bins are typically hard coded. SAX has two parameters: (a) the length l and (b) the number of symbols $|\Sigma|$. In practice, the number of symbols is a small number, with as few as 256 symbols, which can be represented by 8 bits. The number 256 is based on an 8-bit char accommodating 256 states, optimizing space use. More symbols would require 16-bits, doubling space, or slow bit operations, with negligible TLB efficiency gains (see Figure 10 of [34]). The transformation of a single series into a word takes $\mathcal{O}(n + l \times |\Sigma|)$ SAX makes explicit assumptions about the data and may lose information about high-frequency components or fine-grained details due to the averaging process. SAX provides a LBD to the ED called *mindist* [11].

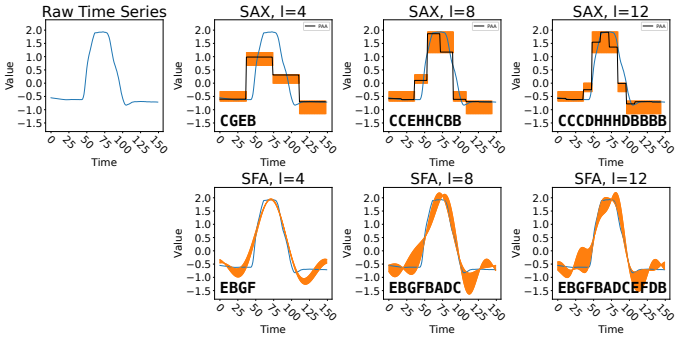

Fig. 2. The figure illustrates the summarization of a DS using SAX (top) and SFA (bottom), both employing an 8-symbol alphabet ('a' to 'h') with 4 to 12 values. SAX generates a staircase-like envelope around the raw signal (shown in orange). In contrast, SFA constructs an envelope around the Fourier transform, closely approximating the original signal.

D. SFA - A Learned Symbolic Representation

The Symbolic Fourier Approximation (SFA) combines (a) the Discrete Fourier Transform (DFT) with (b) feature selection and (c) binning of the actual distribution of the Fourier coefficients. SFA transforms a DS into a word as follows:

1) **Transformation:** A DS is transformed into the frequency domain using the DFT.
2) **Feature Selection:** Only a subset of $l < n$ Fourier values (real or imaginary part) is retained, as these capture the main structure of the data.
3) **Learned Quantization:** Each Fourier value is discretized into a symbol using equi-depth or equi-width binning based on the actual distribution of the frequency spectrum, i.e., the real or imaginary values.

SFA has two parameters: (a) the length l and (b) the number of symbols $|\Sigma|$. For SFA the same default values apply as for SAX. SFA is not based on any assumptions, and learns bins from the actual data distribution in Frequency domain. Yet, it can be more computationally intensive due to the Fourier transform with $\mathcal{O}(n \log n + l \times |\Sigma|)$ for each series, and binning based on actual data distribution.

Figure 2 illustrates the differences between both approaches, when using the same number of symbols and lengths $l \in [4, 8, 12]$ each. SAX gives a staircase like approximation of a DS, which for lower word lengths fails to capture the intrinsic shape of the signal. SFA represents a smooth envelop around the Fourier coefficients of the DS.

1) Transformation and Learned Quantization: The quantization in SFA is learned from the distribution of the real or imaginary Fourier values of the transformed series using a technique called **Multiple Coefficient Binning (MCB)**. The goal of MCB in SFA is to minimize the loss of information introduced by quantization. A better representation of the original signals enhances pruning efficiency during query execution. SFA learns l sets of quantization intervals using MCB, one for each real or imaginary Fourier value. First, all

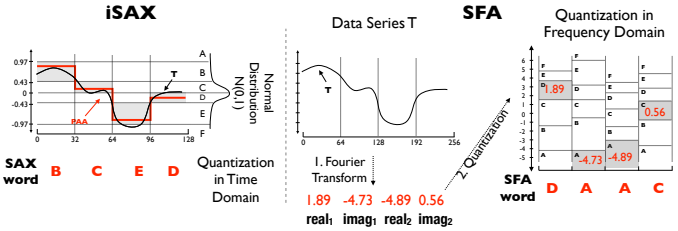

Fig. 3. The figure illustrates the two summarization techniques iSAX (left) and SFA (right). SAX aggregates the DS over intervals using PAA, and quantizes the mean values into symbols BCED using bins derived from equi-depth binning the Gaussian distribution. SFA transforms the DS into frequency domain, and separately quantizes the real and imaginary values into symbols using learned bins.

N series are transformed using the Discrete Fourier Transform (DFT). Next, l real and imaginary Fourier values are selected using a feature selection strategy (Section IV-D3).

2) Binning: When we separately consider the real or imaginary parts of Fourier coefficients, each part consists of a set of N real-valued numbers. The distribution of these real values is used to infer quantization bins. We choose a single (either real or imaginary) Fourier value, and learn $\alpha = |\Sigma|$ bins. This process is repeated for all l Fourier values. The equi-width binning process involves the following steps (Likewise, we may define the equi-depth binning): (a) Determine the value range: Compute the smallest (min) and largest (max) values among the N Fourier values of the N transformed samples. (b) Divide into Bins: Divide the range $[\min, \max]$ into α equal-width intervals (bins). The width of each bin is: $(\max - \min)/\alpha$. As a result of the j-quantization, the following bins are derived:

$$[\beta_j(a-1), \beta_j(a)) \text{ ,for } j \epsilon [0 \dots l) \text{ , } a \epsilon [1 \dots |\Sigma|)$$

We label bins by assigning the a-th symbol of alphabet Σ:

$$symbol^{(a)} \equiv [\beta_j(a-1), \beta_j(a)) \text{ ,for } j \epsilon [0 \dots l) \text{ , } a \epsilon [1 \dots |\Sigma|)$$

When proposed, equi-depth binning was used [14]. However, to achieve a tighter lower bound to the ED, it is essential to maximize the size of each interval. Equi-width proves to be superior, as it generates uniformly sized bins which are equally large, thus enhancing the accuracy of the lower bound (see Section V-B).

Figure 3 (right) illustrates the process of transforming a series into a word. iSAX (left) uses the same set of intervals based on the Normal distribution for all segment means, resulting in words, like *BCED*. SFA (right) first transforms the series into real and imaginary Fourier values, selects those with the highest variance, and learns different sets of bins for each value. These bins are then used to transform the series into words, like *DAAC*.

3) Novel Feature Selection: Commonly, for SFA the first Fourier values are retained, which acts as a low-pass filter. However, this approach reduces high variance components of the DS, which leads to degenerated performance for lower bounding. To address this limitation, we introduce a novel *variance-based* Fourier value selection strategy. The rationale behind this strategy is that maximizing the LBD requires maximizing the width of the quantization bins (intervals).

Given a dataset of DS D, the variance of each Fourier coefficient, we select those l coefficients of each transformation with highest variance:

$$\text{BEST} = \text{K-ARGMAX}\left(VAR(DFT\left(D\right), axis = 1)\right)$$

This strategy begins by computing the variance of each real and imaginary component of the Fourier transform for the N DS. It then selects the top l real and imaginary components with the highest variance.

The reason a larger variance improves the LBD is that it allows for wider quantization bins. Wider intervals capture more variability in the data, leading to a more accurate and informative representation, which in turn enhances the pruning efficiency during query execution. We will show the impact of this selection strategy in our ablation studies (Section V-C).

