Representative Time Series Discovery for Data Exploration

Ge Lee
RMIT University
Melbourne, Australia
ge.lee@student.rmit.edu.au

Shixun Huang
University of Wollongong
Wollongong, Australia
shixunh@uow.edu.au

Zhifeng Bao*
RMIT University
Melbourne, Australia
zhifeng.bao@rmit.edu.au

Yanchang Zhao
Data61, CSIRO
Canberra, Australia
yanchang.zhao@data61.csiro.au

ABSTRACT

In this work, we address the critical task of discovering representative time series in exploratory data mining. We define a representative time series, referred to as similarity-bounded representative time series, as one that represents other time series if their similarity meets a user-defined threshold. Building on this definition, we study the problem of finding the smallest set of such time series that can represent a specified proportion of all time series within the dataset. The representativeness of each similarity-bounded representative time series is controllable and determined by the specified level of similarity, and only the minimum number of such representatives needed to collectively represent the specified proportion of entire set are identified. Identifying representative time series over large-scale data in an efficient and effective manner facilitates exploratory data analysis and summary generation, serving a wide range of data exploration applications across diverse domains. We first prove the NP-hardness of this problem and propose a range of approximation methods with theoretical guarantees, and we refer to them as non-learning-based methods. While effective, these methods often excel in either running time or memory efficiency, but not both concurrently. To overcome these limitations, we further propose a learning-based method that simultaneously optimizes both time and memory efficiency. This method leverages novel data preparation and training strategies, providing adaptability to user-specified representativeness requirements with low memory usage and computational overhead. We conduct extensive experiments across four real-world datasets to demonstrate that our learning-based method is highly competitive with non-learning-based methods in terms of effectiveness (produces similar number of representative time series), while achieving significantly higher efficiency (up to 21× speedups) and lower memory consumption (saving up to 101× memory space).

PVLDB Reference Format:
Ge Lee, Shixun Huang, Zhifeng Bao, and Yanchang Zhao. Representative Time Series Discovery for Data Exploration. PVLDB, 18(3): 915 - 928, 2024.
doi:10.14778/3712221.3712252

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/rmitbggroup/RTSD.

*The corresponding author.

Proceedings of the VLDB Endowment, Vol. 18, No. 3 ISSN 2150-8097.
doi:10.14778/3712221.3712252

1 INTRODUCTION

In recent years, the widespread use of modern sensors has driven a substantial increase in time series data across various fields [2, 17, 21, 44]. Time series data, which record variables over time at regular intervals, are commonly stored in databases for subsequent analysis [9, 19, 28, 45, 54]. However, the high-dimensional nature of these data renders their processing highly complex. Large-scale time series data, in particular, pose difficulties for human analysts to handle, explore and analyze them effectively. To mitigate these challenges, it is essential to efficiently identify representative time series in large-scale data, as this facilitates exploratory data analysis and summary generation [4, 52, 54, 59], serving a wide range of data exploration applications across diverse domains.

Despite the importance of this task, there has been limited focus on the discovery of representative time series, which involves identifying a small subset of time series that effectively captures the patterns present in the entire dataset. We define a representative time series as one that represents other time series if their similarity meets the user-defined threshold, which we refer to as *similarity-bounded representative time series*. Given a set of time series, our objective is to select the minimum number of such representatives so that every remaining time series can always be represented by at least one of the selected representatives, meeting a specified coverage threshold proportion of all time series in the dataset. This problem is formulated as similarity-bounded Representative Time Series Discovery (RTSD). RTSD exhibits two sweet properties: (1) the representativeness of each similarity-bounded representative time series is controllable and determined by the user-specified level of similarity, and (2) only the minimum number of such time series, which collectively represent the required proportion of entire set, are considered as representatives. Next, we explore the practical benefits of applying RTSD in real-world applications.

