GRAIL: Efficient Time-Series Representation Learning

John Paparrizos
University of Chicago
jopa@cs.uchicago.edu

Michael J. Franklin
University of Chicago
mjfranklin@cs.uchicago.edu

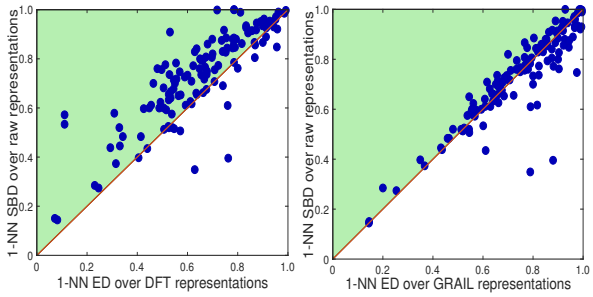
ABSTRACT

The analysis of time series is becoming increasingly prevalent across scientific disciplines and industrial applications. The effectiveness and the scalability of time-series mining techniques critically depend on design choices for three components responsible for (i) representing; (ii) comparing; and (iii) indexing time series. Unfortunately, these components have to date been investigated and developed independently, often resulting in mutually incompatible methods. The lack of a unified approach has hindered progress towards fast and accurate analytics over massive time-series collections. To address this major drawback, we present GRAIL, a generic framework to learn compact time-series representations that preserve the properties of a user-specified comparison function. Given the comparison function, GRAIL (i) extracts landmark time series using clustering; (ii) optimizes necessary parameters; and (iii) exploits approximations for kernel methods to construct representations in linear time and space by expressing each time series as a combination of the landmark time series. We extensively evaluate GRAIL for querying, classification, clustering, sampling, and visualization of time series. For these tasks, methods leveraging GRAIL's representations are significantly faster and at least as accurate as state-of-the-art methods operating over the raw time series. GRAIL shows promise as a new primitive for highly accurate, yet scalable, time-series analysis.

PVLDB Reference Format:
John Paparrizos and Michael J. Franklin. GRAIL: Efficient Time-Series Representation Learning. *PVLDB*, 12 (11): 1762-1777, 2019.
DOI: https://doi.org/10.14778/3342263.3342648

1. INTRODUCTION

Time series are often the outcome of sequentially recording time-varying measurements of natural processes (e.g., earthquakes and weather) or human-made artifacts (e.g., stock market and speech signals) [51, 38]. Recent technological advances permit the collection of enormous amounts of time-varying measurements [89, 77] across scientific applications (e.g., applications in astronomy [4] and neuroscience [16] involve millions of time series) and industrial settings (e.g., large-scale internet services handle billions of time-stamped measurements per day [74, 94, 6]). With the explosion of Internet of Things (IoT) applications, the rapid growth of time-series volumes is expected to continue [77]. Consequently, time-series analysis algorithms will have to operate over increasingly massive IoT collections.



(a) Raw vs. DFT (b) Raw vs. GRAIL

Figure 1: Comparison of classification accuracies across 128 datasets using time series in their raw representations against compact representations of size 20 computed with DFT and GRAIL. Circles over the diagonal indicate datasets over which raw representations outperform low-dimensional representations.

However, most state-of-the-art time-series mining methods cannot scale to millions of time series [70, 38, 9]. The temporal ordering and the high dimensionality of the observations, combined with the large time-series volumes, introduce severe challenges not shared with the analysis of other data types [127, 38]. Specifically, the temporal ordering may introduce distortions (see Section 2.3), which complicate the comparison of time series. In addition, the high dimensionality increases the computation and storage requirements of methods operating directly over time series. Finally, the large time-series volumes can make effective but non-scalable methods impractical in large-scale settings. To address these challenges, effective solutions require decisions for three core components [38]: (i) the *representation* method to construct low-dimensional representations that preserve time-series characteristics; (ii) the *comparison* method, which should offer invariances to distortions in time series; and (iii) the *indexing* method, which organizes time series to facilitate fast retrieval from massive collections.

Unfortunately, these components have to date been investigated and developed independently [38], often resulting in mutually incompatible methods. Therefore, existing time-series mining methods suffer from a number of drawbacks: (i) these methods become prohibitively expensive as they operate directly over raw time series to avoid incompatible representations (e.g., this is the case for most classification methods [9]); (ii) these methods sacrifice accuracy for efficiency as they directly exploit representations (e.g., this

Proceedings of the VLDB Endowment, Vol. 12, No. 11
ISSN 2150-8097.
DOI: https://doi.org/10.14778/3342263.3342648

is the case for online clustering methods that only support less-effective ℓ_p-norms [36]); or (iii) these methods follow a complicated, two-step approach to exploit representations to prune part of the pairwise comparisons (e.g., this is the case for querying methods that exploit representations incompatible with a similarity function [130, 62, 104, 61, 99]).

The two-step approach is the most prominent paradigm to accelerate time-series mining methods. The requirement to perform two steps is due to the dependence on the seminal GEMINI framework [2, 39], which laid the foundations for fast time-series retrieval under Euclidean distance (ED). Specifically, GEMINI (i) constructs low-dimensional representations; and (ii) defines a measure over such representations to *lower bound* (i.e., prune part of) the ED comparisons in the high-dimensional space. For different distances, two-step approaches [130, 62, 104, 61] additionally show that ED lower bounds the new distance. This is a challenging task for which solutions for specific comparison methods resulted in award-winning research [39, 58, 19, 25, 99]. However, the plethora of representation and comparison methods [38, 119, 9, 24], along with the difficulty in developing lower bounding measures, impact the sustainability of such methodology.

The lack of a unified approach has hindered progress towards fast and accurate analytics for time series. Therefore, we need a new primitive to learn time-series representations that builds on and extends the principles of GEMINI. Specifically, given a comparison function, the learned representations should: ($\mathcal{P}1$) preserve the pairwise similarities and serve as feature vectors for machine learning methods; ($\mathcal{P}2$) lower bound the comparison function to accelerate similarity search; ($\mathcal{P}3$) allow using prefixes of the representations (by ranking their coordinates in descending order of importance) for scaling methods under limited resources; ($\mathcal{P}4$) support efficient and memory-tractable computation for new data to enable operations in online settings; and ($\mathcal{P}5$) support efficient and memory-tractable eigendecomposition of the data-to-data similarity matrix to exploit highly effective methods that rely on such cornerstone operation.

In this paper, we develop a primitive to satisfy all aforementioned principles. Specifically, we present a Generic RepresentAtIon Learning (GRAIL) framework to automatically learn compact representations that preserve the properties of a user-specified similarity function. This is fundamentally different from the time-series literature where representation methods are agnostic to the similarity function used in subsequent steps [38]. To illustrate the implications of that point, in Figure 1 we compare the one-nearest-neighbor (1-NN) classification accuracies across 128 datasets [32] using the Shape-Based Distance (SBD) [91, 92] over raw time series against ED over two representations of size 20 computed by (i) Discrete Fourier Transform (DFT), a state-of-the-art representation method [119, 105]; and (ii) our GRAIL representations, which preserve a distance similar to SBD that we discuss later. SBD over raw time series significantly outperforms ED over DFT representations. In contrast, ED over GRAIL representations is at least as accurate as SBD over raw time series and significantly faster.

To learn highly accurate representations, GRAIL exploits kernel methods that unify data modeling and algorithm design [26, 53, 108, 109, 7]. GRAIL requires to perform two steps to learn representations in linear time and space: (i) approximate the sequence-to-sequence (SS) similarity matrix; and (ii) approximate the eigendecomposition of SS. To approximate SS and facilitate easy adaptation of kernel functions (i.e., similarity measures), GRAIL relies on the Nyström method [86, 123], which is agnostic to the choice of the kernel function. Specifically, GRAIL first extracts a dictionary of time-series sequences and constructs the sequence-to-dictionary (SD) and the dictionary-to-dictionary (DD) similarity matrices. Then, Nyström uses these matrices to construct representations by expressing each time series as a linear combination of the time series in the dictionary.

Unfortunately, the quality of Nyström's representations critically depend on an accurate estimation of necessary parameters. In addition, the dimensionality of Nyström's representations might surpass the dimensionality of the original time series, which is undesirable. To circumvent these limitations, we propose a lightweight methodology for unsupervised parameter estimation. In addition, we propose to leverage Nyström's representations to perform an additional approximation of the eigendecomposition of SS in order to learn the final representations of reduced dimensionality.

Considering the importance of using shift-invariant comparison methods for time-series analysis (see Section 2.3), we describe GRAIL by showing how it can support this property. To alleviate the burdens associated with the high memory and runtime costs of computing large similarity matrices (e.g., SD), we first show how we can compute a Shift-INvariant Kernel (SINK) by decomposing the original time series into their frequency components and by operating over the first few frequencies that well approximate the original time series. To compute landmark time series that summarize available data, we study the effectiveness of using time-series clustering methods, such as k-Shape [91], for dictionary learning. Finally, we build the end-to-end solution on top of Apache Spark [131] to facilitate representation learning and time-series analytics over massive collections.

To demonstrate the effectiveness of SINK and GRAIL, we perform an extensive evaluation on 128 datasets and compare their performance against state-of-the-art methods for five tasks, namely, (i) querying; (ii) classification; (iii) clustering; (iv) sampling; and (v) visualization. We use public datasets and make our source code available. In summary, we show that kernel classifiers using SINK are as powerful in terms of accuracy as an ensemble of eleven 1-NN classifiers with state-of-the-art distance measures. GRAIL representations are more accurate than current representations and achieve better pruning power than existing lower bounding measures. Importantly, for all five tasks, methods leveraging GRAIL's representations are significantly faster and at least as accurate as state-of-the-art methods operating over the high-dimensional time series. Finally, we present a case study on millions of time series representing energy usage patterns to emphasize the scalability of our ideas.

We start with a review of the relevant background (see Section 2). We provide an overview of our approach (Section 3.1) and then present our contributions as follows:

- We show how SINK, a shift-invariant kernel function, can operate in a principled manner over the first few low frequencies of time series (Section 3.2).
- We study the effectiviness of time-series clustering for learning dictionaries for Nyström (Section 3.3).
- We develop a solution to estimate necessary parameters and the compactness of representations (Section 3.4).
- We present GRAIL, our end-to-end solution for learning compact time-series representations (Section 3.5).
- We build GRAIL on top of Apache Spark to facilitate large-scale time-series analytics (Section 4).
- We evaluate our ideas by conducting an extensive experimental evaluation for five tasks (Sections 5 and 6).

Finally, we conclude with a discussion of related work (Section 7) and the implications of our work (Section 8).

2. BACKGROUND AND PRELIMINARIES

In this section, we review linear and nonlinear dimensionality reduction methods (Section 2.1). Then, we summarize time-series representation methods (Section 2.2) and common time-series distortions and distance measures (Section 2.3). Finally, we present our problem of focus (Section 2.4).

2.1 Dimensionality Reduction Methods

Dimensionality reduction is the process of mapping high-dimensional vectors into a low-dimensional space while preserving some property of interest [29, 49]. Depending on the mapping, we divide dimensionality reduction methods into *linear* and *nonlinear* methods [67]. To cover concepts necessary for understanding the proposed ideas, we first review linear methods and, subsequently, nonlinear methods.

Linear dimensionality reduction: Consider n real-valued m-dimensional vectors $X = [\vec{x}_1, \ldots, \vec{x}_n]^\top \in \mathbb{R}^{n \times m}$ and a target dimensionality $k < m$. The goal is to produce low-dimensional vectors $Z_k = [\vec{z}_1, \ldots, \vec{z}_n]^\top \in \mathbb{R}^{n \times k}$ such that pairwise similarities, expressed as inner products, in the low-dimensional space closely approximate pairwise similarities in the high-dimensional space. This is the problem that the cornerstone method Singular Value Decomposition (SVD) [93, 54, 116, 27] solves optimally [49, 29]. To see how SVD produces Z_k, we first express the SVD of X as follows:

$$X = U \Sigma V^\top \tag{1}$$

where the matrix $U = [\vec{u}_1, \ldots, \vec{u}_n] \in \mathbb{R}^{n \times n}$ contains the left singular vectors, the matrix $V = [\vec{v}_1, \ldots, \vec{v}_m] \in \mathbb{R}^{m \times m}$ contains the right singular vectors, and the matrix $\Sigma = diag(\sigma_1, \ldots, \sigma_m) \in \mathbb{R}^{n \times m}$ contains the singular values in descending order along its diagonal (i.e., $\sigma_1 \geq \ldots \geq \sigma_m$), with zeroes everywhere else. By retaining the first $k < m$ singular values and vectors, we obtain Z_k as follows:

$$Z_k = U_{1:n,1:k} \Sigma_{1:k,1:k} = X V_{1:m,1:k} \tag{2}$$

SVD requires $\mathcal{O}(min\{nm^2 + m^3, mn^2 + n^3\})$ time but, unfortunately, SVD fails to effectively model data with nonlinear structures. In contrast, kernel methods [26, 53, 107, 108, 109] play a central role in the analysis of real-world data by enabling operations in a high-dimensional feature space.