4) ED Lower Bounding Distance: Given the DFT representations $DFT(A) = A' = (a'_0, \dots, a'_{l-1})$ of DS A and $DFT(B) = B' = (b'_0, \dots, b'_{l-1})$ of DS B the *DFT LBD* to the ED d_{ED} is defined as [59] (for $l < n/2$):

$$d^2_{DFT}(A', B') = (a'_0 - b'_0)^2 + 2\sum_{i=1}^{l-1}(a'_i - b'_i)^2 \leq d^2_{ED}(A, B) \tag{1}$$

Note that the first DFT coefficients a'_0 and b'_0 represents the mean value, which is 0 for z-normalized DS, thus can be omitted. The *SFA Euclidean LBD* between a DFT representation $DFT(B) = B' = (b'_0, \dots, b'_{l-1})$ and an SFA representation $SFA(a) = A' = (a'_0, \dots, a'_{l-1})$ is calculated by exchanging the pairwise difference of the numerical values in Equation 1 by a $dist_i$ function, which measures the distance between the i-th symbol and the i-th numerical value [14]:

$$d^2_{SFA}(A', B') = mind_0(a'_0, b'_0)^2 + 2\sum_{i=1}^{l-1} mind_i(a'_i, b'_i)^2$$
$$d^2_{SFA}(A', B') \leq d^2_{ED}(A, B)$$

Again, the first term is 0 in the case of z-normalized DS and can be omitted. The distance $dist_i$ between a numerical value b'_i and symbol a'_i, represented by its lower and upper breakpoints $a'_i = [\beta_i(a-1), \beta_i(a))$, is defined as the distance to the lower breakpoint if b'_i is smaller or the upper quantization breakpoint if b'_i is larger:

$$mind_i(a'_i, b'_i) \equiv \begin{cases} 0, & \text{if } b'_i \epsilon [\beta_i(a-1), \beta_i(a)) \\ \beta_i(a-1) - b'_i, & \text{if } b'_i < \beta_i(a-1) \\ b'_i - \beta_i(a), & \text{if } b'_i > \beta_i(a) \end{cases} \tag{2}$$

Figure 4 (right) illustrates the SFA $mind_i$ in contrast to the iSAX lower bounding definition (left). For both the distance between the word $A' = DDD$ and $B' = (b_1, b_2, b_3)$ are computed. SFA uses learned bins per Fourier value, as opposed to the one set of fixed intervals used in iSAX.

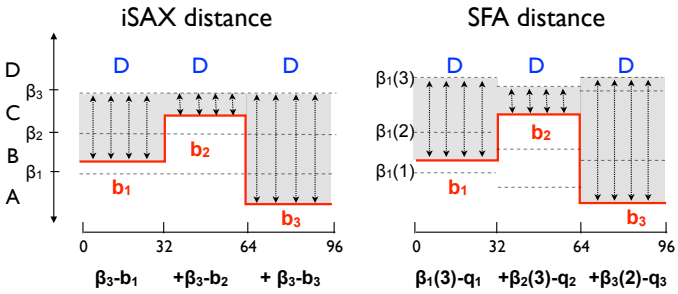

Fig. 4. A comparison of the iSAX (left) and SFA (right) Euclidean LBD. iSAX uses the same fixed break points for each PAA value. SFA uses learned break points for each Fourier value (mean or imaginary values).

Algorithm 1 MCB Quantization

Require: A set X, of data series of length n, with $|X| = N$, the number of coefficients l (default 16), the alphabet size a (default 256), the sampling ratio r (default 1%)

1: *Step 1: Sampling and Discrete Fourier Transform*
2: $X_{subsample} = sample(X, r)$
3: $X_{DFT} = DFT(X_{subsample})$

4: *Step 2: Determine l best coefficient indices by variance*
5: BEST_L = K-ARGMAX $(VAR(X_{DFT}, axis = 1), k = l)$
6: $X_{best} = X_{DFT}[:, best\_l]$

7: *Step 3: Learn l-sets of bins using binning*
8: BINS = *array of size* $(l \times a)$
9: **for** $j = 1$ to l **do**
10: BINS[J] = APPLY-BINNING$(X_{best}[j], a)$
11: **end for**

12: **return** BINS, BEST_L

Algorithm 2 SFA Transform

Require: A single data series T of length n, quantization bins *bins* of size $(l \times a)$, Fourier coefficient indices *best_l*

1: *Step 1: Transform T keeping only best l coefficients*
2: $T_{DFT} = DFT(T)$
3: $T_{best} = T_{DFT}[best\_l]$
4: *Step 2: Apply quantization*
5: $T_{SFA} =$ APPLY-QUANTIZATION$(T_{best}, bins)$
6: **return** T_{SFA}

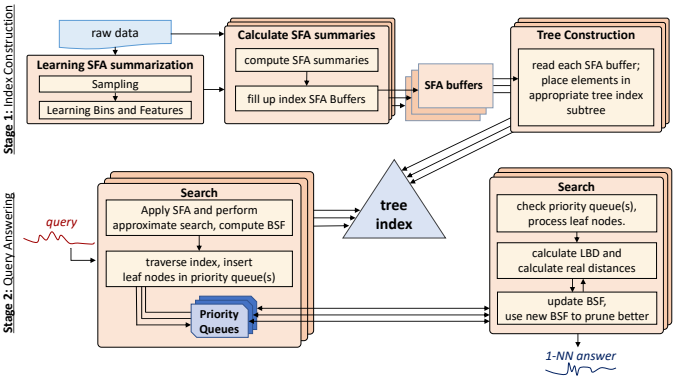

Fig. 5. Workflow of SOFA for exact similarity search. First, a fraction of the DS is sampled and Fourier transformed. Bins are learned, and the best Fourier coefficients selected. Using the learned transformation, all DS are transformed to create the index. To answer a query, the query DS is SFA transformed, and the MESSI-based index is used to retrieve the exact 1-NN using the SFA lower bound.

E. Pseudo-Code

The pseudo-code for SFA is given in Algorithms 1 and 2. MCB is used to learn quantization bins and the best Fourier coefficient indices. SFA transform is used to transform a single series using the learned bins. MCB (Algorithm 1) begins by sub-sampling the dataset using a sampling ratio r (line 2), with a default value of 1%. The impact of varying r is explored in the experiments (Section V-B0g). The subsample is then transformed using the Discrete Fourier Transform (line 3). Subsequently, the real and imaginary Fourier values are ranked by selecting the indices that maximize variance across the subsample (line 5). These top Fourier values are retained for further processing (line 6). Finally, l sets of bins, each containing a bins, are learned using either equi-width or equi-depth binning (lines 8–11). Please note that the real or imaginary Fourier values are simply real numbers, thus we may use any default implementation of equi-depth or equi-width binning to derive bins. The algorithm returns the optimal bin boundaries and selected Fourier coefficient indices.

The SFA transform (Algorithm 2) operates on a single DS using the pre-learned breakpoints (using MCB) and selected Fourier coefficient indices. First, the DFT is applied to the series (line 2), and only the top-ranked coefficients are retained (line 3). Finally, these coefficients are quantized based on the learned breakpoints to generate words (line 5). Although the alphabet size, length and binning strategy could be learned for each dataset, we achieved best results by using fixed values of $\alpha = 256$, $l = 16$ and *equi-width*.

F. SOFA index

The *SOFA (SymbOlic Fourier Approximation)* index is based on the MESSI index and the SFA representation. It allows for fast and exact similarity search is illustrated in Figure 5.

The process begins by sampling a fraction (1%) of the DS to learn the SFA summarization with alphabet size 256 (compare Section IV-C). The sample undergoes a Fourier transformation, followed by the learning of quantization bins from the Fourier values (Section IV-D1). Then, 8 learned Fourier coefficients are selected based on highest variance, which is equal to 16 imag and real float values. Utilizing the learned SFA, all DS are transformed and indexed. To answer a query, the query is first transformed using SFA, and then the index is used to search exact nearest neighbors using GEMINI (Section II-A), and the SFA lower bound (Section IV-D4).