Our work serves as a framework that empowers human-driven analysis across different domains. In many real-world scenarios, effective data analysis relies on human interpretation and expertise, which is expensive and cannot always be automated [50]. Our framework reduces data volume while supporting refinement based on analysis outcomes, enabling analysts to focus on a more manageable subset of data. Through visualization, we also minimize visual clutter. These align with human cognitive capabilities, allowing for a more intuitive analysis process [58]. By making data easier to interpret and analyze, our framework enhances decision-making. In applications such as industrial monitoring, weather analysis, medical diagnosis, and seismic monitoring, our work allows analyst to extract insights without overwhelming them with excessive data. In what follows, we illustrate how our framework can be applied to traffic analysis and industrial monitoring.

Example 1. *Given a collection of 20 sensors, each recording traffic speed as a time series on different roads, an analyst seeks to identify representative speed profiles for a specific area. The goal is to select a subset of sensors that represent others with similar speeds and patterns and ensure full coverage of the road network. Drawing upon domain knowledge, the analyst specifies a desired similarity threshold that a representative road should meet to effectively represent others and sets the coverage threshold to maximum to capture the entire network. To minimize visual clutter, the analyst aims to find the minimum number of representatives needed. Figures 1a and 1b illustrate how two representative time series (sensors 4 and 13) capture the speed patterns of other sensors in similar color. The analyst can further explore specific roads of interest by selecting the sensor for additional information, such as detailed time series data or connections to nearby roads. To gain more precise insight into specific traffic patterns, the analyst increases the similarity threshold. This adjustment limits representation to only time series with higher similarity, reducing the coverage of each representative and requiring more representatives to collectively cover the entire set. Figures 1c and 1d show new representative traffic patterns in red, characterized by low-speed congestion throughout the day, with other roads (sensors 2, 5, 10, 15, 16, 19) exhibiting similar pattern.*

Example 2. *In industrial monitoring, detection of malfunctions is crucial. Large sensor networks often generate extensive time series data with high temporal correlation due to redundancy [26]. Our work effectively manages these massive datasets by capturing a set of representative time series. This representative set consolidates similar time series data, highlights important information and reduces redundancy. Importantly, we ensure that all time series within the specified coverage threshold are represented. This prevents missed signals that could indicate system issues. As industrial environments emphasize preventive maintenance, our framework allows human experts with domain knowledge to identify representative patterns indicative of potential issues and pinpoint specific sensors through their corresponding representatives. Consequently, analysts can focus on the most critical insights through this smaller, manageable representative set, thereby facilitating informed decision-making and preventing oversights that could lead to potentially catastrophic system failures [26]. Furthermore, our framework provides the flexibility to adjust similarity and coverage thresholds and supports different similarity measures to tailor analysis to specific needs.*

Astute readers may find that studies on time series clustering [2, 21, 35] and object diversification [14, 48] share some common ground with our problem. However, extending them to solve our problem poses difficulties. In clustering, explicitly defining the relationship between representative and represented time series remains unclear and is an open problem. If we apply our similarity threshold to each cluster center, they may fall short in covering and representing the required proportion of the time series, potentially causing the analyst to overlook critical traffic patterns from uncovered sensors. On the other hand, object diversification aims to select diverse objects based on a defined diversity constraint, e.g., the selected objects have distances greater than a specified threshold. However, due to this constraint, the number of selected objects can highly exceed the minimum required to represent the