Nonlinear dimensionality reduction: Specifically, kernel methods use a function $\phi : \mathbb{R}^m \rightarrow \mathcal{H}$ to implicitly map data into a Reproducing Kernel Hilbert Space (RKHS) $\mathcal{H}$ of high (and often infinite) dimension. Because explicit computation of coordinates in $\mathcal{H}$ is infeasible, kernel methods invoke the "kernel trick" [3], which permits interaction with data using a kernel function, $k(\vec{x}, \vec{y}) = \langle \phi(\vec{x}), \phi(\vec{y}) \rangle_{\mathcal{H}}$, or its corresponding distance metric, $D_k(\vec{x}, \vec{y}) = k(\vec{x}, \vec{x}) + k(\vec{y}, \vec{y}) - 2k(\vec{x}, \vec{y})$ [106]. To perform dimensionality reduction in $\mathcal{H}$, Kernel Principal Component Analysis (KPCA) [107, 108] performs eigendecomposition over the Gram matrix K, with $K_{ij} = k(\vec{x}_i, \vec{x}_j) = \phi(\vec{x}_i)^\top \phi(\vec{x}_j) \ \forall \ i, j \in \{1, \ldots, n\}$. KPCA requires $\mathcal{O}(n^2)$ space to store K, $\mathcal{O}(n^2)$ time to construct K, and $\mathcal{O}(n^3)$ time to perform an eigendecomposition on K.

Approximations for kernel methods: To alleviate the burdens associated with the high memory and runtime costs of kernel methods, two seminal methods have been proposed to approximate K: the Nyström method [86, 123, 37] and the Random Fourier Features (RFF) method [98]. Specifically, Nyström is a data-aware method that computes landmark vectors from available data and requires $\mathcal{O}(nd^2)$ time to construct Z_d, where d is the number of landmark vectors. In contrast, RFF is a data-agnostic method that samples feature maps from an independent distribution and requires $\mathcal{O}(nmd)$ time to construct Z_d, where d is the number of feature maps. For both methods, the benefits from approximating $\widehat{K} = Z_d Z_d^\top$ are twofold: (i) significant memory savings as Z_d requires only $\mathcal{O}(nd)$ space; and (ii) substantial runtime savings. However, considering the difficulty to generalize RFF on arbitrary kernel functions as well as the impressive theoretical and empirical benefits for Nyström in comparison to RFF [128], in our approach we rely on Nyström to efficiently learn representations for time series.

2.2 Time-Series Representation Methods

Despite their optimality [49, 29], exact linear and nonlinear dimensionality reduction methods are prohibitively expensive in practice. Therefore, representation methods rely on spectral decompositions [49, 29, 95, 56, 88] to reduce the high dimensionality of time series and lower the storage and computational costs of time-series analysis methods. Since the debut of the GEMINI framework [2, 39], research on representation methods has focused on exploring trade-offs between low-dimensional representations, such as reconstruction quality, sensitivity to noise, compactness, and computational cost. Depending on the transformation applied to time series and on the output format, we divide representation methods into *data-agnostic* and *data-aware* methods and into *numeric* and *symbolic* methods [38].

Data-agnostic methods: The GEMINI framework represented time series as a set of sinusoidal coefficients using the DFT [2, 39]. Subsequently, many methods were proposed to replace DFT, including the Discrete Cosine Transform (DCT) [63], the Discrete Wavelet Transform (DWT) [22], Daubechies and Haar wavelets [96, 21], Coiflets [111], and Chebychev polynomials [19]. More specific to time series, the Piecewise Aggregate Approximation (PAA) [129, 57] represents time series as mean values of segments.

Data-aware methods: In contrast to data-agnostic methods, data-aware methods tune transformation parameters on available data to improve their effectiveness. For example, data-aware methods relying on spectral decompositions select a subset of DFT [117] or DWT [115] coefficients. A data-aware version of PAA uses vector quantization to construct a codebook of segments [78, 79], whereas other approaches, namely Piecewise Linear Approximation (PLA) [112] and Adaptive Piecewise Constant Approximation (APCA) [58], fit a polynomial model or use a constant approximation for each segment, respectively. SVD is also inherently a data-aware method proposed to represent time series [63, 101].

Symbolic methods: The output of all previous methods is numeric. Symbolic methods additionally quantize the numeric output. For example, the Symbolic Aggregate approXimation (SAX) [72] and the Symbolic Fourier Approximation (SFA) [105] rely on alphabets to transform PAA and DFT representations, respectively, into short words.

The previously described methods construct representations to lower bound or approximate ED [135, 20]. Unfortunately, recent experimental evaluations of distance measures [91, 92, 35, 119, 9] show that ED is less effective for most applications. Next, we review alternative distance measures.

2.3 Invariances and Distance Measures

Distance measures handle the majority of distortions (e.g., noise and outliers) by preprocessing time series before comparison [59, 60, 5, 88, 48]. However, for many important distortions, preprocessing is not effective and, therefore, sophisticated distance measures offer invariances during comparison. For example, to satisfy the *shift invariance*, SBD [91, 92] compares out-of-phase time series, whereas DTW [103] compares time series with misaligned regions.

Shape-based Distance: SBD, a parameter-free distance measure, compares time series in $\mathcal{O}(m \log m)$ runtime cost. Let $\mathcal{F}(\vec{x})$ and $\mathcal{F}^{-1}(\vec{x})$ denote the DFT and the inverse DFT of $\vec{x}$ [56, 88], respectively, we compute SBD as follows:

$$SBD(\vec{x}, \vec{y}) = 1 - \max\left(\frac{\mathcal{F}^{-1}\{\mathcal{F}(\vec{x}) * \mathcal{F}(\vec{y})\}}{||\vec{x}|| \cdot ||\vec{y}||}\right) \quad (3)$$

where $*$ is the complex conjugate in the frequency domain.

Dynamic Time Warping: DTW first constructs a matrix $M \in \mathbb{R}^{m \times m}$ containing in each cell the ED of any two coordinates of $\vec{x}$ and $\vec{y}$. Then, DTW computes in $\mathcal{O}(m^2)$ time a warping path in M, a contiguous set of matrix elements $L = \{l_1, l_2, \ldots, l_r\}$, with $r \geq m$, with the minimum warping cost among the exponential number of possible paths:

$$DTW(\vec{x}, \vec{y}) = \min \sqrt{\sum_{i=1}^{r} l_i} \quad (4)$$

Due to the high computation cost of DTW, constrained Dynamic Time Warping (cDTW) visits only a subset of the cells on M by defining the *band*, the shape of the subset matrix M, and the *warping window*, the width of the band.

2.4 Problem Definition

We address the problem of efficiently learning compact representations that preserve the invariances offered by a given kernel function, $k(\vec{x}, \vec{y})$, for any $\vec{x}$ and $\vec{y}$ in the original high-dimensional space. We require the learned representations to satisfy the following principles: ($\mathcal{P}1$) preserve similarities (i.e., $\langle Z_k(\vec{x}), Z_k(\vec{y})\rangle \approx k(\vec{x}, \vec{y})$); ($\mathcal{P}2$) offer lower bounding measure (i.e., $ED(Z_k(\vec{x}), Z_k(\vec{y})) \leq D_k(\vec{x}, \vec{y})$); ($\mathcal{P}3$) permit operations over prefixes of coordinates (i.e., $Z_q = [Z_k(1), \ldots, Z_k(q)]$, with $q < k$); ($\mathcal{P}4$) support efficient and memory-tractable computation of new data; and ($\mathcal{P}5$) support efficient and memory-tractable eigendecomposition of K.

3. THE GRAIL FRAMEWORK

Our objective is to simplify and unify the design of time-series mining methods by enabling them to operate directly over representations. To achieve this goal, we present GRAIL, a framework to facilitate efficient representations learning based on a user-specified comparison method. We start by providing an overview of GRAIL and its core components.

3.1 Overview

A promising direction to construct representations that natively preserve the invariances offered by a chosen comparison method arises with the use of kernel methods [26, 53, 108, 109, 7]. Unfortunately, exact methods for representation learning with kernels, such as KPCA [107], are prohibitively expensive in practice as they require to operate over the full Gram matrix, which consists of pairwise similarities computed with a chosen kernel function. A number of approaches have been proposed to approximate the kernel matrix and reduce the high memory and runtime cost (see Section 2.1). For GRAIL, we rely on Nyström, which is agnostic to the choice of kernel function. Nyström approximates $k(\vec{x}_i, \cdot) = \phi(\vec{x}_i)\ \forall\ i \in \{1, \ldots, n\}$ as a linear combination of d vectors $G = [\vec{g}_1, \ldots, \vec{g}_d]^\top \in \mathbb{R}^{d \times m}$ [87]:

$$\arg\min_{a \in \mathbb{R}^{d \times n}} \sum_{i=1}^{n} ||K(\vec{x}_i, \cdot) - \sum_{j=1}^{d} a_{j,i} K(\vec{g}_j, \cdot)||_{\mathcal{H}}^2 \quad (5)$$

For each $\vec{x}_i$, this is a convex problem depending on a single column of a and the optimal solution is $a = W^{-1}E^T$, where $W \in \mathbb{R}^{d \times d}$ is the DD matrix, with $W_{ij} = k(\vec{g}_i, \vec{g}_j)\,\forall\, i, j \in G$, $E \in \mathbb{R}^{n \times d}$ is the SD matrix, with $E_{ij} = k(\vec{x}_i, \vec{g}_j)\,\forall\, i \in X,\ j \in G$, and $W^{-1} = Q_W \Lambda_W^{-1} Q_W^{-1}$ is the inverse of W. Nyström constructs $\widehat{K} \approx K$ as follows:

$$\widehat{K} = EW^{-1}E^\top = (EQ_W\Lambda_W^{-0.5})(EQ_W\Lambda_W^{-0.5})^\top = Z_d Z_d^\top \quad (6)$$

Unfortunately, Nyström suffers from two main drawbacks. First, the quality of the learned representations, Z_d, critically depend on an accurate estimation of necessary parameters, a step that is often omitted in the literature or the assumption is taken that parameters are tuned in a supervised manner. Second, the dimensionality of the learned representation might surpass the dimensionality of the original time series (i.e., $d \gg m$), which defeats the original purpose of producing compact representations.

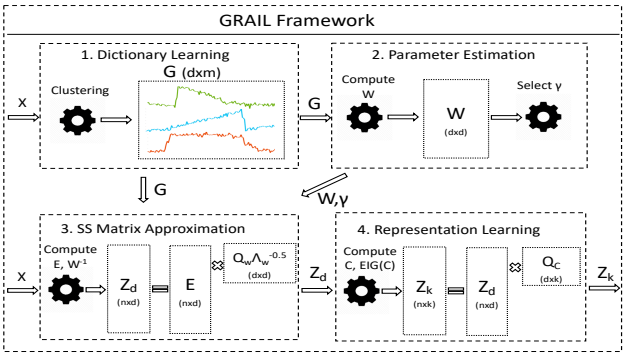


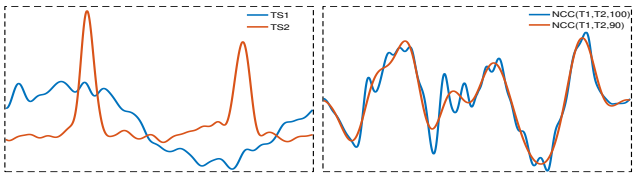
Figure 2: Overview of GRAIL.

To circumvent these limitations, we present an end-to-end solution for learning time-series representations, which introduces two novelties. Specifically, we propose (i) a methodology to estimate parameters with the objective to maximize both the variance of the data in the low-dimensional space and the compactness of the learned representations; and (ii) an additional approximation of the eigendocomposition of K using Z_d to learn the final representation, which permit us to satisfy principles $\mathcal{P}1$ through $\mathcal{P}5$ in Section 2.4. Figure 2 provides an overview of GRAIL's components that we discuss in detail next. We start by showing how we can approximate in a principled manner SINK, a shift-invariant kernel function, using only the first few Fourier coefficients of the original time series (Section 3.2). Given a kernel function, GRAIL proceeds in three steps. First, GRAIL extracts landmark time series using clustering (Section 3.3). Second, GRAIL estimates necessary parameters (Section 3.4). Third, given the landmark time series and the estimated parameters, GRAIL performs two approximations for kernel methods to learn representations (Section 3.5).