G. SOFA LBD calculation using SIMD

Similar to MESSI, SOFA uses SIMD to accelerate both lower-bound and real distance calculations. Algorithm 3 and Figure 6 illustrate the process of lower bound distance (LBD) computations between a query's DFT coefficients and an SFA word of an indexed series using SIMD. Using SIMD enables parallel computation of 8 or 16 distances in a single instruction, utilizing 256-bit or 512-bit vectors with eight 32-bit floating-point elements each. The SIMD implementation

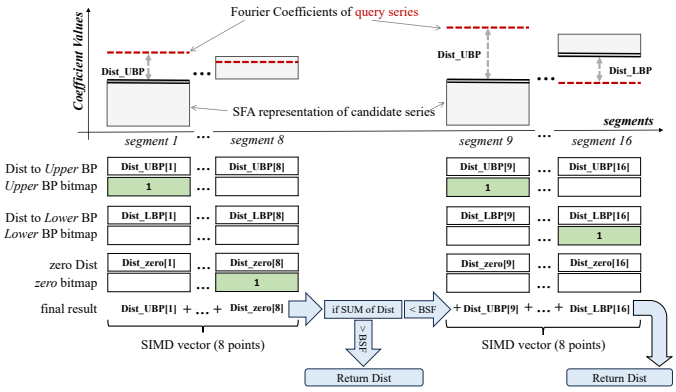

Fig. 6. SIMD lower bound distance calculation illustration using bitmaps for conditional branching, and chunks for early abandoning.

Algorithm 3 SIMD-Optimized Min Distance Calculation

Require: quantization bins *bins*, FFT coefficient representation of query data F_Q, SFA representation of candidate S_C, BSF bsf.

1: Initialize vectors V_{F_Q},V_{S_C}, V_D, $i = 0$, $D_C = 0$
2: **for** ith SIMD block size data **do**
3: $V_{F\_Q} \leftarrow F_Q[i]$
4: $V_{S\_C} \leftarrow S_C[i]$
5: V_{B_U}, $V_{B\_L} \leftarrow$ Gather_bound (V_{S_C}, $bins$)
6: V_{D_U}, V_{D_Z}, $V_{D\_L} \leftarrow$ Caldist(V_{F_Q}, V_{B_U}, V_{B_L})
7: V_{M_U}, V_{M_Z}, $V_{M\_L} \leftarrow$ Genmask(V_{F_Q}, V_{B_U}, V_{B_L})
8: $V_D \leftarrow (V_{D\_L} and V_{M\_L}) or (V_{D\_U} and V_{M\_U})$
9: $D_C \leftarrow Sum(V_D)$
10: **if** D_C exceeds bsf **then**
11: **return** D_C
12: **end if**
13: $i++$
14: **end for**
15: **return** D_C

addresses two key challenges: (i) optimizing branching based on upper, lower, and zero bounds, and (ii) enabling early termination while utilizing SIMD registers.

a) Conditional Branching: Unlike simpler branching techniques [26], LBD computations require distinct conditional branches and specific assignments for each SIMD vector position. To compute the LBD between the DFT coefficients of the query series and an SFA word, three conditions are evaluated: (i) the DFT coefficient lies above the *UPPER* breakpoint of the corresponding SFA representation of the candidate series (quantization interval), (ii) below the *LOWER* breakpoint, or (iii) within the SFA quantization interval, denoted as *ZERO* (compare Figure 4 and Eq. 2). All three conditions have to be processed simultaneously across SIMD positions, and a conditional mask extracts the correct result.

b) Early Abandoning: The Fourier coefficients in SFA are prioritized by relevance (compare Section IV-D3), with high-variance coefficients contributing more to the overall distance between two series. While SIMD processes calculations in parallel batches, it cannot inherently leverage early abandoning after each computation. Figure 6 provides an overview of the complete SIMD LBD computation process. Initially, the distance between each query's Fourier coefficient and the SFA word's intervals is calculated (top) using Eq. 2. The query is then processed in chunks of 8 points (bottom). After processing each chunk, the current distance is compared to the BSF, and if it exceeds the BSF, the current distance is returned. To ensure efficient branching, *UPPER*, *LOWER*, and *ZERO* conditions are evaluated using bitmaps, and aggregated into the final distance. Fully vectorizing all calculations enhances the speed of distance computations while minimizing the overhead of switching between vector and scalar registers, which operate on separate hardware.

c) Pseudocode: To address *early abandoning*, Algorithm 3 splits the coefficients into smaller chunks of up to 8 data points for 256 vector size (lines 2-14), enabling SIMD acceleration. After processing each chunk, intermediate results are aggregated, allowing early abandoning if the cumulative sum exceeds the best-so-far distance (lines 10-12). This approach combines the speed of SIMD with the efficiency of early abandoning, optimizing the balance between performance and computational savings.

To address *conditional branching*, Algorithm 3 generates three branch masks (*UPPER*, *LOWER*, and *ZERO*), with each mask containing a value of *1* at positions in the SIMD vector where the corresponding condition is met (line 7). I.e., if the first segment of the query lies above the interval of the word representation, the *UPPER* mask is set to *1* for that position. The distance value for the *UPPER* branch is then selected for that position. Using SIMD instructions (e.g., AVX, AVX2, SSE3) [60], these masks are generated efficiently (lines 7). A logical *AND* operation is applied between each branch result and its corresponding mask, setting irrelevant branch results to zero (lines 8). Finally, the results from all branches are combined into a single vector, retaining only the correct values for each position and completing the computation.

V. Experimental Evaluation

In this section, we present a threefold experimental evaluation on 17 real datasets, with a total of 1 billion DS or $\sim$ 1 TB of data. First, we outline our competitors, setup, and datasets.

a) Competitors: We compared **SOFA** with **MESSI**, **UCR Suite-P** [17], and **FAISS IndexFlatL2** [18]. UCR Suite-P is a parallel implementation of the state-of-the-art optimized serial scan technique using SIMD. In UCR Suite-P, each thread is allocated a segment of the in-memory DS array, allowing all threads to concurrently and independently process their assigned segments. The real distance calculations are performed using SIMD, and synchronization occurs only at the end to compile the final result. For FAISS, we used the CPU implementation of IndexFlatL2, the exact similarity search index under L2 (ED). The design principle for MESSI and SOFA involves sequential query processing, handling queries one after another. This approach simulates an exploratory analysis scenario where users generate new queries based on the results of previous ones. FAISS, however, cannot leverage parallelism within single query processing. Therefore, to take advantage of FAISS's capabilities, we process queries in mini-batches equal to the number of available cores, which exploits the inherent embarrassingly parallel nature of batch processing.

FAISS (IndexFlatL2) was installed with Intel MKL support, and the number of OMP threads was configured according to the available cores in each experiment. The code, scripts, and notebooks for all algorithms are available online [28].

b) Setup: We used a server with 2x Intel Xeon 6254 3.1Ghz CPUs (18 cores each, and 36 cores in total) and 756GB RAM. All algorithms were implemented in C, and compiled using GCC v7.5.0 on OpenSuse Linux v15.5. As we have 36 cores available, we performed experiments with 9, 18, 36 cores, each. For both MESSI and SOFA, we used consistent configurations across all experiments: 'initial-lbl-size', 'min-leaf-size', and 'leaf-size' were set to 20000, with the queue size matching the number of cores. To ensure a fair comparison, the alphabet size is fixed at 256 and the word length at 16 for both SAX and SFA across all datasets. Unless stated otherwise, SOFA employs the *equi-width* binning strategy. In other words, these parameters are not optimized individually for each dataset but are kept constant across all experiments. SFA quantization was learned using a fraction of 1% of the dataset, and selected the highest variance Fourier values (real or imaginary) from the first 16 Fourier coefficients. As we test the batch case, where all data is available, we chose 1% to avoid overfitting the train dataset, which could lead to decreased performance on the independent query data.

c) Datasets: Table I provides a summary of the properties of the 17 datasets uted in this study. They originate from various fields, including seismology (ETHZ, Iquique, LenDB, NEIC, OBS, SCEDC), astronomy (Astro), neuroscience (SALD), and vector datasets (Deep1B, BigANN, SIFT1b). Only 5 of these datasets (Astro, BigANN, Deep1B, SALD, SIFT1b) have been previously used for similarity search. Each is accompanied by a distinct set of 100 queries. 12 datasets are seismic datasets sourced from [61]. The samples were obtained by segmenting the seismic data into non-overlapping windows. Queries were generated using only the location of the primary (P)-wave of a seismic event, which travels faster than the secondary (S)-wave. The dataset sizes vary from 500k to 100M series, with series lengths ranging from 96 to 256 floats. Overall, there are 1 Billion real DS, corresponding to $\sim$ 1 TB on disk. We used distinct sets of 100 query series for each dataset, ensuring they were kept separate from the indexed data.