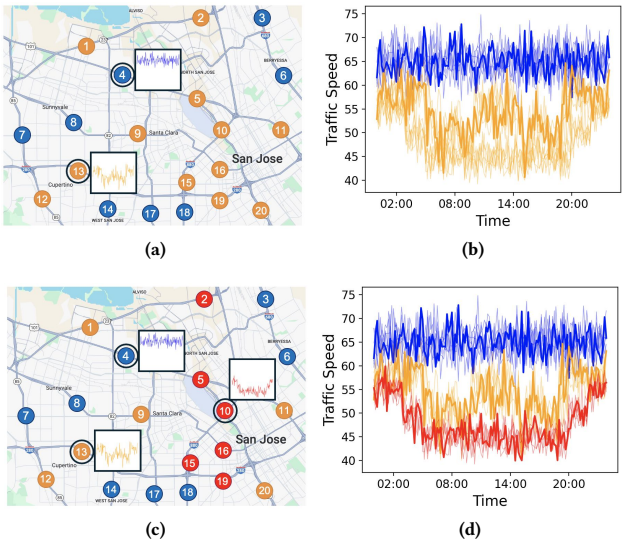


Figure 1: (a) shows sensors on various road segments, each colored according to their respective traffic patterns (time series) shown in (b). Representative time series are depicted in thick, distinct colors, with their corresponding sensors circled on the map. The remaining time series, represented by these representatives, are shown in thinner lines of the same color, corresponding to the uncircled sensors. e.g., Sensor 4 represents other sensors in blue since their traffic patterns are highly similar. (c) and (d) illustrate the scenario with a higher similarity threshold.

required set, which may lead to inefficiencies in subsequent analysis for the analyst. We refer readers to Section 6 for a detailed comparison between time series clustering, object diversification and our problem.

We prove that the RTSD problem is NP-hard and its coverage function is monotone and submodular. This motivates us to develop a greedy algorithm (`Greedy`) to approximate the optimal solution with theoretical guarantees. However, the greedy algorithm relies on time-intensive computations of similar time series for representatives selection, which becomes a bottleneck due to the high dimensionality of the data. To mitigate this, we introduce an early termination technique (`GreedyET`) that prunes time series with low coverage during selection, which significantly reduces the runtime. Further optimizing time efficiency, we propose two alternative variants of the greedy algorithm (`PreGreedy` and `PreGreedyET`) that precompute similar time series, thereby saving substantial computational time compared to the previous methods. Nevertheless, these approaches result in a notable increase in memory consumption due to precomputation. We refer to these four time series selection methods above as non-learning-based methods, as summarized in Table 2.

Considering the aforementioned limitations, we propose a novel self-supervised learning approach (`MLGreedyET`). It involves: (1) preparing high-quality data by generating informative features based on the `GreedyET` framework, (2) leveraging these features in a learning model to estimate the representativeness of a time series, and (3) selecting the representative time series based on the

Table 1: Frequently used notations.

Notation	Description
$\mathcal{T}$	a time series database
T, E	a univariate time series, a time series embedding
S_T	the similar set of T
SIM	the similarity measure
τ, β	the similarity threshold, the coverage threshold
$\mathcal{X}$	a set of representative time series
$\sigma(\mathcal{X})$	the set of time series covered/represented by $\mathcal{X}$

GreedyET framework. Estimating the representativeness of a time series not only saves considerable time from computing similar time series but also eliminates the need for additional memory storage for precomputing similar time series. This learning-based method produces effective solutions while striking a balance between time efficiency and memory cost, as outlined in Table 2.

In summary, we make the following technical contributions:

- We formulate and study the similarity-bounded Representative Time Series Discovery (RTSD) problem, which aims to select the minimal number of similarity-bounded representative time series that effectively represent the specified proportion of time series within a dataset (Section 2).
- We prove that the RTSD problem is NP-hard and its coverage function is monotone and submodular (Section 2). Therefore, we propose several variants of the greedy algorithm, including the initial Greedy, followed by GreedyET to speed up Greedy, as well as the enhanced faster version PreGreedy and PreGreedyET to efficiently produce solutions with an approximation ratio (Section 3).
- To reduce time cost and memory footprints, we further propose a self-supervised learning approach that incorporates the greedy algorithm design into both training and testing phases to produce highly effective solution while ensuring high efficiency and low memory consumption simultaneously (Section 4).
- We conduct extensive experiments across four real-world datasets to demonstrate that our learning-based method is highly competitive with non-learning-based methods in terms of effectiveness (produces similar number of representative time series), while achieving significantly higher efficiency (up to 21× speedups) and lower memory consumption (saving up to 101× memory space). Furthermore, we present a visualization case study that illustrates the practicality of our method in a real-world scenario (Section 5).