3.2 Shift-Invariant Kernel for Time Series

Comparing time-series under the shift invariance is of critical importance for effective time-series analysis. GRAIL interacts with time series using kernel functions that satisfy the positive semi-definiteness property [28, 109], which we discuss next. Unfortunately, successful methods for time-series comparison under these invariances, such as SBD and DTW (see Section 2.3), cannot construct, based on their current formulation, proper kernel functions [118, 30]. State-of-the-art kernel functions for time series, such as the Global Alignment (GA) kernel [31, 30], scale quadratically with the time-series length and are therefore prohibitively expensive to use in practice. Our novel SINK function is intended to circumvent this efficiency limitation.

The positive semi-definiteness (p.s.d.) property of kernel functions (often referred to as Mercer's condition [81]) constitutes a critical condition for the existence of RKHS (see



(a) Time series $T1$ and $T2$ (b) NCC sequences

Figure 3: Examples of NCC sequences produced by SINK.

Section 2.1) and leads to convex solutions for many learning problems involving kernels [26, 7]. In simple terms, a symmetric function $k(\cdot,\cdot)$ is a p.s.d. kernel if its Gram matrix K has non-negative eigenvalues. Unfortunately, for many functions it is challenging to show that they satisfy Mercer's condition. For example, it required multiple attempts to define a proper DTW-like kernel function, such as GA [113, 11, 50, 31, 30]. Haussler's convolution kernel [52] is a seminal framework for engineering new p.s.d. kernels [114].

The central idea behind convolution kernels is to infer the similarity of two data objects based on the similarity of their parts. Specifically, convolution kernels combine p.s.d. kernels (i.e., return the sum of the values of the kernels) computed on all pairs of the components of the composite data objects. More formally, we assume that a time series $\vec{x} \in \mathcal{X}$ is composed of P parts $x_p \in \mathcal{X}_p$. Let us also assume that a p.s.d. kernel $k' : \mathcal{X}_p \times \mathcal{X}_p \to \mathbb{R}$ exists and that a relation $R(\vec{x})$ denotes the possible ways in which we can transform $\vec{x}$ into $x_1, \ldots, x_P$. Then, we can define a kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ as follows [52, 53, 114]:

$$k(\vec{x},\vec{y}) = \sum_{x' \in R(\vec{x})} \sum_{y' \in R(\vec{y})} k'(x',y') \tag{7}$$

Therefore, we need to define a relation $R_s(\vec{x})$ to decompose time series in such a way to be shift-invariant. We adapt the kernel from [118] and we introduce critical normalizations.

Shift-invariant kernel: $R_s(\vec{x})$ needs to decompose time series $\vec{x}$ into all the shifted versions of $\vec{x}$. By using a circular shift operator over $\vec{x}$ as our relation, we can rearrange the coordinates of $\vec{x} = (x_1, \ldots, x_m)$ as follows: $R_s(\vec{x}, S) = (x_{(1+S) \mod m}, \ldots, x_{(m+S) \mod m})$. This process closely relates to how SBD (see Section 2.3) finds the shift that maximizes the inner product between $\vec{y}$ and $R_s(\vec{x}, S), \forall S \in [1, m]$. Instead of finding the best shift, [118] proposed to weight higher (lower) the inner products in shifted positions with large (small) values by setting $k'(\vec{x},\vec{y},\gamma) = e^{\gamma\langle\vec{x},\vec{y}\rangle}$, with bandwidth $\gamma > 0$, in Equation 7. To eliminate numerical issues that arise due to the exponentiation of long time series, we use the same normalization as for SBD:

$$k_s(\vec{x},\vec{y},\gamma) = \sum e^{\gamma NCC(\vec{x},\vec{y})} \tag{8}$$

where $NCC(\vec{x},\vec{y}) = \frac{\mathcal{F}^{-1}\{\mathcal{F}(\vec{x}) * \mathcal{F}(\vec{y})\}}{||\vec{x}|| \cdot ||\vec{y}||}$ is the normalized cross-correlation sequence. This formulation might result in a Gram matrix with off-diagonal values higher than values on the diagonal (i.e., a time series may be more similar to another time series than to itself!), which contradicts how methods often operate over similarity matrices. Therefore, to solve this issue, we define SINK as follows:

$$k'_s(\vec{x},\vec{y},\gamma) = \frac{k_s(\vec{x},\vec{y},\gamma)}{\sqrt{k_s(\vec{x},\vec{x},\gamma) \cdot k_s(\vec{y},\vec{y},\gamma)}} \tag{9}$$

For every pairwise comparison, SINK transfers the time series in their Fourier domain using the Fast Fourier Transform (FFT), where the cross-correlation operation can be efficiently computed, and return back to the original space using the inverse FFT. For GRAIL, we need to employ SINK

Algorithm 1: Shift-Invariant Kernel (SINK)

Input : $\vec{x}$ is a 1-by-m z-normalized time series
$\vec{y}$ is a 1-by-m z-normalized time series
γ is a scaling parameter
e denotes the preserved energy in Fourier domain
Output: Sim is the kernel value between $\vec{x}$ and $\vec{y}$

1 **function** $Sim = SINK(\vec{x},\vec{y},\gamma,e)$**:**
2 $\quad Sim = \frac{SumNCC(\vec{x},\vec{y},\gamma,e)}{sqrt(SumNCC(\vec{x},\vec{x},\gamma,e) \cdot SumNCC(\vec{y},\vec{y},\gamma,e))}$
3 **def** $cc = SumNCC(\vec{x},\vec{y},\gamma,e)$**:**
4 $\quad cc = sum(exp(\gamma \cdot NCC(\vec{x},\vec{y},e)))$
5 **def** $cc = NCC(\vec{x},\vec{y},e)$**:**
6 $\quad FFTx = PreservedEnergy(\vec{x},e)$
7 $\quad FFTy = PreservedEnergy(\vec{y},e)$
8 $\quad cc = IFFT(FFTx * FFTy)$
9 $\quad cc = \frac{cc}{norm(\vec{x}) \cdot norm(\vec{y})}$
10 **def** $FFTx = PreservedEnergy(\vec{x},e)$**:**
11 $\quad FFTx = FFT(\vec{x}, 2^{nextpower2(2 \cdot len(\vec{x})-1)})$
12 $\quad NormCumSum = \frac{cumsum(abs(FFTx).^2)}{sum(abs(FFTx).^2)}$
13 $\quad k = find(NormCumSum \geq \frac{e}{2})$
14 $\quad FFTx((k+1) : (endindex - k - 1)) = 0$

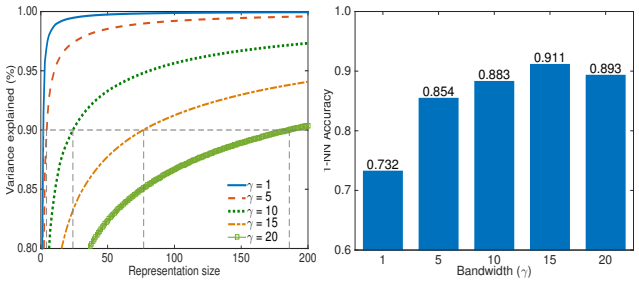
for the computation of the SD and DD matrices. In both matrices, the set of time series in the dictionary remains static and, therefore, we explore two additional optimizations that lead to significant speedup. Specifically, we transfer the time-series in the dictionary in their Fourier domain only once and reuse them in every pairwise computation to avoid unnecessary FFT calls. In addition, considering that most natural time series exhibit a skewed energy spectrum [39], which implies that most of the energy consentrates in the first few low frequencies, it suffices to retain only the first few Fourier coefficients of the time-series that retain some user-specified level of energy (e.g., 90%). Algorithm 1 shows how SINK compares two time series in a few lines of code using modern mathematical software. Specifically, from line 10 to 14, SINK computes number of Fourier coefficients required to preserve the user-specified energy level e. SINK requires $\mathcal{O}(m \log m)$ time due to its dependence on FFT. For fixed-size inputs, which is the focus of this work, SINK satisfies the p.s.d. property as it combines p.s.d. kernels computed for each possible alignment between two time series. However, for time series of variable lengths it requires deeper analysis, which we leave for future work. In Figure 3, we present an example of how SINK constructs NCC sequences with and without energy preservation (Figure 3b) for two time series (Figure 3a). We observe that with 90% of preserved energy, which reduces the size of time series from 1024 to only 21, SINK accurately approximates exact NCC.

3.3 Time-Series Dictionary Learning

Given SINK, GRAIL requires to compute a dictionary of landmark time series to summarize the underlying data. Such selection of time series for Nyström is a difficult combinatorial problem. A large body of work focuses on empirical and theoretical analysis of sampling and projection methods for Nyström [123, 37, 65, 47]. Unfortunately, existing approaches have not been developed with time series in mind. Therefore, we study the value of using the centroids of time-series clustering methods to construct landmark time series. The approximation error for Nyström is bounded by the quantization error of mapping each vector $\vec{x}_i \in X$ to the closest of the d landmark vectors $\vec{g}_i \in G$ [133]:

$$||K - EW^{-1}E^T||_F \leq \sum_{i=1}^{n} ||\vec{x}_i - g_{map(i)}||^2 \tag{10}$$

where $map(i) = \arg\min_{j=1,\ldots,d} ||\vec{x}_i - \vec{g}_j||^2$. Clustering methods relying on the principles of k-means [76] minimize the



(a) Cumulative variance (b) Classification accuracy

Figure 4: Comparison of the compactness and the classification accuracy of learned representations with different γ values for SINK for the StarLightCurve dataset.

exact same quantization error. Unfortunately, k-means is not suitable for time-series clustering. Instead, k-Shape efficiently clusters time series and the cluster centroids effectively summarize time series for distance measures offering shift invariance [91, 92]. Therefore, for GRAIL, we use the cluster centroids of k-Shape as dictionary of landmark time series. k-Shape requires $\mathcal{O}(\max\{n \cdot d \cdot m \cdot \log(m), n \cdot m^2, d \cdot m^3\})$ time, where d is the number of centroids requested. Our evaluation (see Sections 5 and 6) shows that k-Shape significantly outperforms all state-of-the-art methods for selecting landmark time series, including random sampling.

Next, we discuss how we can rely on the landmark time series to estimate necessary parameters for GRAIL.

3.4 Parameter Estimation

After the construction of landmark time series, GRAIL proceeds with the estimation of kernel function parameters, which affect (i) the compactness; and (ii) the quality of the learned representations. To illustrate this point, Figure 4 explores how different γ values of SINK affect those two factors for the StarLightCurve dataset [32]. Specifically, Figure 4a measures the cumulative variance explained in varying representation sizes for different γ values. Figure 4b presents the 1-NN classification accuracies of representations capturing 90% of the variance in time series for different γ values. We observe that small γ values require only a few eigenvectors to explain 90% of the variation in the similarities of time series, whereas large γ values require significantly more eigenvectors (i.e., small γ values lead to more compact representations). On the other hand, when all representations explain the same targeted amount of variance in data, different γ values lead to significant differences in the quality of the representations for various tasks, such as for classification in Figure 4b. Therefore, selecting small γ values due to the storage and computation benefits might lead to significant reductions in accuracy. Due to this difficulty in tuning kernel function parameters, the literature often relies on supervised tuning, which is not always realistic.

For GRAIL, we propose a simple, yet effective, method to estimate the bandwidth γ of SINK that relies on two key observations. First, considering that some loss of information is unavoidable in the low-dimensional space, time-series similarities with large variance are much more likely to be preserved in the low-dimensional space than time-series similarities with low variance. Therefore, we want to select γ values such that the variance is maximized. Second, to determine how effectively we can capture the variance in the low-dimensional space, we need to consider how much of the variance each eigenvalue explains. Therefore, we want to select γ values such that for a small number of r eigenvalues, such as $r = 20$, the variance explained is maximized. Our method combines those two observations in a scoring function to permit effective selection for γ. Specifically, considering the eigendecomposition of matrix $W = Q_W \Lambda_W Q_W^T$, where W (i.e., the DD matrix) consists of the pairwise kernel values of the landmark time series (see Section 3.3), we compute the scoring function of each γ value as follows:

$$Score(\gamma) = var(W, \gamma) \cdot \frac{\sum_{i=1}^{r} \Lambda_W(i)}{\sum_{j=1}^{d} \Lambda_W(j)} \quad (11)$$

where $var(W, \gamma)$ denotes the variance of the kernel values in W computed with SINK using γ. Algorithm 2 shows how we can tune γ for SINK according to our scoring function (Equation 11). The cost of Algorithm 2 is dominated by the need to compute the eigendecomposition of W, which requires $\mathcal{O}(d^3)$ time. In our evaluation (Sections 5 and 6), we show that our method for parameter tuning leads to representations with similar compactness and accuracy to representations tuned in a supervised manner. Next, we show how GRAIL learns time-series representations.