TABLE I
CHARACTERISTICS OF THE 17 DATASETS USED

Dataset Name	# of Series	Series Length
Astro [62]	100,000,000	256
BigANN [63]	100,000,000	100
Deep1b [64]	100,000,000	96
ETHZ [61]	4,999,932	256
Iquique [65]	578,853	256
ISC_EHB_DepthPhases [66]	100,000,000	256
LenDB [67]	37,345,260	256
Meier2019JGR [61]	6,361,998	256
NEIC [68]	93,473,541	256
OBS [69]	15,508,794	256
OBST2024 [70]	4,160,286	256
PNW [71]	31,982,766	256
SALD [72]	100,000,000	128
SCEDC [73]	100,000,000	256
SIFT1b [74]	100,000,000	128
STEAD [75]	87,323,433	256
TXED [76]	35,851,641	256

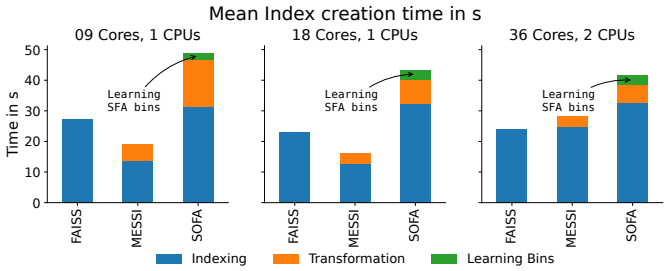

Fig. 7. Comparison of mean indexing times using 9, 18 or 36 cores. MESSI is faster than SOFA. SFA involves some overhead for learning SFA bins (green) and the DFT (orange). Using more CPUs (18 to 36 cores) may increase runtime due to synchronization overhead.

A. Index Creation

Figure 7 presents the mean runtime results for index creation across three systems: SOFA using SFA, MESSI using SAX for summarization, and FAISS (IndexFlatL2, CPU) using all 17 datasets. The measurements exclude I/O times for reading datasets from disk. On average, index creation times range from 15 seconds to 1 minute. MESSI is the fastest, averaging around 15 seconds, followed by FAISS. There is minimal improvement in index creation times when utilizing more cores. Notably, we observed an increase in index creation times when scaling from one CPU (18 cores) to two CPUs (36 cores), due to an increased overhead in synchronization. The SFA method in SOFA incurs an overhead for learning the quantization from a 1% sample of the data. This overhead is negligible compared to the time required for transformation and index creation. Overall, SFA has a higher transformation time than SAX, due to the use of the Fourier transform, which has a complexity of $\mathcal{O}(n \log n)$ in the DS length n, compared to $\mathcal{O}(n)$ for PAA. Index creation times are also higher for SOFA, potentially indicating more node splits. We measured the average indexing speed of each method on the dataset. The results are as follows using 36 cores: FAISS achieved a rate of 2.3 million series per second, MESSI indexed at 1.9 mio, and SOFA 1.4 mio. While SOFA is $\sim$ 30% slower than MESSI at indexing, this is faster than real time for most applications.

Overall there is little difference in the structure of the indices over all 17 datasets (see supporting webpage). SOFA has a slightly higher average tree depth and a smaller fill degree of the leafs (center). The fanout of the root node is slightly lower for SOFA. For index construction, MESSI divides the data into chunks, processes each chunk independently, and then merges them into subtrees, which are subsequently combined into the full index. Therefore, the structure of the index may vary depending on the number of workers or cores used.

TABLE II
MEAN AND MEDIAN 1-NN QUERY TIMES IN MS FOR MIXED WORKLOAD ON 17 DATASETS. SOFA IS FASTEST.

Method	Cores	median	mean
FAISS IndexFlatL2 [18]	09	358	**510**
	18	206	309
	36	248	344
MESSI	09	335	932
	18	185	486
	36	112	299
SOFA	09	**149**	581
	18	**81**	**293**
	36	**58**	**209**
UCR SUITE-P [17]	09	1448	1654
	18	790	867
	36	557	587

TABLE III
MEDIAN k-NN QUERY TIMES IN MS FOR MIXED WORKLOAD ON THE 17 DATASETS USING 36 CORES. SOFA STAYS FASTEST.

Method	**1-NN**	**3-NN**	**5-NN**	**10-NN**	**20-NN**	**50-NN**
UCR suite	557	-	-	-	-	-
FAISS	248	283	276	284	307	314
MESSI	112	139	145	181	193	209
SOFA	58	70	70	83	87	98

B. Exact Similarity Search

a) 1-NN Exact Search: Table II shows the results for the exact 1-NN similarity search under ED. We report mean and median query times over all 17 datasets. SOFA is the fastest method for all configurations, except for median on 9 cores, where FAISS is fastest. On average, SOFA is > 10 times faster than UCR SUITE-P, 2-4 times faster than FAISS and 2-3 times faster than MESSI. All methods scale with the number of cores, except FAISS, which degenerates from 18 to 36 cores. With SOFA an exact 1-NN similarity query can be answered in 58 ms in the median case on our hardware and with our datasets, faster than a blink of an eye.

b) k-NN Exact Search: Table III presents the results for the exact k-NN similarity search using ED. We report the median query times across all 17 datasets, utilizing 36 cores. Among the methods, SOFA consistently demonstrated the fastest performance. For this experiment, we did not compute the full k-NN results for the UCR suite, as even for 1-NN, its query times were already an order of magnitude larger compared to SOFA. Overall, SOFA exhibits the best scalability as the number of nearest neighbors increases 8. Notably, all methods scale efficiently with increasing k.

c) Scalability in the number of cores: Figure 9 displays the query times for all datasets using box-plots, illustrating performance across an increasing number of cores. SOFA consistently shows the lowest median runtimes. However, both MESSI and SOFA exhibit high variance in query times across the different datasets. In contrast, the query times for FAISS and the UCR SUITE are more tightly clustered around their median values, indicating more consistent performance.

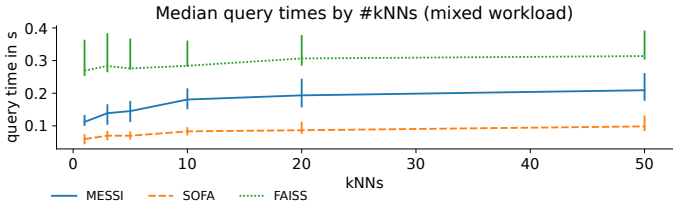

Fig. 8. A comparison of median query times with an increasing number of nearest neighbors k. SOFA shows overall lowest query times. Notably, all methods scale efficiently with increasing k.