2 PROBLEM FORMULATION AND HARDNESS ANALYSIS

In this section, we present the problem formulation and analyze its hardness and theoretical properties. Frequently used notations are summarized in Table 1. Our work focuses on univariate time series [18, 21], which we simply refer to as *time series* throughout the paper.

Definition 1 (Time Series). *A time series $T = \{p_1, p_2, \ldots, p_{|T|}\}$ is a sequence of points, where each point $p_i = (t_i, v_i)$ is associated with a timestamp t_i and a value v_i. $|T|$ denotes the length of the time series and each point p_i is ordered chronologically.*

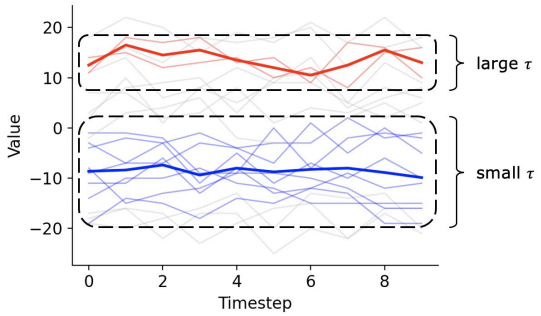


Figure 2: An illustration on the coverage of similar set of two representative time series (colored in thick red and blue), one with a larger τ and the other with a smaller τ. The thinner colored lines depict the corresponding time series represented by each representatives, while the gray lines depict the time series not represented by either of them.

Definition 2 (Similar Set). *Given a time series database $\mathcal{T}$ of multiple time series, a similarity measure SIM, a similarity threshold $\tau \in [0, 1]$ and a time series $T \in \mathcal{T}$, the similar set S_T of T consists of all time series in $\mathcal{T}$ whose similarity to T is at least τ based on SIM, i.e., $S_T = \{T' | T' \in \mathcal{T}, SIM(T, T') \geq \tau\}$.*

In our work, we select Euclidean distance [22] as our similarity measure, i.e., $SIM(T, T') = 1 - D(T, T')$, where $D \in [0, 1]$ is the normalized Euclidean distance. The Euclidean distance is a widely adopted metric in similarity search literature [18, 19, 32, 46, 51] and it is proven to be an effective measure, particularly for large collections of time series [12, 53].

Definition 3 (Coverage Function). *A coverage function $\sigma(X) = \cup_{T \in X} S_T$ is defined as the union of similar sets of a set of time series X, where $X \subseteq \mathcal{T}$.*

Definition 4 (Representative Time Series Discovery (RTSD)). *Given a time series database $\mathcal{T}$, a similarity measure SIM, a similarity threshold τ and a coverage threshold $\beta \in [0, 1]$, the objective of the RTSD problem is to find the smallest set of representative time series $\mathcal{X}$ that covers at least β proportion of the entire time series database, i.e., $\mathcal{X}^* = \arg\min_{\mathcal{X} \subseteq \mathcal{T} \wedge |\sigma(\mathcal{X})| \geq \beta \cdot |\mathcal{T}|} |\mathcal{X}|$.*

We introduce a coverage threshold β into our problem formulation to provide greater flexibility in defining the proportion of the time series database to be covered. This parameter allows for the mitigation of outliers depending on the dataset characteristics. Setting $\beta = 1$ implies that the selected representative time series will collectively cover the entire time series database $\mathcal{T}$.

The *representativeness* of a time series is reflected in the coverage of its associated similar set. A larger coverage within the similar set for a given time series allows it to encompass a greater number of similar time series, thereby enhancing its overall representativeness. The similarity threshold τ is the key to controlling the coverage of the similar set. A lower value of τ results in a broader coverage. This, in turn, has an impact on the representativeness, as a lower τ allows fewer representative time series to effectively capture the desired coverage of the time series database. Figure 2 illustrates the impact of τ on the coverage of the similar set.

Table 2: Overview of our proposed solutions.