Algorithm 2: Parameter Estimation

Input : G is a d-by-m matrix containing d landmark sequences
GV is a 1-by-l vector containing l values for γ
e denotes the preserved energy in Fourier domain
Output: *score* is the score of selected γ
gamma is the selected kernel scaling parameter γ

function $[score, gamma] = GammaSel(G, GV, e)$:
  foreach $\gamma \in GV$ **do**
    for $i = 1$ **to** d **do**
      for $j = 1$ **to** d **do**
        $W(i,j) = SINK(G(i,:), G(j,:), \gamma, e)$
    $GVar(\gamma) = var(W)$
    $[Q, L] = eig(W)$
    $VarTop20(\gamma) = \frac{cumsum(L(1:20))}{sum(L)}$
  $[score, gamma] = max(GVar \circ VarTop20)$

3.5 Time-Series Representation Learning

GRAIL proceeds in two steps: (i) constructs a temporal representation, Z_d, where d is the number of landmark time series, to approximate the Gram matrix $\widehat{K} = Z_d Z_d^\top$ using the Nyström method (Section 3.1); and (ii) uses Z_d to approximate the eigendecomposition of $\widehat{K}$ and learn the final representation Z_k, where $k \leq d$, which satisfies principles $\mathcal{P}1$ through $\mathcal{P}5$ in Section 2.4.

Approximation of K: Nyström requires as input two matrices: (i) matrix $W \in \mathbb{R}^{d \times d}$, which contains all pairwise similarities between the d landmark time series; and (ii) matrix $E \in \mathbb{R}^{n \times d}$, which contains pairwise similarities between the n time series and the d landmark time series. Once we construct those two matrices, Nyström computes the inverse of W, $W^{-1} = Q_W \Lambda_W^{-1} Q_W^{-1}$ and constructs Z_d as follows:

$$Z_d = E Q_W \Lambda_W^{-0.5} \quad (12)$$

The dimensionality d of Z_d corresponds to the number of landmark time series extracted from the entire dataset and, therefore, d is significantly smaller than the size n of the entire dataset. Unfortunately, for very large datasets, Nyström might require thousands of landmark time series to accurately approximate K. In such cases, the dimensionality of the learned representation might surpass the dimensionality of the original time series, which defeats the original purpose of producing low-dimensional representations.

Representation Learning: To eliminate this issue, we approximate the eigendecomposition of $\widehat{K}$ using Z_d and, subsequently, reduce the dimensionality from d to k by retaining

Algorithm 3: Representation Learning with GRAIL

Input : X is a n-by-m matrix containing n time series
d is the number of landmark time series to extract
f is a scalar to tune the dimensionality k of Z_k
GV is a 1-by-l vector containing l values for γ
e denotes the preserved energy in Fourier domain

Output: Z_k is a n-by-k matrix of the representations

function $Z_k = GRAIL(X, k, GV, m, f)$**:**
  $G = k - Shape(X, d)$
  $[score, gamma] = GammaSel(G, GV, e)$
  for $i = 1$ **to** d **do**
    for $j = 1$ **to** d **do**
      $W(i,j) = SINK(G(i,:), G(j,:), gamma, e)$
  for $i = 1$ **to** n **do**
    for $j = 1$ **to** d **do**
      $E(i,j) = SINK(X(i,:), G(j,:), gamma, e)$
  $[Q, L] = eig(W)$
  $Z_d = E \cdot Q \cdot L^{-0.5}$
  $b = 0.5 \cdot d$
  $B = zeros(b, d)$
  for $i = 1$ **to** n **do**
    $B = [B; Z_d(i,:)]$
    if *B has no zero valued rows* **then**
      $[U, S, V] = SVD(B)$
      $B = sqrt(max(0, S^2 - S^2_{\frac{b}{2},\frac{b}{2}} \cdot I_b)) \cdot V^T$
  $[Q, L] = eig(B^T \cdot B)$
  $k = find(\frac{cumsum(L)}{sum(L)} > f)$
  $Z_k = Z_d \cdot Q(1:d, 1:k)$

only the top eigenvalues and eigenvectors of K. An alternative approach is to directly exploit the eigendecomposition of matrix W, which Nyström computes to construct Z_d, to approximate the eigendecomposition. Unfortunately, this approach offers a poor approximation of the eigendecomposition of $\widehat{K}$. Importantly, it may lead to non-orthogonalized eigenvectors of $\widehat{K}$ [42], as required by definition (see Section 2.1), which impacts the effectiveness of the learned representation. Therefore, assuming the eigendecomposition of $K = Q\Lambda Q^\top$, we can approximate $\widehat{Q} = Z_d Q_C \Lambda_C^{-0.5}$ in $\mathcal{O}(nd^2)$, where $Q_C \in \mathbb{R}^{d\times d}$ and $\Lambda_C \in \mathbb{R}^{d\times d}$ are the eigenvectors and eigenvalues of $C = Z_d^\top Z_d \in \mathbb{R}^{d\times d}$. Unfortunately, for a large number d of landmark time series, this computation becomes prohibitively expensive. We employ a matrix sketching algorithm over Z_d, namely, the Frequent Directions (FD) method [71], to efficiently approximate matrix C by retaining only a sketch of size b of Z_d in $\mathcal{O}(ndb)$, where $b \leq d \leq n$, and we construct $Z_k = Z_d Q_{C\,1:d,1:k}$. This procedure leads to a number of benefits: (i) guarantees that $\widehat{Q}$ remains orthonormal and permits lower bounding of the kernel function ($\mathcal{P}2$); (ii) permits the estimation of the variance explained in each coordinate of Z_k and, therefore, we can use prefixes of the coordinates of Z_k ($\mathcal{P}3$); and (iii) enables efficient approximation of the eigenvalues and eigenvectors of $\widehat{K}$ and, therefore, seamless integration with existing algorithms relying on such key operation ($\mathcal{P}5$). Overall, the costs of Nyström and k-Shape dominate the runtime behavior of GRAIL, $\mathcal{O}(\max\{n\cdot d\cdot m\cdot\log(m),\ n\cdot m^2,\ d\cdot m^3, n\cdot d^2\})$, which, importantly, is linear to the size n of the dataset. The approximation guarantees known for the Nyström method (or its variants) depend on the spectrum of the kernel matrix, which are entirely data-dependent. We refer the reader to [82] for recent results. As for SINK, providing guarantees for settings with variable-length inputs requires deeper analysis, which we leave for future work. Next, we review the implementation of GRAIL on top of Apache Spark.

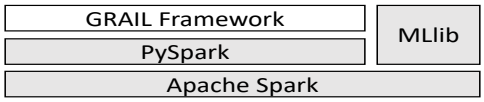


Figure 5: GRAIL is a thin layer on top of Apache Spark.

4. GRAIL OVER APACHE SPARK

We build GRAIL as a thin layer on top of Apache Spark [131], a widely used distributed dataflow system, to faciliate processing of large-scale IoT datasets. We utilize Spark's MLlib library [80] to access existing distributed implementations of popular data mining and machine learning methods. Figure 5 depicts the GRAIL layer. GRAIL first uses k-shape to compute a dictionary of landmark time series. We initialize k-Shape by assigning all time series in $\mathcal{X}$ to random clusters, from which we extract centroids. Each iteration of k-shape includes two operations: (i) updating the cluster membership of each time series in $\mathcal{X}$ with SBD; and (ii) refining cluster centroids according to updated membership. To parallelize the first step, we distribute $\mathcal{X}$ evenly across all Spark executors. Each executor then uses SBD on its small chunk of time series against the centroids and find the closest centroid for each timeseries $\vec{x_i}$. To parallelize the second step, we need to keep the aligned sequence $\vec{x_i}'$ of $\vec{x_i}$ towards its closest centroid, which is already computed in (i). We then obtain a distributed dataset (stored in Spark's RDD) of aligned sequences and utilize distributed matrix multiplication to compute the covariance matrix required during centroid computation. Finally, we perform eigendecomposition using FD to compute the refined centroid of each cluster (each executor performs FD on its chunk).

Then, GRAIL performs the parameter tuning process on the computed centroids (i.e., the dictionary) for various γ values. For large dictionary sizes, we parallelize the computation of each γ by distributing the eigendecomposition of matrix W to spark executors. For small dictionary sizes, because computation for each γ is fast, we let each executor compute different γ and report its score. For the next step, GRAIL constructs representations Z_d based on acquired centroids and tuned parameters. We parallelize the calculation of the E matrix by distributing $\mathcal{X}$ across executors and let each executor compute the pairwise similarities between its small chunk of $\mathcal{X}$ and the d landmark time series. Next, we utilize distributed matrix multiplication to compute $Z_d = EQ_W\Lambda_W^{-0.5}$. Note Q_W and Λ_W are already computed during parameter tuning.

Finally, GRAIL uses FD to approximate the eigendecomposition of the Gram matrix $\widehat{K}$ and reduce the dimensionality of data (i.e., learn Z_k). We parallelize FD by distributing Z_d to executors, letting each executor compute the SVD in batch, and we aggregate the results to obtain a sketch of size b for Z_d. We use the sketch to compute C locally, and then distribute C and compute SVD in parallel to obtain Q_C and Λ_C. For the last step, we use again distributed matrix multiplication to construct the representation Z_k.

We now turn to the experimental evaluation of GRAIL.

5. EXPERIMENTAL SETTINGS

In this section, we review the settings for the evaluation of (i) our kernel function, SINK; (ii) k-Shape as a dictionary learning algorithm; (iii) our parameter selection approach; (iv) GRAIL representations; and (v) five time-series mining methods that exploit GRAIL representations.

Datasets: We use the 128 datasets from the UCR archive [32], the largest collection of class-labeled time-series datasets.

Datasets span different domains, are z-normalized, and split into training and test sets. Each dataset contains from 40 to $24,000$ sequences. We employ standardized resampling and interpolation methods to fix issues with varying lengths and missing values [90] that were deliberately left to reflect the real world. The maximum sequence length is $2,844$. Every sequence in each dataset belongs to only one class.

Platform: We ran our experiments on a cluster of 300 servers with an identical configuration: Dual Intel Xeon E5-2650v4 (12-core with 2-way SMT) processor with clock speed at 2.2 GHz and up to 128 GB RAM. Each server ran Red Hat Linux 6.6 (64-bit) and Matlab R2018a (64-bit).

Implementation: We implemented all methods in Matlab for a consistent evaluation. We also built GRAIL on top of Apache Spark (using PySpark) for large-scale analysis. For repeatability purposes, we make our source code available.[1]

Baselines: We compare SINK against the following state-of-the-art distance measures and kernel functions for time series: (i) **SBD**, an efficient, accurate, and parameter-free distance measure [91]; (ii) **cDTW**, the constrained version of DTW, with improved accuracy and efficiency [103]; and (iii) **GA**, a state-of-the-art global alignment kernel [30].

Following [35, 119], we use the 1-NN classifier, which is a simple and parameter-free classifier, to evaluate distance measures. Importantly, 1-NN classifiers, combined with elastic distance measures, such as cDTW, achieve state-of-the-art performance on time-series classification [126, 119, 9]. For kernel functions, we use the Support Vector Machine (SVM) classifier [26] as implemented by [23] to evaluate SINK (**SVM+SINK**) and GA (**SVM+GA**) kernels. Additionally, we compare these approaches to the recently proposed Elastic Ensemble (**EE**) classifier that combines 11 1-NN classifiers using several elastic distance measures [73] and the **COTE** classifier that combines 35 classifiers [10].

We compare k-Shape against state-of-the-art sampling and projection methods used to compute landmark vectors for Nyström. Specifically, we consider the following sampling methods: (i) **Random**, a simple and efficient uniform sampling method [123]; (ii) **AFKMC2**, an approximate version of the k-means++ sampling method [8]; (iii) **GibbsDPP**, an approximate version of the Determinantal Point Processes (DPP) sampling method [64, 1] that relies on Gibbs sampling [69]; and (iv) **LevScore**, a non-uniform sampling method based on leverage scores [47]. In addition, we consider two projection methods: (i) a projection method based on Fourier transform (**SRFT**) [47]; and (ii) a projection method based on Gaussian Processes (**GP**) [47].