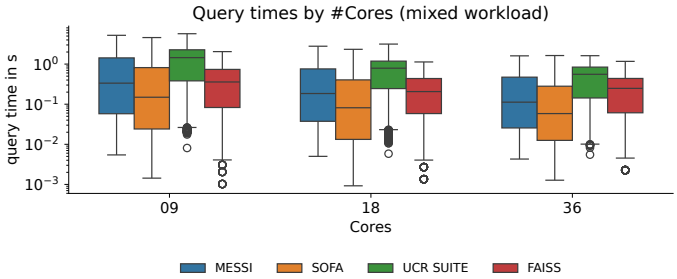

Fig. 9. A comparison of 1-NN query times with an increasing number of cores on a logarithmic axis. SOFA shows overall lowest query times with most queries in the range of 100ms and some in the range of a few ms.

d) Scalability in Leaf Size: Figure 10 displays the impact of increasing leaf sizes on the overall query times. With increasing leaf size, the query time decreases and plateaus around 10k series. We used 20k as the default value for both MESSI and SOFA.

e) SOFA vs MESSI: To evaluate the impact of using SFA versus SAX, we conducted a comparative analysis between MESSI and SOFA. Figure 11 illustrates the relative improvement in average query times of SOFA compared to MESSI, with MESSI serving as the baseline (100%). The results demonstrate that SOFA consistently outperforms its MESSI across all datasets, with some cases showing remarkable improvements of up to 38 times faster query processing (notably on the LenDB dataset). Figure 1 highlights datasets where SOFA exhibits its most significant performance gains. In these cases, the SAX summarization technique employed by MESSI proves inadequate, as the mean-based segmentation results in oversimplified straight-line representations, particularly for datasets that do not follow a Normal distribution. In contrast, SOFA's use of SFA allows for better adaptation to diverse data characteristics, resulting in substantially improved query times—up to 40 times faster in some instances.

This performance disparity underscores the superiority of SFA in scenarios with high-frequency DS, such as complex data patterns, and its ability to provide more accurate and efficient summarizations across a wide range of dataset types. The adaptive nature of SFA in SOFA proves particularly advantageous for datasets where traditional SAX-based approaches fall short, highlighting the importance of choosing appropriate summarization techniques.

f) Vector Datasets: To demonstrate the versatility of SOFA beyond DS, we evaluated its performance on six widely

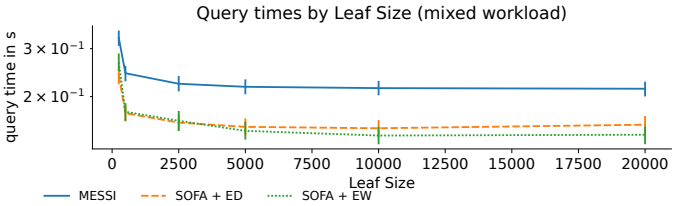

Fig. 10. Comparison of 1-NN query times with increasing leaf size. Query times decrease with increasing leaf node size and plateau around 10k series.

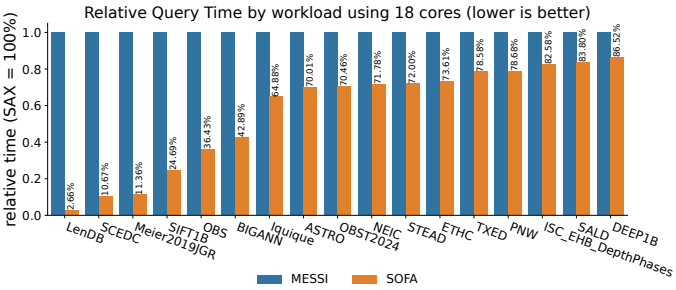

Fig. 11. Comparison of relative query times for 1-NN using 18 cores. The runtime improvement of SOFA over MESSI can be up to 38x faster (LenDB).

used vector datasets derived from image and text embeddings. The results, shown in Figure 12, reveal that 1-NN query times on these datasets are significantly higher than those observed for SOFA datasets (Table III). Specifically, the median query times are as follows: FAISS at 345 ms, MESSI at 496 ms, SOFA at 324 ms, and UCR SUITE at 630 ms. Notably, TEXTTOIMAGE and TURINGANN are particularly challenging, with response times approaching 1 second—comparable to those of a sequential scan. Despite these challenges, SOFA consistently achieves the lowest 1-NN query times among the methods tested. However, these times are approximately six times higher than those reported for DS. Unlike DS, vector data lacks explicit ordering. This absence of structure makes it difficult to approximate such data using methods like sine waves (SFA) or mean values (SAX), presenting a significant challenge for indexing and querying.

g) Effect of Sampling on Query Times: SFA utilizes 1% of the indexing samples to learn quantization bins. To explore the trade-offs associated with varying sampling rates, we conducted an experiment (Section IV-F). As shown in Table IV, the median query times stabilize at around a 1% sampling rate, reaching 58 ms. However, the mean query times continue to improve up to a sampling rate of 5%. Conversely, using less than 1% of the data results in a slight increase in mean and median query times.

h) When SOFA excels: SFA excels on datasets with rapid changes by selecting frequencies based on their largest variance (Section IV-D1). To investigate the hypothesis that higher variance in frequency correlates with greater speedup, we examined the mean index of the Fourier coefficients selected by SOFA and its corresponding speedup over MESSI for each dataset. For instance, if SOFA selects eight Fourier coefficients with indices $[8, 9, 10, 11, 12, 13, 14, 15]$, their mean index would be 11.5. The highest possible index that can be selected is 32. Our experiment (Figure 13) reveals a clear trend, with a positive Pearson correlation of 0.51. This suggests that Fourier coefficients corresponding to higher frequencies are associated with increased speedup—indicating that when SOFA prioritizes higher-frequency coefficients, it tends to outperform MESSI. The datasets used in this experiment cover a wide range of frequency variances. Those characterized by low frequency variance include: Meier2019JGR, ASTRO, Iquique, NEIC, ETHZ, PNW, and SALD.

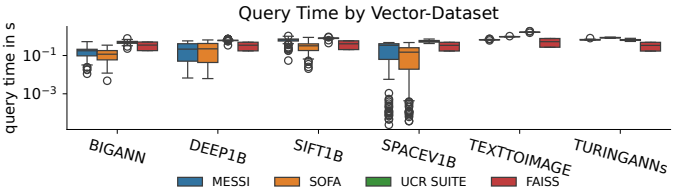

Fig. 12. A comparison of 1-NN query times on vector datasets.

TABLE IV
PERFORMANCE OF SOFA AT DIFFERENT SAMPLING RATES ON THE 17 DATASETS USING 36 CORES.

Method	Sampling	Mean Time in ms	Median Time in ms
SOFA	0.1%	220	67
	0.5%	211	64
	1%	209	58
	5%	192	61
	10%	192	62

We analyzed the average frequency of each dataset in the SOFA collection by calculating the zero-crossing rate (ZCR) [77], which measures how often a signal crosses the zero axis. The ZCR was averaged across all samples, serving as an indicator of the signal's dominant frequency [77]. To enable comparisons across datasets, we normalized the ZCR to a range of $[0, 0.5]$. For example, a ZCR of 10% indicates that a dataset segment of length 256 has, on average, 26 zero crossings, while 50% is the maximum, and represents one zero crossing every other value. Figure 14 illustrates that the SOFA datasets span a wide range of frequencies, from very high (vector data) to low. SOFA demonstrates strong performance across all frequency ranges (compare query times in Figure 11). Interestingly, while there is some overlap between frequency characteristics and performance trends, the relationship is not entirely linear. There are performance gains for the very-high frequency datasets, like Deep1B (49% ZCR and 16% speedup), or the very low-frequency datasets such as ISC-EHB (1.3% ZCR and 21% speedup). Though, SOFA's improvements over MESSI are most pronounced for mid- to high-range frequencies, such as SCEDC (23% ZCR and $10x$ speedup) and SIFT1b (31.5% ZCR and $4x$ speedup).