Method	Time Complexity	Space Complexity
Greedy (Section 3.1)	$O(\beta\|\mathcal{T}\|^3)$	$O(\|\mathcal{T}\|)$
GreedyET (Section 3.2)	$O(\beta\|\mathcal{T}\|^3)$	$O(\|\mathcal{T}\|)$
PreGreedy (Section 3.3)	$O(\|\mathcal{T}\|^2 + \beta\|\mathcal{T}\|^2)$	$O(\|\mathcal{T}\|^2)$
PreGreedyET (Section 3.3)	$O(\|\mathcal{T}\|^2 + \beta\|\mathcal{T}\|^2)$	$O(\|\mathcal{T}\|^2)$
MLGreedyET (Section 4)	$O(\beta\|\mathcal{T}\|^2)$	$O(\|\mathcal{E}\| + \|\mathcal{T}\|)$

In the following, we will prove that RTSD problem is NP-hard and our coverage function $\sigma(\cdot)$ is monotone and submodular.

Theorem 1. *The RTSD problem is NP-hard.*

Proof. We prove the NP-hardness via a reduction from the problem of Euclidean m-Center on Points (EmCP) which is known NP-complete [42].

Definition 5 (Euclidean m-Center on Points). *Given positive integers m and r, and a set of points $\mathcal{P}$ on a plane Q, the objective of the EmCP problem is to determine if it is possible to select m points from $\mathcal{P}$, where $m < |\mathcal{P}|$, such that each selected point forms the center of a circle c with radius r, covering all points in $\mathcal{P}$. Here, cover means that every point in $\mathcal{P}$ must lie within the circle c of the m selected points.*

For the EmCP problem, we aim to determine whether there exist m circles of radius r that can cover the set $\mathcal{P}$ on the plane Q. We show that this problem can be cast as a special case of the RTSD problem, where each time series comprises a single two-dimensional point, the coverage threshold β is set to 1, and the similarity measure *SIM* is set as the Euclidean distance. Specifically, for each point P in $\mathcal{P}$ on the plane Q, we treat it as a time series T with $r = 1-\tau$, resulting in $S_T = c$. This reduction can be executed in polynomial time. The EmCP problem is equivalent to determining whether there exists a set of m time series such that the union of their similar sets equals $\mathcal{T}$. It is evident that we can find m circles to cover all points in $\mathcal{P}$ if and only if there exist m time series such that the union of their similar sets is $\mathcal{T}$. If a polynomial algorithm exists to solve the RTSD problem optimally, it can also be used to solve EmCP optimally. This is only possible if P = NP. Therefore, the RTSD problem is NP-hard. □

Theorem 2. *The function $\sigma(\cdot)$ is monotonically non-decreasing. Formally, for any $\mathcal{X}_1 \subseteq \mathcal{X}_2$, $|\sigma(\mathcal{X}_1)| \leq |\sigma(\mathcal{X}_2)|$.*

Proof. Since $\sigma(\mathcal{X}_1) = \cup_{T\in\mathcal{X}_1} S_T \subseteq \cup_{T\in\mathcal{X}_2} S_T = \sigma(\mathcal{X}_2)$, the theorem is deduced. □

Theorem 3. *The function $\sigma(\cdot)$ is submodular. Formally, for any $\mathcal{X}_1 \subseteq \mathcal{X}_2$ and $T \in \mathcal{T} \setminus \mathcal{X}_2$, $|\sigma(\mathcal{X}_1 \cup \{T\})| - |\sigma(\mathcal{X}_1)| \geq |\sigma(\mathcal{X}_2 \cup \{T\})| - |\sigma(\mathcal{X}_2)|$.*