We evaluate **GRAIL-PT**, our parameter tuning method, against three approaches for kernel function parameter estimation: (i) **MinVariance**, a simple method to select kernel function parameters such that the variance in data is minimized; (ii) **MaxVariance**, a simple method to select kernel function parameters such that the variance in data is maximized; and (iii) **LOOCAcc**, a supervised method to always select kernel function parameters such that the leave-one-out classification accuracy is maximized.

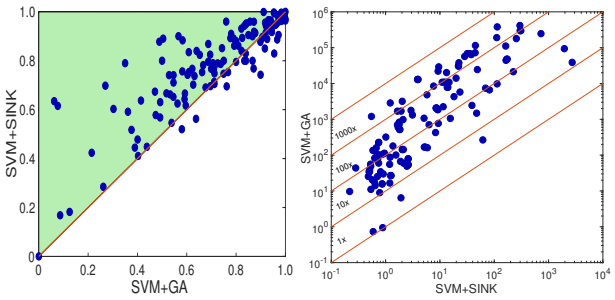
To understand the performance of GRAIL representations in practice, we evaluate learned representations combined with suitable kernel methods, in five major time-series mining tasks. In addition, we perform an extensive experimentation against different approaches proposed in the literature to learn representations for time series. Specifically, we compute representations using the Shift-invariant Dictionary Learning (**SIDL**) method [134], the Similarity Preserving Representation Learning method (**SPIRAL**) [68], the Random Warping Series (**RWS**) [125], and the Encoder-ADAPT deep learning method (**EncoderA**) [110], which shares a very similar architecture to the Fully Convolutional Neural Network (**FCN**) [121].

For *querying*, we compare **GRAIL-LB**, our lower bound for SINK, against two state-of-the-art lower bounds for ED and cDTW distance measures: (i) **DFT-LB:** a method that uses Fourier transform to represent time series and uses the first-k Fourier coefficients to lower bound ED [39]; and (ii) **Keogh-LB,** the state-of-the-art lower bound for cDTW [61]. The GRAIL-LB and DFT-LB methods operate over low-dimensional representations, whereas Keogh-LB operates over the original raw time-series representation. For *classification*, we evaluate **GRAIL-SVM**, our classifier that exploits linear SVM [40] against the same classifiers that we considered above to evaluate SINK. In addition, we compare **GRAIL-SVM** against linear SVM classifiers trained over SIDL (**SIDL-SVM**), SPIRAL (**SPIRAL-SVM**), and RWS representations (**RWS-SVM**). For **EncoderA**, we use the training and testing procedures as suggested in the original paper [110]. For *clustering*, we compare **GRAIL-SC**, a spectral clustering method [85] running over GRAIL representations against two state-of-the-art clustering methods [91, 92]: (i) **k-AVG+ED**, the original, highly efficient, but less accurate k-means clustering method [76]; and (ii) **k-Shape:** an efficient and highly accurate time-series clustering method [91]. In addition, we compare **GRAIL-SC** against k-means methods running over SIDL (**SIDL-KM**), SPIRAL (**SPIRAL-KM**), and RWS (**RWS-KM**) representations. For *sampling*, we compare **GRAIL-DPP**, a DPP sampling method [64, 1] running over GRAIL representations against two state-of-the-art methods for approximate sampling using the k-means++ mechanism, namely **AFKMC2**, and the DPP method, namely **GibbsDPP**, that we discussed earlier. Finally, for *visualization*, we compare the approximation error (see below) between our representations (**GRAIL+Rep**) when used to approximate the exact KPCA method (**KPCA+Rep**) to visualize time series in two dimensions. In both approaches, Rep denotes the representation used (e.g., GRAIL+Z95 denotes the GRAIL representation, Z_k, that explains 95% of the variance in Z_d).

Parameter settings: Among the distance measures discussed above, cDTW requires setting a parameter to constrain its warping window. We compute the optimal window by performing a leave-one-out classification step over the training set of each dataset (**cDTW^{opt}**). To evaluate SINK and GA kernels using SVM, we need to set a regularization parameter $\mathcal{C}$. We tune the $\mathcal{C}$ value in the training set of each dataset using a grid search with power of two $\mathcal{C}$ values ranging from -10 to 20 with step 0.11. For both kernels, we consider scaling parameters ranging from 1 to 20. We denote SINK^w the version of SINK that operates over a reduced number of Fourier coefficients computed to preserve w% of the signal's energy. For dictionary learning, all methods select or construct the same number of landmark time series for each dataset, which corresponds to 40% of the number of time series in each dataset (capped to 100 for large datasets). For the evaluation of the learned representations, we extract landmark time series using k-Shape and GRAIL-PT to estimate parameters for SINK.

For querying, the GRAIL-LB and DFT-LB methods operate over low-dimensional representations with a fixed number of coordinates, 10, across each dataset. Keogh-LB operates over raw time-series using cDTW^5, the cDTW with window 5% of the length of the time series of each dataset.

[1] https://github.com/johnpaparrizos/GRAIL



(a) Classification accuracy (b) Runtime comparison

Figure 6: Comparison of SVM+SINK and SVM+GA classifiers over 128 datasets.

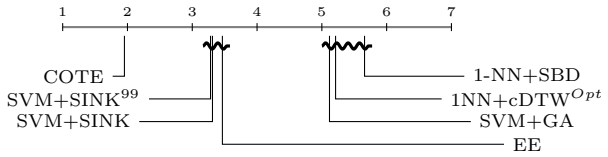
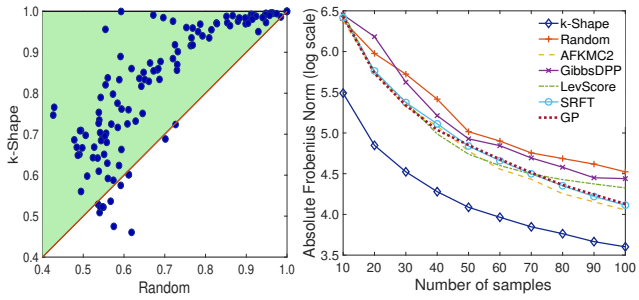


Figure 7: Ranking of classifiers based on the average of their ranks across datasets. The wiggly line connects methods that do not perform statistically differently.



(a) k-Shape against Random (b) Approximation error on SLC

Figure 8: Comparison of dictionary learning algorithms.

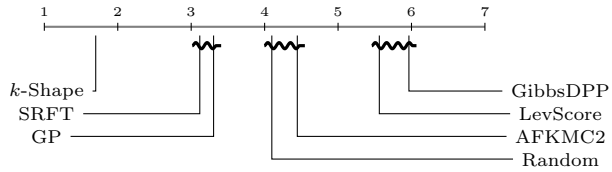


Figure 9: Ranking of dictionary learning algorithms based on the average of their ranks across datasets.

For classification, we set the regularization parameter $\mathcal{C}$ of GRAIL-SVM using the same values for grid search as before and exploit GRAIL representations constructed using d landmark time series. We use the same training procedure for SIDL-SVM, SPIRAL-SVM, and RWS-SVM. We tune additional parameters these methods require following the recommended values from their corresponding papers. The only difference is that we force the learned representations to have the same size as ours for a fair comparison. For clustering and sampling, we use GRAIL representations that explain $f = 95\%$ of the variance and we use the number of classes in each dataset as the number of clusters and number of samples, respectively.

Metrics: We compare our approaches on runtime and accuracy. For runtime, we compute CPU time utilization and measure the time ratios for our comparisons for each dataset. To evaluate classifiers, we report the classification accuracy (i.e., number of correctly classified instances over all instances) by performing classification over the training and test sets of each dataset. To evaluate representations we report the approximation error using the Frobenius norm between the original kernel matrix K and the approximated kernel $\widehat{K} = Z_k Z_k^\top$. To visualize our results over the 128 datasets, we employ a min-max normalization to the approximation error to bound it between 0 and 1, and report 1 minus the error as an accuracy value. We use the Rand Index (RI) [100] to evaluate clustering accuracy over the fused training and test sets of each dataset. For clustering and sampling, we report the average RI over 10 runs; in every run we use a different random initialization.

Statistical analysis: Following [33, 91, 9], we use the Wilcoxon test [122] with a 99% confidence level, a test that is less affected by outliers than is the t-test [102], to evaluate pairs of algorithms over multiple datasets. We also use the Friedman test [43] followed by the post-hoc Nemenyi test [84] with 95% confidence level for comparison of multiple algorithms over multiple datasets.

6. EXPERIMENTAL RESULTS

In this section, we report our experimental results. We aim to: (1) evaluate SINK against state-of-the-art distance and kernel functions (Section 6.1); (2) compare k-Shape against dictionary learning methods (Section 6.2); (3) evaluate our parameter tuning method (Section 6.3); and (4) evaluate GRAIL representations on five tasks: (i) querying; (ii) classification; (iii) clustering; (iv) sampling; and (v) visualization (Section 6.4). Finally, we present our case study (Section 6.5) and highlight our findings (Section 6.6).

6.1 Evaluation of Distance Measures

We evaluate the accuracy of SVM classifiers combined with the SINK and GA kernels. Figure 6a presents the pairwise differences in accuracy over 128 datasets. SVM+SINK outperforms SVM+GA in the vast majority of the datasets (i.e., most circles, with each circle representing a dataset, are above the diagonal) and Wilcoxon suggests this difference in accuracy is statistically significant. SVM+SINK also significantly outperforms SVM+GA in terms of efficiency. Specifically, Figure 6b shows that SVM+SINK is significantly faster than SVM+GA[2] across all datasets we considered with differences ranging between one order to three orders of magnitude. Subsequently, we evaluate the performance of all classifiers together. Figure 7 shows the average rank across 85 out of 128 datasets (because accuracy values for COTE and EE are only available for this subset). The wiggly line represents no statistically significant differences between the rankings according to Friedman followed by the Nemenyi test. We observe three clusters of methods: COTE forms the first cluster, SVM+SINK, SVM+SINK^{99}, and EE form the second cluster, and 1-NN+SBD, 1-NN+cDTW^{Opt}, and SVM+GA form the third cluster. COTE outperforms the methods in the second cluster and, in turn, the methods in the second cluster are significantly better than methods in the third cluster. We observe no statistically significant differences between SVM+SINK and SVM+SINK^{99}, which indicates no accuracy loss despite SINK^{99} operating only over the first few Fourier coefficients. The number of coefficients is determined for each dataset by retaining 99% of the energy of the time series. On average, this step leads to 67% reduction in the size of required coefficients, which, in turn, results on a 4.2x speedup of SVM+SINK^{99} against SVM+SINK.

[2]We used the C/Mex Matlab code provided by the author.

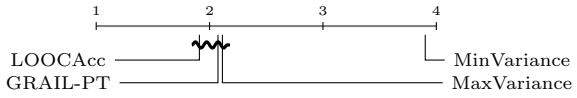


Figure 10: Ranking of parameter tuning methods based on the average of their ranks, using accuracy as measure, across datasets.

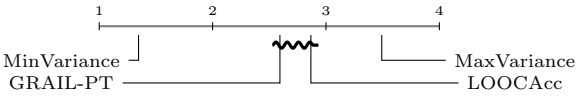


Figure 11: Ranking of parameter tuning methods based on the average of their ranks, using length as measure, across datasets.

We tested different energy levels for SINK and we observed gradual reduction in accuracy as we compress more and more the original time series (e.g., SINK_{90} becomes significantly less accurate than SINK). Additionally, we observe variants of SINK, when combined with appropriate kernel methods, to perform as accurate as EE, an ensemble of 11 classifiers. This is remarkably high performance for a standalone similarity function, which implies, as we will show, that learning representations that preserve SINK, achieves state-of-the-art performance on all five tasks we consider in our analysis despite operating over low-dimensional (compressed) representations. In contrast, variants of SINK are significantly worse than COTE, a state-of-the-art classifier. However, we note that COTE is an ensemble of 35 classifiers. Considering how powerful SVM+SINK is, we believe a new version of COTE that includes SINK and other kernel functions omitted previously (e.g., GA) will result to new state-of-the-art classification performance.