C. Ablation Study

We conducted an ablation study using the *tightness of lower bound (TLB)* metric, two benchmark datasets and the *aeon*

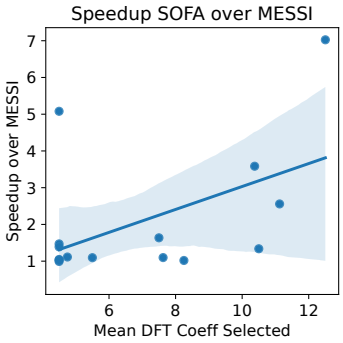

Fig. 13. Comparison of the average index of the Fourier coefficients selected by SOFA against its speedup, relative to MESSI. Each point represents a single dataset. The results indicate a clear trend: higher selected frequencies are generally associated with greater speedup.

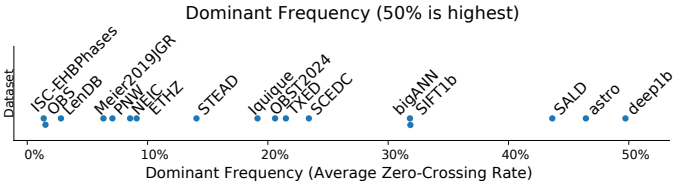

Fig. 14. Average dominant frequency of SOFA datasets, as measured by the zero-crossing-rate (ZCR), with 50% (i.e., max) being every second value.

TABLE V
MEAN TLB ON UCR DATASETS FOR INCREASING ALPHABET SIZES.

Alphabet Size Method	4	8	16	32	64	128	256
SFA ED +VAR	**0.56**	**0.67**	**0.73**	**0.77**	0.79	0.80	0.80
SFA EW +VAR	0.53	0.65	**0.73**	**0.77**	**0.80**	**0.81**	**0.81**
iSAX	0.40	0.51	0.59	0.65	0.69	0.71	0.73
ASAX	0.17	0.31	0.42	0.49	0.52	0.53	0.53
SAX-TD	0.11	0.21	0.30	0.37	0.42	0.45	0.47

TABLE VI
MEAN TLB ON SOFA DATASETS FOR INCREASING ALPHABET SIZES.

Alphabet Size Method	4	8	16	32	64	128	256
SFA ED +VAR	**0.41**	**0.49**	**0.54**	0.57	0.58	0.60	0.61
SFA EW +VAR	0.34	0.47	**0.54**	**0.59**	**0.61**	**0.63**	**0.64**
iSAX	0.37	0.45	0.45	0.52	0.54	0.55	0.55
ASAX	0.20	0.33	0.40	0.45	0.48	0.51	0.51
SAX-TD	0.12	0.22	0.31	0.37	0.40	0.41	0.42

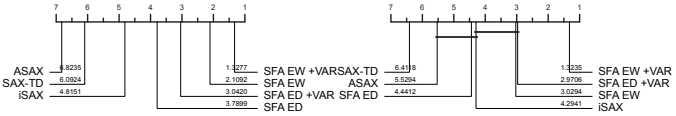

Fig. 15. Critical Difference plot on average ranks of TLB on 120 UCR (left) and SOFA (right) datasets for $l = 16$ and $\alpha = 256$ (lower rank is better).

framework [78]. The TLB is defined as the LBD over the true distance [19]. Higher is better, and the TLB reaches one if both are equal. A tighter (higher) TLB results in a better pruning of the search tree and thus a lower runtime. We used the *UCR dataset archive* [3], which consists of approximately 120 datasets spanning a wide range of applications. Each dataset includes a training and test set. The training set was used to learn SFA, while the test set served as queries. We also employed the 17 *SOFA datasets* used in this study (Table I). Each dataset is split into an indexing set for learning SFA and a query set for performing queries.

We test SFA in combination with equi-width (EW) or equi-depth (ED) binning, with and without using the variance for feature selection. For all summarizations we used length $l = 16$. For SAX-TD [47], we use 8 mean and 8 trend features. For ASAX [43] we use 16 variable length segments. Tables V and VI reveal that the TLB improvement is notably higher for smaller alphabet sizes, and is up to 16 percentage points higher for alphabet size 4 on the UCR datasets. Overall, SFA using EW and variance for feature selection yields the highest TLB. ASAX and SAX-TD perform worst among all competitors. For ASAX, using variable-length segments appears to yield inferior results compared to fixed-length segments. Additionally, incorporating trend features in SAX-TD seems to provide less informative representations than simply using twice as many mean values, as done in iSAX. A high TLB enables the pruning of ED calculations, which is a key reasons why SOFA outperforms MESSI in similarity search. Finally, we use critical difference diagrams to compare mean ranks of approaches, with horizontal bars indicating statistically indistinguishable cliques (p-value 0.05, Wilcoxon-Holm post-hoc analysis). Figure 15 shows that SFA EW+Var achieves significantly better ranks for both benchmarks, followed by ED and finally iSAX. Variance-based feature selection ensures optimal bin sizes. Equi-width outperforms equi-depth by avoiding small bins in z-normalized DS, which occur when using 256 equi-depth bins in the -1 to +1 range. Again, SAX-TD and ASAX are inferior to the other approaches.

VI. CONCLUSIONS

We introduced SOFA, a fast and exact index for data series (DS) similarity search that uses the Symbolic Fourier Approximation (SFA), a new symbolization technique that is learned from the data distribution in the frequency domain. SFA demonstrates its advantage across a variety of datasets. In our extensive experimental evaluation on a comprehensive benchmark comprising 17 real use cases covering 1 billion DS and ~ 1 TB, we show that SOFA is up to 38x faster than MESSI, and on average $2-10$ times faster than the SotA for exact similarity search. In our future work, we plan to study techniques for approximate similarity search using SFA, as well as other distance measures, symbolic representations, and parallelization opportunities through the use of GPUs.

ACKNOWLEDGMENTS

Supported by EU Horizon projects AI4Europe (101070000), TwinODIS (101160009), ARMADA (101168951), DataGEMS (101188416) and RECITALS (101168490), and by National Natural Science Foundation of China (62202450).

REFERENCES

[1] T. Palpanas, "Data series management: The road to big sequence analytics," *SIGMOD Record*, 2015.

[2] K. Echihabi, K. Zoumpatianos, and T. Palpanas, "Scalable machine learning on high-dimensional vectors: From data series to deep network embeddings," in *WIMS*. New York; NY; United States: Association for Computing Machinery, 2020.

[3] H. A. Dau, A. Bagnall, K. Kamgar, C.-C. M. Yeh, Y. Zhu, S. Gharghabi, C. A. Ratanamahatana, and E. Keogh, "The UCR time series archive," *IEEE/CAA Journal of Automatica Sinica*, vol. 6, no. 6, pp. 1293–1305, 2019.

[4] K. Echihabi, K. Zoumpatianos, T. Palpanas, and H. Benbrahim, "The lernaean hydra of data series similarity search: an experimental evaluation of the state of the art," *PVLDB*, vol. 12, no. 2, pp. 112–127, 2018.

[5] M. Middlehurst, P. Schäfer, and A. Bagnall, "Bake off redux: a review and experimental evaluation of recent time series classification algorithms," *Data Min. Knowl. Discov.*, pp. 1–74, 2024.

[6] C. Holder, M. Middlehurst, and A. Bagnall, "A review and evaluation of elastic distance functions for time series clustering," *KIS*, vol. 66, no. 2, pp. 765–809, 2024.

[7] P. Wenig, S. Schmidl, and T. Papenbrock, "TimeEval: A benchmarking toolkit for time series anomaly detection algorithms," *PVLDB*, vol. 15, no. 12, pp. 3678–3681, 2022.