Proof. $|\sigma(\mathcal{X}_1 \cup \{T\})| - |\sigma(\mathcal{X}_1)| = |\sigma(\mathcal{X}_1 \cup \{T\}) \setminus \sigma(\mathcal{X}_1)| \geq |\sigma(\mathcal{X}_2 \cup \{T\})| - |\sigma(\mathcal{X}_2)| = |\sigma(\mathcal{X}_2 \cup \{T\}) \setminus \sigma(\mathcal{X}_2)|$. $\sigma(\mathcal{X}_1 \cup \{T\}) \setminus \sigma(\mathcal{X}_1)$ denotes the set of elements that are in $\sigma(\{T\})$ but are not in the union $\cup_{T\in\mathcal{X}_1} S_T$. Clearly, this set is at least at large as the set of elements that are in $\sigma(\{T\})$ but are not in the larger union $\cup_{T\in\mathcal{X}_2} S_T$. That is, $\sigma(\mathcal{X}_1 \cup \{T\}) \setminus \sigma(\mathcal{X}_1) \supseteq \sigma(\mathcal{X}_2 \cup \{T\}) \setminus \sigma(\mathcal{X}_2)$. Therefore, the theorem is deduced. □

Solution Overview. As we have proven that the Representative Time Series Discovery problem is NP-hard, obtaining an optimal solution is not feasible. To provide a practical approach for addressing this challenge, we can attain an efficient approximation of the solution through the greedy strategy, provided our coverage function adheres to both monotonicity and submodularity. Thus, we propose Greedy and GreedyET with early termination technique that speeds up Greedy. However, they still face serious time efficiency issues. To mitigate this, we propose the faster version PreGreedy and PreGreedyET, which notably improve the running time at the expense of high memory usage. Subsequently, we propose a self-learning approach that yields similarly effective solutions, while simultaneously optimizes both time and memory efficiency. We summarize our methods in Table 2.

3 NON-LEARNING-BASED REPRESENTATIVE TIME SERIES SELECTION

As we have established that the RTSD problem is NP-hard in Section 2, obtaining an optimal solution is not feasible. To the best of our knowledge, there exist no approaches that simultaneously address this problem effectively and efficiently. Nonetheless, to provide a practical approach for addressing the inherent challenges, we can attain an efficient approximation of the solution through the greedy strategy, provided that our coverage function adheres to both monotonicity and submodularity.

3.1 Greedy Method

The most straightforward method is known as Greedy, with its pseudocode outlined in Algorithm 1. Greedy operates iteratively, selecting the time series T with the maximum marginal coverage. It covers the most uncovered time series within its associated similar set S_T until the number of covered time series reaches the coverage threshold (lines 3-11). This approach incurs a time cost of $O(\beta|\mathcal{T}|^2)$ for finding the representative time series. Next, we prove that the greedy algorithm for selecting representative time series yields solution with a $1 + ln(\beta|\mathcal{T}|)$ approximation ratio via the following lemma.

Lemma 1. *For all $c > 0$, $\left(1 - \frac{1}{c}\right)^c \leq \frac{1}{e}$, where e is the base of the natural logarithm.*

Proof. We use the fact that for any $z \in \mathbb{R}$, $1 + z \leq e^z$. This follows from the Taylor's expansion, which expresses $e^z = 1 + z + \frac{z^2}{2!} + \frac{z^3}{3!} + \cdots \geq 1 + z$. If we substitute $-\frac{1}{c}$ for z, we can derive $\left(1 - \frac{1}{c}\right) \leq e^{-\frac{1}{c}}$. By raising both sides to the power of c, we obtain the desired result. □

Theorem 4. *Let $\mathcal{X}^*$ be the output of the greedy algorithm and let OPT be an optimal solution. Then, $|\mathcal{X}^*| \leq |OPT| \cdot (1 + ln(\beta|\mathcal{T}|))$.*

Proof. Let g denotes the size of output of the greedy algorithm $|\mathcal{X}^*|$ and c denotes the size of the optimal solution $|OPT|$. We will first demonstrate that $g \leq c \cdot ln(m)$, where $m = \beta|\mathcal{T}|$. Let m_i be the number of time series remaining to be covered after i iterations of the greedy algorithm. Initially, there are $m_0 = m$ time series to be covered. After $i-1$ iterations, m_{i-1} time series remain to be covered. We know that there is a cover of size c for these time series, which

Algorithm 1: `Greedy`

Input : Time series database $\mathcal{T}$, a similarity threshold τ and a coverage threshold β.
Output : A set of representative time series $\mathcal{X}$.