6.2 Evaluation of Dictionaries

Having shown the robustness of SINK, we now evaluate the performance of k-Shape as a dictionary learning algorithm. First, we evaluate k-Shape against Random, the simplest, yet effective, method to sample time series. Figure 8a compares the two methods using their approximation error (transformed into accuracy as discussed in Section 5). k-Shape outperforms Random in most datasets and the Wilcoxon test suggests that this difference in accuracy is statistically significant. To ensure that the performance of k-Shape is not an artifact of choosing k landmark time series as the number of classes per datasets, we perform an additional experiment where we vary the number of k landmark time series from 10 to 100. Figure 8b presents the approximation error (transformed into accuracy as discussed in Section 5) on the StarLightCurve (SLC) dataset (chosen randomly). We observe that k-Shape outperforms all methods across all different values of k and that Wilcoxon suggests that all differences in accuracy are statistically significant. We observe similar behaviors across datasets.

To verify the superiority of k-Shape against the other methods, we evaluate the significance of their differences in accuracy when considered all together. Figure 9 shows the average rank across datasets for each method. k-Shape is the top method, meaning that k-Shape performed best in the majority of the datasets. We observe three clusters of methods whose ranks do not present a statistically significant difference: SRFT and GP, the two projection methods, form the first cluster; AFKMC2 and Random, two sampling methods, form the second cluster; and GibbsDPP and LevScore, two sampling methods, form the third cluster. The methods in the first cluster are significantly better than the methods in the second cluster and, in turn, the methods in the second cluster are significantly better than the methods in the third cluster. Therefore, we can conclude that projection methods outperform sampling methods for the dictionary learning task and, importantly, k-Shape is the only method that significantly outperforms all methods.

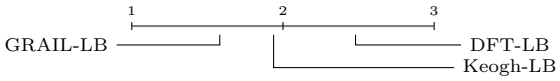


Figure 12: Ranking of lower bounding methods based on the average of their ranks, using the pruning power as measure, across datasets.

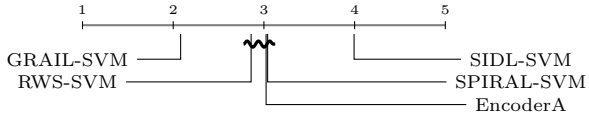


Figure 13: Ranking of representation methods for the classification task based on the average of their ranks, using accuracy as measure, across datasets.

6.3 Evaluation of Parameter Estimation

We now turn our focus to our parameter selection method. Figure 10 shows the average rank for each method using their classification accuracy as measure. LOOCAcc, a supervised method to select parameters using the training set of each dataset, is ranked first, as expected, meaning that LOOCAcc performed best in the majority of the datasets. Interestingly, we observe that GRAIL-PT, our unsupervised method to tune parameters, and MaxVariance achieve similar classification accuracy to LOOCAcc. According to the Friedman test followed by a post-hoc Nemenyi test to evaluate the significance of the differences in the ranks, only MinVariance achieves a significant reduction in terms of accuracy in comparison to the other methods.

Figure 11 presents the average rank across datasets for each method using as measure the dimensionality (i.e., the length) of the learned representations. We observe that MinVariance is ranked first meaning that MinVariance produces the most compact representations in comparison to all other methods. However, as we have seen previously, MinVariance also leads to a significant loss in terms of classification accuracy. In contrast, we observe that GRAIL-PT produces representations as compact as those from LOOCAcc, the supervised method to tune parameters. Importantly, GRAIL-PT significantly outperforms MaxVariance in terms of the dimensionality of the learned representations, despite the similar performance of GRAIL-PT and MaxVariance in terms of classification accuracy. Therefore, we can conclude that GRAIL-PT is the only unsupervised method that produces accurate and compact representations, similar to those produced in a supervised manner by LOOCAcc. In contrast, MaxVariance and MinVariance produce either very high-dimensional representations or very low-dimensional representations, respectively. Very low-dimensional representations are desirable. Unfortunately, MinVariance is not competitive in terms of accuracy when selecting such low-dimensional representations.

6.4 Evaluation of GRAIL on Five Tasks

Having shown the robustness of all critical components of GRAIL, we now focus our evaluation on the performance of the learned representation for five time-series mining tasks.

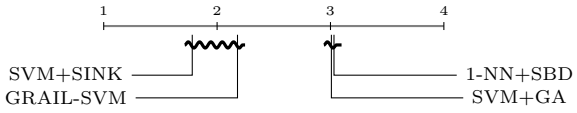


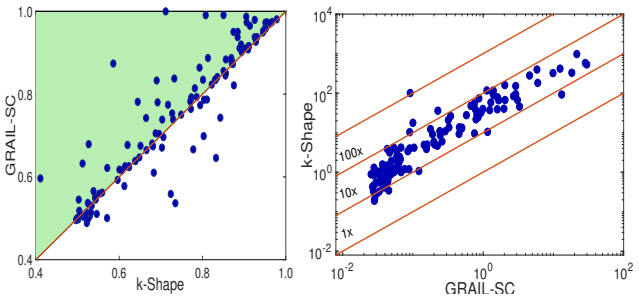
Figure 14: Ranking of classification methods based on the average of their ranks, using accuracy as measure, across datasets.

Querying: We evaluate GRAIL-LB, our lower bound for SINK against state-of-the-art lower bounds for ED and cDTW. We use as measure the pruning power (i.e., the number of comparisons the methods avoid from all possible pairwise comparisons). Figure 12, shows that GRAIL-LB significantly outperforms Keogh-LB, a method to lower bound DTW as well as the state-of-the-art representation method for ED, namely DFT-LB. This is a critical achievement as, to the best of our knowledge, this is the first time that the construction of lower bounding measures is automated and, importantly, leads to signficantly better results than existing hand-crafted solutions.

Classification: Figure 13 compares the performance of GRAIL representations against state-of-the-art representation learning methods for the classification task. We observe that RWS-SVM, SPIRAL-RWS, and EncoderA methods perform similarly while SIDL-SVM is significantly worse than all other methods. Interestingly, GRAIL-SVM outperforms significantly all methods. To understand this result, in Figure 14 we show that GRAIL-SVM, our SVM classifier over GRAIL representations achieves very similar performance to SVM+SINK (as GRAIL-SVM essentially approximates the accuracy of SVM+SINK), indicating the robustness of GRAIL representations for this task. Importantly, GRAIL-SVM outperforms significantly SVM+GA and 1-NN+SBD methods, showing that despite operating over reduced dimensionality, GRAIL-SVM performs better than methods operating over the original, high-dimensional, time series. In contrast, RWS-SVM, which is inspired by the GA kernel, performs significantly worse than SVM+SINK. SPIRAL-SVM is a parameter-free method and therefore the benefit of using SVMs is limited. SIDL-SVM relies on shift-invariant properties similar to out method, however, the tuned parameters did not reveal any significant improvements in performance. Finally, for the EncoderA method, we report worse results than what appears in the original paper (i.e., EncoderA performs similarly to COTE) because we enforce the representations of EncoderA to have limited size (similar to ours).

Clustering: We evaluate GRAIL-SC, our spectral clustering algorithm against k-Shape and k-AVG+ED. Figure 15a shows that GRAIL-SC, despite operating over a low-dimensional representation, performs similarly to k-Shape, a highly accurate and efficient time-series clustering method. Importantly, due to the reduced dimensionality of time series, GRAIL-SC leads to a significantly faster algorithm to cluster time series than the k-Shape algorithm (Figure 15b). GRAIL-SC is the only method that significantly outperforms k-AVG+ED and achieves similar accuracy performance to k-Shape. All other methods, namely, SPIRAL-KM, RWS-KM, and SIDL-KM, perform similar to or worse than k-AVG+ED (Figure 16). SPIRAL-KM and RWS-KM achieve similar results to k-AVG+ED, however, SIDL-KM, performs significantly worse.

Sampling: We evaluate GRAIL-DPP, our DPP-based sampling method against two state-of-the-art methods for approximate sampling using the k-means++ mechanism and the DPP method. Figure 17 considers all methods together. GRAIL-DPP outperforms both methods. AFKMC2 and GibbsDPP show no significant differences.



(a) Rand Index accuracy (b) Runtime

Figure 15: Comparison of clustering algorithms.

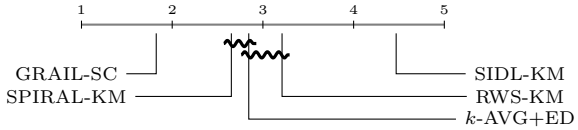


Figure 16: Ranking of clustering methods based on the average of their ranks across datasets.

Visualization: We evaluate the performance of GRAIL representations against representations produced using the exact KPCA for the purpose of visualization. Figure 18 presents the average rank across datasets for two KPCA-Z85, KPCA-Z90, GRAIL-Z90, and GRAIL-Z95. We observe that KPCA-Z90 is ranked first, meaning that KPCA-Z90 had the best approximation error in the majority of the datasets. GRAIL-Z95 is ranked second, followed by KPCA-Z85, but the difference in their approximation error is not statistically significant. Therefore, we observe that GRAIL representations must explain larger percentages of the variance in data to achieve visualization performance similar to that of exact representations, an expected result considering that GRAIL is an approximate method.

6.5 Case Study on 10M Time Series

We now demonstrate the scalability of GRAIL using Spark. Through a collaboration with a prominent energy provider at Illinois, we obtained access to TBs of energy smart meter data that contain the energy usage of clients for a period of two years. We created a dataset of ten million time series (clients' data across different weeks) with length of 1000 energy measurements. We performed two experiments to demonstrate the scalability of GRAIL across different dimensions using machines from our cluster (see Section 5). For both experiments we report the runtime for learning GRAIL representations and performing k-means clustering over the learned representations to group clients together based on their energy usage patterns. In Figure 19a, we utilize 100 cores (5 machines) to show how our framework scales with increasing dataset sizes (from 10K to 10M time series) and dictionary sizes of 100, 500, and 1000. We observe that our framework scale linearly with increasing data size and that the size of dictionary doesn't effect the scalability. In Figure 19b, we keep the dataset fixed but changed the length of time series into consideration. We observed that our framework scales linearly when more cores are available and the length does not effect the scalability.

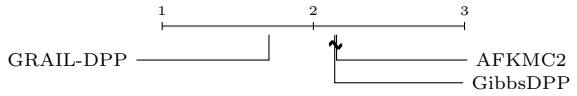


Figure 17: Ranking of sampling methods based on the average of their ranks, using the approximation error as measure.

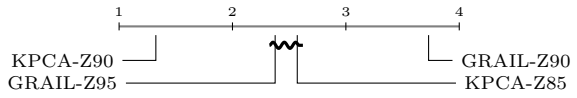


Figure 18: Ranking of representation methods based on the average of their ranks, using the approximation error as measure.

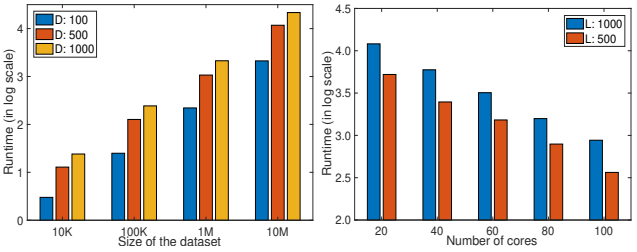
6.6 Summary of Experimental Evaluation

In short, our experimental evaluation suggests that: (1) kernel methods combined with suitable kernel functions, such as SINK, significantly outperform state-of-the-art distance measures combined with 1-NN classifiers; (2) cluster centroids, such as those computed using k-Shape, can effectively serve as dictionaries of landmark time series for representation learning tasks; (3) unsupervised tuning of kernel function parameters leads to accurate and compact time-series representations; (4) GRAIL learns time-series representations that are more compact and more accurate than state-of-the-art representations; (5) GRAIL representations achieve excellent pruning of time-series comparisons; (6) GRAIL representations, combined with suitable methods achieve high accurate and significantly improves the runtime of algorithms while operating over low-dimensional representations; (7) GRAIL is suitable for large-scale time-series analytics as it scales linearly across all its parameters with increasingly larger datasets.

7. RELATED WORK

We focused on efficient representation learning from time series. Beyond unsupervised approaches, traditional model-based approaches assume a model, which is often expressed in the form of analytical equations with parameters, to describe time series and use the estimated parameters of such model as features in time-series mining tasks [66]. There is a plethora of model-based approaches in the literature [55, 17, 44] that rely on different models to serve different application needs, such as Hidden Markov Models [97], Gaussian Process models [18], and autoregressive models [75]. Unfortunately, unrealistic assumptions of such models combined with their limited power to model highly complex and high-dimensional time series with analytical equations, impact the effectiveness of model-based approaches as standalone feature extraction methods for real-world problems [66].