[8] T. Palpanas, "Evolution of a Data Series Index," *CCIS*, vol. 1197, 2020.

[9] L. Zhang, N. Alghamdi, M. Y. Eltabakh, and E. A. Rundensteiner, "TARDIS: Distributed indexing framework for big time series data," in *ICDE*. IEEE, 2019, pp. 1202–1213.

[10] J. Wei, X. Lee, Z. Liao, T. Palpanas, and B. Peng, "Subspace collision: An efficient and accurate framework for high-dimensional approximate nearest neighbor search," *SIGMOD*, vol. 3, no. 1, pp. 1–29, 2025.

[11] J. Shieh and E. Keogh, "iSAX: indexing and mining terabyte sized time series," in *SIGKDD*. New York, USA: ACM, 2008, pp. 623–631.

[12] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos, "Fast subsequence matching in time-series databases," in *SIGMOD*. Minneapolis, Minnesota, USA: ACM, 1994, pp. 419–429.

[13] R. Agrawal, C. Faloutsos, and A. Swami, "Efficient similarity search in sequence databases," in *FODO*. Chicago, Illinois, USA: Springer, 1993, pp. 69–84.

[14] P. Schäfer and M. Högqvist, "SFA: a symbolic fourier approximation and index for similarity search in high dimensional datasets," in *EDBT*. Berlin, Germany: OpenProceedings, 2012, pp. 516–527.

[15] B. Peng, P. Fatourou, and T. Palpanas, "MESSI: In-memory data series indexing," in *ICDE*, Dallas, Texas, 2020, pp. 337–348.

[16] ——, "Fast data series indexing for in-memory data," *VLDB Journal*, vol. 30, no. 6, pp. 1041–1067, 2021.

[17] T. Rakthanmanon, B. Campana, A. Mueen, G. Batista, B. Westover, Q. Zhu, J. Zakaria, and E. Keogh, "Searching and mining trillions of time series subsequences under dynamic time warping," in *SIGKDD*. Beijing, China: ACM, 2012, pp. 262–270.

[18] J. Johnson, M. Douze, and H. Jégou, "Billion-scale similarity search with GPUs," *Big Data*, vol. 7, no. 3, pp. 535–547, 2019.

[19] E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra, "Dimensionality reduction for fast similarity search in large time series databases," *KIS*, vol. 3, pp. 263–286, 2001.

[20] A. Aggarwal and J. S. Vitter, "The input/output complexity of sorting and related problems," *Commun. ACM*, vol. 31, no. 9, pp. 1116–1127, 1988.

[21] R. Bellman and R. Kalaba, "On adaptive control processes," *Automatic Control, IRE Transactions on*, vol. 4, no. 2, pp. 1–9, 1959.

[22] S. Sprenger, P. Schäfer, and U. Leser, "Multidimensional range queries on modern hardware," in *SSDBM*. Bozen-Bolzano Italy: ACM, 2018, pp. 1–12.

[23] ——, "Bb-tree: A main-memory index structure for multidimensional range queries," in *ICDE*. IEEE, 2019, pp. 1566–1569.

[24] J. Lin, E. Keogh, S. Lonardi, and B. Chiu, "A symbolic representation of time series, with implications for streaming algorithms," in *SIGMOD workshops*. New York, USA: ACM, 2003, pp. 2–11.

[25] C. Lomont, "Introduction to intel advanced vector extensions," *Intel White Paper*, 2011.

[26] B. Tang, M. L. Yiu, Y. Li *et al.*, "Exploit Every Cycle: Vectorized Time Series Algorithms on Modern Commodity CPUs," in *IMDM*, 2016.

[27] B. Peng, P. Fatourou, and T. Palpanas, "Paris+: Data series indexing on multi-core architectures," *TKDE*, vol. 33, no. 5, pp. 2151–2164, 2020.

[28] "SOFA Supporting Webpage," https://github.com/patrickzib/SOFA, 2025.

[29] K. Chakrabarti, E. J. Keogh, S. Mehrotra, and M. J. Pazzani, "Locally adaptive dimensionality reduction for indexing large time series databases," *TODS*, vol. 27, no. 2, pp. 188–228, 2002.

[30] Q. Chen, L. Chen, X. Lian, Y. Liu, and J. X. Yu, "Indexable PLA for efficient similarity search," in *PVLDB*, Vienna, Austria, 2007, pp. 435–446.

[31] Y. Cai and R. Ng, "Indexing spatio-temporal trajectories with chebyshev polynomials," in *SIGMOD*. Paris, France: ACM, 2004, pp. 599–610.

[32] I. Popivanov and R. J. Miller, "Similarity search over time-series data using wavelets," in *ICDE*, IEEE. San Jose, CA, USA: IEEE, 2002, pp. 212–221.

[33] J. Shieh and E. Keogh, "iSAX: disk-aware mining and indexing of massive time series datasets," *Data Min. Knowl. Discov.*, vol. 19, pp. 24–57, 2009.

[34] A. Camerra, J. Shieh, T. Palpanas, T. Rakthanmanon, and E. Keogh, "Beyond one billion time series: indexing and mining very large time series collections with SAX2+," *KAIS*, vol. 39, no. 1, pp. 123–151, 2014.

[35] M. Linardi and T. Palpanas, "Scalable Data Series Subsequence Matching with ULISSE," *VLDB Journal*, 2020.

[36] K. Echihabi, P. Fatourou, K. Zoumpatianos, T. Palpanas, and H. Benbrahim, "Hercules against data series similarity search," *PVLDB*, vol. 15, no. 10, pp. 2005–2018, 2022.

[37] J. Wei, B. Peng, X. Lee, and T. Palpanas, "DET-LSH: A locality-sensitive hashing scheme with dynamic encoding tree for approximate nearest neighbor search," *PVLDB*, vol. 17, no. 9, pp. 2241–2254, 2024.

[38] Q. Wang, I. Ileana, and T. Palpanas, "LeaFi: Data Series Indexes on Steroids with Learned Filters," *SIGMOD*, vol. 3, no. 1, pp. 51:1–51:27, 2025.

[39] H. Yahyaoui and R. Al-Daihani, "A novel trend based sax reduction technique for time series," *Expert Systems with Applications*, vol. 130, pp. 113–123, 2019.

[40] Y. Yu, Y. Zhu, D. Wan, H. Liu, and Q. Zhao, "A novel symbolic aggregate approximation for time series," in *IMCOM*. Springer, 2019, pp. 805–822.

[41] H. Chen, J. Du, W. Zhang, and B. Li, "An iterative end point fitting based trend segmentation representation of time series and its distance measure," *Multimedia Tools and Applications*, vol. 79, pp. 13 481–13 499, 2020.

[42] L. Yan, X. Wu, and J. Xiao, "An improved time series symbolic representation based on multiple features and vector frequency difference," *Journal of Computer and Communications*, vol. 10, no. 6, pp. 44–62, 2022.

[43] L. Djebour, R. Akbarinia, and F. Masseglia, "Variable-size segmentation for time series representation," in *Transactions on Large-Scale Data-and Knowledge-Centered Systems LIII*. Springer, 2023, pp. 34–65.

[44] Y. Li and D. Shen, "A new symbolic representation method for time series," *Information Sciences*, vol. 609, pp. 276–303, 2022.

[45] P. Schäfer and U. Leser, "WEASEL 2.0: a random dilated dictionary transform for fast, accurate and memory constrained time series classification," *Machine Learning*, vol. 112, no. 12, pp. 4763–4788, 2023.

[46] P. Schäfer, "The BOSS is concerned with time series classification in the presence of noise," *Data Min. Knowl. Discov.*, vol. 29, pp. 1505–1530, 2015.

[47] Y. Sun, J. Li, J. Liu, B. Sun, and C. Chow, "An improvement of symbolic aggregate approximation distance measure for time series," *Neurocomputing*, vol. 138, pp. 189–198, 2014.