As a consequence, there has been significant effort in time-series literature to extract generic features to represent time series using combinations of statistical measures that summarize different time-series properties, including their distribution, correlation structure, stationarity, entropy, and fitting to a range of different time-series models [83, 120, 34, 46, 45]. Despite the effectiveness of such methods for classification and forecasting tasks, where the supervised selection of features reduces the dimensionality of feature vectors and increases accuracy, these approaches are not competitive for unsupervised tasks. Similarly, many time-series classification methods involve operations for feature extraction. We refer the reader to [9] for a great survey and evaluation of such approaches. Unfortunately, for unsupervised



(a) Varying dictionary size (b) Varying length

Figure 19: Runtime results for GRAIL over Spark.

settings, such methods are not competitive, as it was shown for shapelet-based clustering methods [92].

Alternative approaches to learn feature representations rely on deep learning methods [132, 15, 14, 13, 41, 12]. An advantage of these methods is that they can learn multiple layers of feature representations. A recent deep learning method, namely Encoder [110], has achieved similar accuracy performance to the state-of-the-art COTE classifier [10]. Unfortunately, when we limit the size of the representations (to match GRAIL's size) this method is no longer competitive. There are many different architectures for learning representations with neural networks. We refer the reader to [66] for a thorough review. Recently, a number of approaches have been proposed to learn representations for time series that are related to our method. SIDL [134] attempts to capture informative local patterns in different locations of time series and relies on sparse coding to learn representations. SPIRAL [68], similarly to our method, learns representations for time series by preserving their DTW distances. Finally, RWS [125, 124], builds on the RFF kernel method described in Section 2 to learn representations.

8. CONCLUSIONS

In this paper, we addressed the problem of efficiently learning data-aware representations. First, we developed SINK, a fast kernel function to compare time series under shift invariances. We constructed landmark time series using effective time-series clustering and we presented a method to estimate kernel function parameters and improve the compactness of the representations. Then, we learned representations using GRAIL by exploiting approximations for kernel methods. Finally, we showed how GRAIL representations accelerate kernel methods for five major time-series mining tasks. We evaluated our ideas by conducting an extensive experimental evaluation on 128 datasets using a rigorous statistical analysis. Additionally, we implemented GRAIL over Apache Spark to analyze real-world IoT data. Our findings suggest that by using SINK and GRAIL, we can significantly outperform existing state-of-the-art methods for querying, classification, clustering, sampling, and visualization of time series. GRAIL emerges as a new primitive capable to unify the design of time-series methods.

Acknowledgments: We thank the anonymous reviewers whose comments have greatly improved this manuscript. We also thank Christos Faloutsos and Eamonn Keogh for useful discussions and Luis Gravano and Daniel Hsu for invaluable feedback. We gratefully acknowledge the contribution of Yiyang Ou in the implementation of GRAIL over Apache Spark. This research was supported in part by NetApp, Cisco, and an NSF CISE Expeditions Award CCF-1139158. Part of this work was done while J.P. was at Columbia University using computing resources from Columbia University's Shared Research Computing Facility project.

9. REFERENCES

[1] R. H. Affandi, A. Kulesza, E. Fox, and B. Taskar. Nystrom approximation for large-scale determinantal processes. In *Artificial Intelligence and Statistics*, pages 85–98, 2013.

[2] R. Agrawal, C. Faloutsos, and A. Swami. Efficient similarity search in sequence databases. *FODA*, pages 69–84, 1993.

[3] A. Aizerman, E. Braverman, and L. Rozoner. Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control*, 25:821–837, 1964.

[4] S. Alam, F. D. Albareti, C. A. Prieto, F. Anders, S. F. Anderson, T. Anderton, B. H. Andrews, E. Armengaud, É. Aubourg, S. Bailey, et al. The eleventh and twelfth data releases of the sloan digital sky survey: final data from sdss-iii. *The Astrophysical Journal Supplement Series*, 219(1):12, 2015.

[5] T. Argyros and C. Ermopoulos. Efficient subsequence matching in time series databases under time and amplitude transformations. In *ICDM*, pages 481–484. IEEE, 2003.

[6] O. T. at Twitter. Observability at Twitter: technical overview, part I, 2016.

[7] F. R. Bach and M. I. Jordan. Predictive low-rank decomposition for kernel methods. In *ICML*, pages 33–40. ACM, 2005.

[8] O. Bachem, M. Lucic, H. Hassani, and A. Krause. Fast and provably good seedings for k-means. In *NeurIPS*, pages 55–63, 2016.

[9] A. Bagnall, J. Lines, A. Bostrom, J. Large, and E. Keogh. The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *Data Mining and Knowledge Discovery*, 31(3):606–660, 2017.

[10] A. Bagnall, J. Lines, J. Hills, and A. Bostrom. Time-series classification with cote: the collective of transformation-based ensembles. *IEEE Transactions on Knowledge and Data Engineering*, 27(9):2522–2535, 2015.

[11] C. Bahlmann, B. Haasdonk, and H. Burkhardt. Online handwriting recognition with support vector machines-a kernel approach. In *Frontiers in handwriting recognition, 2002. proceedings. eighth international workshop on*, pages 49–54. IEEE, 2002.

[12] R. Bamler and S. Mandt. Improving optimization in models with continuous symmetry breaking. In *ICML*, pages 432–441, 2018.

[13] Y. Bengio, A. C. Courville, and P. Vincent. Unsupervised feature learning and deep learning: A review and new perspectives. *CoRR, abs/1206.5538*, 1, 2012.

[14] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle. Greedy layer-wise training of deep networks. In *NeurIPS*, pages 153–160, 2007.

[15] Y. Bengio, Y. LeCun, et al. Scaling learning algorithms towards ai. *Large-scale kernel machines*, 34(5):1–41, 2007.

[16] B. B. Biswal, M. Mennes, X.-N. Zuo, S. Gohel, C. Kelly, S. M. Smith, C. F. Beckmann, J. S. Adelstein, R. L. Buckner, S. Colcombe, et al. Toward discovery science of human brain function. *Proceedings of the National Academy of Sciences*, 107(10):4734–4739, 2010.

[17] G. E. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

[18] S. Brahim-Belhouari and A. Bermak. Gaussian process for nonstationary time series prediction. *Computational Statistics & Data Analysis*, 47(4):705–712, 2004.

[19] Y. Cai and R. Ng. Indexing spatio-temporal trajectories with chebyshev polynomials. In *SIGMOD*, pages 599–610. ACM, 2004.

[20] A. Camerra, T. Palpanas, J. Shieh, and E. Keogh. isax 2.0: Indexing and mining one billion time series. In *ICDM*, pages 58–67. IEEE, 2010.

[21] F.-P. Chan, A.-C. Fu, and C. Yu. Haar wavelets for efficient similarity search of time-series: with and without time warping. *IEEE Transactions on Knowledge and Data Engineering*, 15(3):686–705, 2003.

[22] K.-P. Chan and A. W.-C. Fu. Efficient time series matching by wavelets. In *ICDE*, pages 126–133. IEEE, 1999.

[23] C.-C. Chang and C.-J. Lin. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2(3):27, 2011.

[24] L. Chen and R. Ng. On the marriage of lp-norms and edit distance. In *VLDB*, pages 792–803. VLDB Endowment, 2004.

[25] L. Chen, M. T. Özsu, and V. Oria. Robust and fast similarity search for moving object trajectories. In *SIGMOD*, pages 491–502, 2005.

[26] C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[27] T. F. Cox and M. A. Cox. *Multidimensional scaling*. CRC press, 2000.

[28] N. Cristianini and J. Shawe-Taylor. *An introduction to support vector machines and other kernel-based learning methods*. Cambridge university press, 2000.

[29] J. P. Cunningham and Z. Ghahramani. Linear dimensionality reduction: survey, insights, and generalizations. *Journal of Machine Learning Research*, 16(1):2859–2900, 2015.

[30] M. Cuturi. Fast global alignment kernels. In *ICML*, pages 929–936, 2011.

[31] M. Cuturi, J.-P. Vert, O. Birkenes, and T. Matsui. A kernel for time series based on global alignments. In *ICASSP*, volume 2, pages II–413. IEEE, 2007.

[32] H. A. Dau, E. Keogh, K. Kamgar, C.-C. M. Yeh, Y. Zhu, S. Gharghabi, C. A. Ratanamahatana, Yanping, B. Hu, N. Begum, A. Bagnall, A. Mueen, and G. Batista. The ucr time series classification archive, October 2018. `https://www.cs.ucr.edu/~eamonn/time_series_data_2018/`.

[33] J. Demšar. Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research*, 7:1–30, 2006.

[34] H. Deng, G. Runger, E. Tuv, and M. Vladimir. A time series forest for classification and feature extraction. *Information Sciences*, 239:142–153, 2013.

[35] H. Ding, G. Trajcevski, P. Scheuermann, X. Wang, and E. Keogh. Querying and mining of time series data: experimental comparison of representations and distance measures. *PVLDB*, 1(2):1542–1552, 2008.

[36] R. Ding, Q. Wang, Y. Dang, Q. Fu, H. Zhang, and D. Zhang. Yading: Fast clustering of large-scale time

series data. *PVLDB*, 8(5):473–484, 2015.
[37] P. Drineas and M. W. Mahoney. On the nyström method for approximating a gram matrix for improved kernel-based learning. *journal of machine learning research*, 6(Dec):2153–2175, 2005.
[38] P. Esling and C. Agon. Time-series data mining. *ACM Computing Surveys*, 45(1):12, 2012.
[39] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos. Fast subsequence matching in time-series databases. In *SIGMOD*, pages 419–429, 1994.
[40] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. Liblinear: A library for large linear classification. *Journal of Machine Learning Research*, 9(Aug):1871–1874, 2008.
[41] M. Fiterau, J. Fries, E. Halilaj, N. Siranart, S. Bhooshan, and C. Re. Similarity-based lstms for time series representation learning in the presence of structured covariates. In *29th Conference on Neural Information Processing Systems*, 2016.
[42] C. Fowlkes, S. Belongie, F. Chung, and J. Malik. Spectral grouping using the nystrom method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(2):214–225, 2004.
[43] M. Friedman. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association*, 32:675–701, 1937.
[44] B. D. Fulcher. Feature-based time-series analysis. *arXiv preprint arXiv:1709.08055*, 2017.
[45] B. D. Fulcher and N. S. Jones. Highly comparative feature-based time-series classification. *IEEE Transactions on Knowledge and Data Engineering*, 26(12):3026–3037, 2014.
[46] B. D. Fulcher, M. A. Little, and N. S. Jones. Highly comparative time-series analysis: the empirical structure of time series and their methods. *Journal of The Royal Society Interface*, 10(83):20130048, 2013.
[47] A. Gittens and M. W. Mahoney. Revisiting the nyström method for improved large-scale machine learning. *J. Mach. Learn. Res*, 28(3):567–575, 2013.
[48] D. Q. Goldin and P. C. Kanellakis. On similarity queries for time-series data: constraint specification and implementation. In *CP*, pages 137–153. Springer, 1995.
[49] G. H. Golub and C. F. Van Loan. *Matrix computations*, volume 3. JHU Press, 2012.
[50] B. Haasdonk and C. Bahlmann. Learning with distance substitution kernels. In *DAGM-Symposium*, volume 3175, pages 220–227. Springer, 2004.
[51] J. Han, J. Pei, and M. Kamber. *Data mining: concepts and techniques*. Elsevier, 2011.
[52] D. Haussler. Convolution kernels on discrete structures. Technical report, Technical report, Department of Computer Science, University of California at Santa Cruz, 1999.
[53] T. Hofmann, B. Schölkopf, and A. J. Smola. Kernel methods in machine learning. *The annals of statistics*, pages 1171–1220, 2008.
[54] H. Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of educational psychology*, 24(6):417, 1933.
[55] R. J. Hyndman and G. Athanasopoulos. *Forecasting: principles and practice*. OTexts, 2014.
[56] A. K. Jain. *Fundamentals of digital image processing*. Prentice-Hall, Inc., 1989.
[57] E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra. Dimensionality reduction for fast similarity search in large time series databases. *Knowledge and information Systems*, 3(3):263–286, 2001.
[58] E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra. Locally adaptive dimensionality reduction for indexing large time series databases. *ACM Sigmod Record*, 30(2):151–162, 2001.
[59] E. Keogh and S. Kasetty. On the need for time series data mining benchmarks: a survey and empirical demonstration. *Data Mining and knowledge discovery*, 7(4):349–371, 2003.
[60] E. Keogh, T. Palpanas, V. B. Zordan, D. Gunopulos, and M. Cardle. Indexing large human-motion databases. In *VLDB*, VLDB '04, pages 780–791. VLDB Endowment, 2004.
[61] E. Keogh and C. A. Ratanamahatana. Exact indexing of dynamic time warping. *Knowledge and information systems*, 7(3):358–386, 2005.
[62] S.-W. Kim, S. Park, and W. W. Chu. An index-based approach for similarity search supporting time warping in large sequence databases. In *ICDE*, pages 607–614. IEEE, 2001.
[63] F. Korn, H. V. Jagadish, and C. Faloutsos. Efficiently supporting ad hoc queries in large datasets of time sequences. In *SIGMOD*, SIGMOD '97, pages 289–300, New York, NY, USA, 1997. ACM.
[64] A. Kulesza and B. Taskar. Structured determinantal point processes. In *NeurIPS*, pages 1171–1179, 2010.
[65] S. Kumar, M. Mohri, and A. Talwalkar. Sampling methods for the nyström method. *Journal of Machine Learning Research*, 13(Apr):981–1006, 2012.
[66] M. Längkvist, L. Karlsson, and A. Loutfi. A review of unsupervised feature learning and deep learning for time-series modeling. *Pattern Recognition Letters*, 42:11–24, 2014.
[67] J. A. Lee and M. Verleysen. *Nonlinear dimensionality reduction*. Springer Science & Business Media, 2007.
[68] Q. Lei, J. Yi, R. Vaculin, L. Wu, and I. S. Dhillon. Similarity preserving representation learning for time series analysis. *arXiv preprint arXiv:1702.03584*, 2017.
[69] C. Li, S. Jegelka, and S. Sra. Fast dpp sampling for nystr\" om with application to kernel methods. *arXiv preprint arXiv:1603.06052*, 2016.
[70] T. W. Liao. Clustering of time series dataa survey. *Pattern recognition*, 38(11):1857–1874, 2005.
[71] E. Liberty. Simple and deterministic matrix sketching. In *SIGKDD*, pages 581–588. ACM, 2013.
[72] J. Lin, E. Keogh, S. Lonardi, and B. Chiu. 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*, pages 2–11. ACM, 2003.
[73] J. Lines and A. Bagnall. Time series classification with ensembles of elastic distance measures. *Data Mining and Knowledge Discovery*, 29(3):565–592, 2015.
[74] C. Loboz, S. Smyl, and S. Nath. Datagarage: Warehousing massive performance data on

commodity servers. *PVLDB*, 3(1-2):1447–1458, 2010.
[75] H. Lütkepohl. *New introduction to multiple time series analysis*. Springer Science & Business Media, 2005.
[76] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *BSMSP*, pages 281–297, 1967.
[77] M. S. Mahdavinejad, M. Rezvan, M. Barekatain, P. Adibi, P. Barnaghi, and A. P. Sheth. Machine learning for internet of things data analysis: A survey. *Digital Communications and Networks*, 2017.
[78] V. Megalooikonomou, G. Li, and Q. Wang. A dimensionality reduction technique for efficient similarity analysis of time series databases. In *CIKM*, pages 160–161. ACM, 2004.
[79] V. Megalooikonomou, Q. Wang, G. Li, and C. Faloutsos. A multiresolution symbolic representation of time series. In *ICDE*, pages 668–679. IEEE, 2005.
[80] X. Meng, J. Bradley, B. Yavuz, E. Sparks, S. Venkataraman, D. Liu, J. Freeman, D. Tsai, M. Amde, S. Owen, et al. Mllib: Machine learning in apache spark. *The Journal of Machine Learning Research*, 17(1):1235–1241, 2016.
[81] J. Mercer. Functions of positive and negative type, and their connection with the theory of integral equations. *Philosophical transactions of the royal society of London. Series A, containing papers of a mathematical or physical character*, 209:415–446, 1909.
[82] C. Musco and C. Musco. Recursive sampling for the nystrom method. In *NeurIPS*, pages 3833–3845, 2017.
[83] A. Nanopoulos, R. Alcock, and Y. Manolopoulos. Feature-based classification of time-series data. *International Journal of Computer Research*, 10(3):49–61, 2001.
[84] P. Nemenyi. *Distribution-free Multiple Comparisons*. PhD thesis, Princeton University, 1963.
[85] A. Y. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *NeurIPS*, pages 849–856, 2002.
[86] E. J. Nyström. Über die praktische auflösung von integralgleichungen mit anwendungen auf randwertaufgaben. *Acta Mathematica*, 54(1):185–204, 1930.
[87] D. Oglic and T. Gärtner. Nyström method with kernel k-means++ samples as landmarks. In D. Precup and Y. W. Teh, editors, *ICML*, volume 70 of *Proceedings of Machine Learning Research*, pages 2652–2660, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.
[88] A. V. Oppenheim and R. W. Schafer. *Discrete-Time Signal Processing*. Prentice Hall Press, Upper Saddle River, NJ, USA, 3rd edition, 2009.
[89] T. Palpanas. Data series management: the road to big sequence analytics. *ACM SIGMOD Record*, 44(2):47–52, 2015.
[90] J. Paparrizos. 2018 ucr time-series archive: Backward compatibility, missing values, and varying lengths, January 2019. `https://github.com/johnpaparrizos/UCRArchiveFixes`.
[91] J. Paparrizos and L. Gravano. k-shape: Efficient and accurate clustering of time series. In *SIGMOD*, pages 1855–1870. ACM, 2015.
[92] J. Paparrizos and L. Gravano. Fast and accurate time-series clustering. *ACM Transactions on Database Systems (TODS)*, 42(2):8, 2017.
[93] K. Pearson. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901.
[94] T. Pelkonen, S. Franklin, J. Teller, P. Cavallaro, Q. Huang, J. Meza, and K. Veeraraghavan. Gorilla: A fast, scalable, in-memory time series database. *PVLDB*, 8(12):1816–1827, 2015.
[95] D. B. Percival and A. T. Walden. *Wavelet methods for time series analysis*, volume 4. Cambridge university press, 2006.
[96] I. Popivanov and R. J. Miller. Similarity search over time-series data using wavelets. In *ICDE*, pages 212–221. IEEE, 2002.
[97] L. Rabiner and B. Juang. An introduction to hidden markov models. *IEEE ASSP Magazine*, 3(1):4–16, 1986.
[98] A. Rahimi and B. Recht. Random features for large-scale kernel machines. In *NeurIPS*, pages 1177–1184, 2008.
[99] 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*, pages 262–270. ACM, 2012.
[100] W. M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66(336):846–850, 1971.
[101] K. V. Ravi Kanth, D. Agrawal, and A. Singh. Dimensionality reduction for similarity searching in dynamic databases. In *SIGMOD*, SIGMOD '98, pages 166–176, New York, NY, USA, 1998. ACM.
[102] J. Rice. *Mathematical statistics and data analysis*. Cengage Learning, 2006.
[103] H. Sakoe and S. Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE transactions on acoustics, speech, and signal processing*, 26(1):43–49, 1978.
[104] Y. Sakurai, M. Yoshikawa, and C. Faloutsos. Ftw: fast similarity search under the time warping distance. In *PODS*, pages 326–337. ACM, 2005.
[105] P. Schäfer and M. Högqvist. Sfa: a symbolic fourier approximation and index for similarity search in high dimensional datasets. In *EDBT*, pages 516–527. ACM, 2012.
[106] B. Schölkopf. The kernel trick for distances. In *NeurIPS*, pages 301–307, 2001.
[107] B. Schölkopf, A. Smola, and K.-R. Müller. Kernel principal component analysis. In *ICANN*, pages 583–588. Springer, 1997.
[108] B. Schölkopf, A. Smola, and K.-R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural computation*, 10(5):1299–1319, 1998.
[109] B. Schölkopf and A. J. Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002.
[110] J. Serrà, S. Pascual, and A. Karatzoglou. Towards a universal neural network encoder for time series. In *CCIA*, pages 120–129, 2018.
[111] D. Shasha and Y. Zhu. *High Performance Discovery In Time Series: Techniques And Case Studies*

(Monographs in Computer Science). SpringerVerlag, 2004.

[112] H. Shatkay and S. B. Zdonik. Approximate queries and representations for large data sequences. In *ICDE*, pages 536–545. IEEE, 1996.

[113] H. Shimodaira, K.-i. Noma, M. Nakai, and S. Sagayama. Dynamic time-alignment kernel in support vector machine. In *NeurIPS*, pages 921–928, 2002.

[114] K. Shin and T. Kuboyama. A generalization of haussler's convolution kernel: mapping kernel. In *ICML*, pages 944–951. ACM, 2008.

[115] Z. R. Struzik and A. Siebes. Measuring time series similarity through large singular features revealed with wavelet transformation. In *Database and Expert Systems Applications, 1999. Proceedings. Tenth International Workshop on*, pages 162–166. IEEE, 1999.

[116] W. S. Torgerson. Multidimensional scaling: I. theory and method. *Psychometrika*, 17(4):401–419, 1952.

[117] M. Vlachos, D. Gunopulos, and G. Das. Indexing time-series under conditions of noise. *Data mining in time series databases*, 57:67–100, 2004.

[118] G. Wachman, R. Khardon, P. Protopapas, and C. R. Alcock. Kernels for periodic time series arising in astronomy. In *ECML-PKDD*, pages 489–505. Springer, 2009.

[119] X. Wang, A. Mueen, H. Ding, G. Trajcevski, P. Scheuermann, and E. Keogh. Experimental comparison of representation methods and distance measures for time series data. *Data Mining and Knowledge Discovery*, pages 1–35, 2013.

[120] X. Wang, K. Smith, and R. Hyndman. Characteristic-based clustering for time series data. *Data Mining and Knowledge Discovery*, 13(3):335–364, 2006.

[121] Z. Wang, W. Yan, and T. Oates. Time series classification from scratch with deep neural networks: A strong baseline. In *IJCNN*, pages 1578–1585. IEEE, 2017.

[122] F. Wilcoxon. Individual comparisons by ranking methods. *Biometrics Bulletin*, pages 80–83, 1945.

[123] C. K. Williams and M. Seeger. Using the nyström method to speed up kernel machines. In *NeurIPS*, pages 682–688, 2001.

[124] L. Wu, I. E.-H. Yen, F. Xu, P. Ravikumar, and M. Witbrock. D2ke: From distance to kernel and embedding. *arXiv preprint arXiv:1802.04956*, 2018.

[125] L. Wu, I. E.-H. Yen, J. Yi, F. Xu, Q. Lei, and M. Witbrock. Random warping series: A random features method for time-series embedding. In *AISTATS*, pages 793–802, 2018.

[126] X. Xi, E. Keogh, C. Shelton, L. Wei, and C. A. Ratanamahatana. Fast time series classification using numerosity reduction. In *ICML*, pages 1033–1040. ACM, 2006.

[127] Q. Yang and X. Wu. 10 challenging problems in data mining research. *International Journal of Information Technology & Decision Making*, 5(04):597–604, 2006.

[128] T. Yang, Y.-F. Li, M. Mahdavi, R. Jin, and Z.-H. Zhou. Nyström method vs random fourier features: A theoretical and empirical comparison. In *NeurIPS*, pages 476–484, 2012.

[129] B.-K. Yi and C. Faloutsos. Fast time sequence indexing for arbitrary lp norms. VLDB, 2000.

[130] B.-K. Yi, H. Jagadish, and C. Faloutsos. Efficient retrieval of similar time sequences under time warping. In *ICDE*, pages 201–208. IEEE, 1998.

[131] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. J. Franklin, S. Shenker, and I. Stoica. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In *NSDI*, pages 2–2. USENIX Association, 2012.

[132] G. Zhang, B. E. Patuwo, and M. Y. Hu. Forecasting with artificial neural networks:: The state of the art. *International Journal of Forecasting*, 14(1):35–62, 1998.

[133] K. Zhang, I. W. Tsang, and J. T. Kwok. Improved nyström low-rank approximation and error analysis. In *ICML*, pages 1232–1239. ACM, 2008.

[134] G. Zheng, Y. Yang, and J. Carbonell. Efficient shift-invariant dictionary learning. In *SIGKDD*, pages 2095–2104. ACM, 2016.

[135] K. Zoumpatianos, S. Idreos, and T. Palpanas. Indexing for interactive exploration of big data series. In *SIGMOD*, pages 1555–1566. ACM, 2014.