[48] H. Yin, S.-q. Yang, X.-q. Zhu, S.-d. Ma, and L.-m. Zhang, "Symbolic representation based on trend features for knowledge discovery in long time series," *Frontiers of Information Technology & Electronic Engineering*, vol. 16, pp. 744–758, 2015.

[49] S. Elsworth and S. Güttel, "ABBA: adaptive Brownian bridge-based symbolic aggregation of time series," *Data Min. Knowl. Discov.*, vol. 34, no. 4, pp. 1175–1200, 2020.

[50] A. Shifaz, C. Pelletier, F. Petitjean, and G. I. Webb, "Elastic similarity and distance measures for multivariate time series," *KIS*, vol. 65, no. 6, pp. 2665–2698, 2023.

[51] J. Shieh and E. Keogh, "iSAX: indexing and mining terabyte sized time series," in *SIGKDD*, 2008.

[52] N. S. Alghamdi, L. Zhang, E. A. Rundensteiner, and M. Y. Eltabakh, "Scalable time series compound infrastructure," in *SIGMOD*, 2022, pp. 1685–1698.

[53] X. Xie, H. Liu, W. Hou, and H. Huang, "A brief survey of vector databases," in *BigDIA*, IEEE. Budapest, Hungary: IEEE, 2023, pp. 364–371.

[54] Y. Tian, Z. Yue, R. Zhang, X. Zhao, B. Zheng, and X. Zhou, "Approximate Nearest Neighbor Search in High Dimensional Vector Databases: Current Research and Future Directions." *IEEE Data Eng. Bull.*, vol. 46, no. 3, pp. 39–54, 2023.

[55] S. Zhong and A. Mueen, "MASS: distance profile of a query over a time series," *Data Min. Knowl. Discov.*, pp. 1–27, 2024.

[56] K. Echihabi, K. Zoumpatianos, T. Palpanas, and H. Benbrahim, "Return of the Lernaean Hydra: Experimental Evaluation of Data Series Approximate Similarity Search," *PVLDB*, 2019.

[57] L. Zhang, M. Y. Eltabakh, E. A. Rundensteiner, and K. Alnuaim, "CLIMBER: Pivot-Based Approximate Similarity Search Over Big Data Series," in *ICDE*. IEEE, 2024, pp. 3933–3946.

[58] L. Zhang, N. Alghamdi, H. Zhang, M. Y. Eltabakh, and E. A. Rundensteiner, "PARROT: pattern-based correlation exploitation in big partitioned data series," *VLDB Journal*, vol. 32, no. 3, pp. 665–688, 2023.

[59] D. Rafiei and A. Mendelzon, "Efficient retrieval of similar time sequences using DFT," in *FODO*. Kobe, Japan: Springer, 1998, pp. 249–257.

[60] I. Coorporation, "Intel 64 and ia-32 architectures optimization reference manual," 2016.

[61] J. Woollam, J. Münchmeyer, F. Tilmann, A. Rietbrock, D. Lange, T. Bornstein, T. Diehl, C. Giunchi, F. Haslinger, D. Jozinović *et al.*, "SeisBench—A toolbox for machine learning in seismology," *Seismological Society of America*, vol. 93, no. 3, pp. 1695–1709, 2022.

[62] S. Soldi, V. Beckmann, W. H. Baumgartner, G. Ponti, C. R. Shrader, P. Lubiński, H. Krimm, F. Mattana, and J. Tueller, "Long-term variability of AGN at hard X-rays," *Astronomy & Astrophysics*, vol. 563, p. A57, 2014.

[63] H. V. Simhadri, G. Williams, M. Aumüller, M. Douze, A. Babenko, D. Baranchuk, Q. Chen, L. Hosseini, R. Krishnaswamny, G. Srinivasa *et al.*, "Results of the NeurIPS'21 challenge on billion-scale approximate nearest neighbor search," in *NeurIPS*, PMLR. Online: PMLR, 2022, pp. 177–189.

[64] A. Babenko and V. Lempitsky, "Efficient indexing of billion-scale datasets of deep descriptors," in *CVPR*. Las Vegas, NV, USA: IEEE, 2016, pp. 2055–2063.

[65] J. Woollam, A. Rietbrock, A. Bueno, and S. De Angelis, "Convolutional neural network for seismic phase classification, performance demonstration over a local seismic network," *Seismological Research Letters*, vol. 90, no. 2A, pp. 491–502, 2019.

[66] J. Münchmeyer, J. Saul, and F. Tilmann, "Learning the deep and the shallow: Deep-learning-based depth phase picking and earthquake depth estimation," *Seismological Research Letters*, vol. 95, no. 3, pp. 1543–1557, 2024.

[67] F. Magrini, D. Jozinović, F. Cammarano, A. Michelini, and L. Boschi, "Local earthquakes detection: A benchmark dataset of 3-component seismograms built on a global scale," *Artificial Intelligence in Geosciences*, vol. 1, pp. 1–10, 2020.

[68] W. L. Yeck, J. M. Patton, Z. E. Ross, G. P. Hayes, M. R. Guy, N. B. Ambruz, D. R. Shelly, H. M. Benz, and P. S. Earle, "Leveraging deep learning in global 24/7 real-time earthquake monitoring at the National Earthquake Information Center," *Seismological Society of America*, vol. 92, no. 1, pp. 469–480, 2021.

[69] T. Bornstein, D. Lange, J. Münchmeyer, J. Woollam, A. Rietbrock, G. Barcheck, I. Grevemeyer, and F. Tilmann, "PickBlue: Seismic phase picking for ocean bottom seismometers with deep learning," *Earth and Space Science*, vol. 11, no. 1, 2024.

[70] A. Niksejel and M. Zhang, "OBSTransformer: a deep-learning seismic phase picker for OBS data using automated labelling and transfer learning," *Geophysical Journal International*, vol. 237, no. 1, pp. 485–505, 2024.

[71] Y. Ni, A. Hutko, F. Skene, M. Denolle, S. Malone, P. Bodin, R. Hartog, and A. Wright, "Curated Pacific Northwest AI-ready Seismic Dataset," *Seismica*, vol. 2, no. 1, pp. –, May 2023.

[72] S. University, "Southwest university adult lifespan dataset (sald)," 2018.

[73] E. Center, "Southern California earthquake center (SCEDC)," *Caltech. Dataset*, vol. -, no. -, pp. –, 2013.

[74] H. Jégou, R. Tavenard, M. Douze, and L. Amsaleg, "Searching in one billion vectors: re-rank with source coding," in *ICASSP*, IEEE. Prague, Czech Republic: IEEE, 2011, pp. 861–864.

[75] S. M. Mousavi, Y. Sheng, W. Zhu, and G. C. Beroza, "STanford EArthquake Dataset (STEAD): A global data set of seismic signals for AI," *IEEE Access*, vol. 7, pp. 179 464–179 476, 2019.

[76] Y. Chen, A. Savvaidis, O. M. Saad, G.-C. Dino Huang, D. Siervo, V. O'Sullivan, C. McCabe, B. Uku, P. Fleck, G. Burke *et al.*, "TXED: The Texas earthquake dataset for AI," *Seismological Research Letters*, vol. 95, no. 3, pp. 2013–2022, 2024.

[77] C.-h. Chen, *Signal processing handbook*. CRC Press, 1988, vol. 51.

[78] M. Middlehurst, A. Ismail-Fawaz, A. Guillaume, C. Holder, D. Guijo-Rubio, G. Bulatova, L. Tsaprounis, L. Mentel, M. Walter, P. Schäfer *et al.*, "aeon: a Python toolkit for learning from time series," *JMLR*, vol. 25, no. 289, pp. 1–10, 2024.