$\mathcal{X} \leftarrow \emptyset$;
$U \leftarrow \mathcal{T}$; *// store the residual time series*
while $|\mathcal{T} \setminus U| < |\mathcal{T}| \cdot \beta$ **do**
  $max\_coverage = 0$;
  foreach $T \in \mathcal{T} \setminus \mathcal{X}$ **do**
    $S_T \leftarrow$ similar set of T based on τ;
    if $|S_T \cap U| > max\_coverage$ **then**
      $max\_coverage = |S_T \cap U|$;
      $T^r = T$;
  $\mathcal{X} \leftarrow \mathcal{X} \cup \{T^r\}$; *// add new representative time series*
  $U \leftarrow U \setminus S_{T^r}$;
return $\mathcal{X}$;

is the optimal cover. Thus, according to the pigeonhole principal, there exists some set that covers at least $\frac{m_{i-1}}{c}$ time series. Since the greedy algorithm selects the set that covers the maximum number of remaining time series, it must select a set covering at least this quantity of time series. The number of time series remaining to be covered is at most $m_i \leq m_{i-1} - \frac{m_{i-1}}{c} = m_{i-1}\left(1 - \frac{1}{c}\right)$. Therefore, in each iteration, the number of remaining time series decreases by a factor of at least $(1 - \frac{1}{c})$. After repeating this process i times, we obtain $m_i \leq m_0\left(1 - \frac{1}{c}\right)^i = m\left(1 - \frac{1}{c}\right)^i$. Given that the greedy algorithm runs for g iterations, it is certain that just prior to the final iteration, there must have been at least one remaining uncovered element. Hence, we have $1 \leq m_g \leq m\left(1 - \frac{1}{c}\right)^g = m\left(\left(1 - \frac{1}{c}\right)^c\right)^{\frac{g}{c}}$. By Lemma 1, we have $1 \leq m\left(\frac{1}{e}\right)^{\frac{g}{c}}$. If we multiply both sides by $e^{\frac{g}{c}}$ and take the natural logarithm, we find that g satisfies $e^{\frac{g}{c}} \leq m$, $\frac{g}{c} \leq ln(m)$, and $g \leq c \cdot ln(m)$. To ensure the validity of the inequality for $m > 0$, we add 1 to $ln(m)$. Therefore, the solution of greedy algorithm for selecting representative time series is larger than the optimum solution by a factor of at most $1 + ln(m)$. □

3.2 `GreedyET` Method

`Greedy` offers an acceptable albeit suboptimal solution, accompanied by the drawback of impractical time costs. To address this, we introduce a considerably more time-efficient method known as *Greedy with Early Termination* (`GreedyET`), with its pseudocode presented in Algorithm 2. Drawing inspiration from an outbreak detection technique called CELF [33], `GreedyET` leverages the submodularity of our objective function to estimate upper bounds of coverage, facilitating the pruning of time series with low coverage. This reduces the number of expensive computations for marginal coverage in each iteration.

Formally, let $\mathcal{X}_i$ denotes the set of selected representative time series after the i-th iteration, and

$$\sigma_\Delta(T|\mathcal{X}_i) = \sigma(\mathcal{X}_i \cup \{T\}) - \sigma(\mathcal{X}_i) \tag{1}$$

denote the marginal coverage of T w.r.t. $\mathcal{X}_i$. Leveraging the submodularity of our coverage function, $\sigma_\Delta(T|\mathcal{X}_i)$ serves as the upper bound for any $\sigma_\Delta(T|\mathcal{X}_j)$, s.t. $\mathcal{X}_i \subseteq \mathcal{X}_j$. Therefore, `GreedyET` first

Algorithm 2: `GreedyET`

Input : Time series database $\mathcal{T}$, a similarity threshold τ and a coverage threshold β.
Output : A set of representative time series $\mathcal{X}$.

1 $\mathcal{X} \leftarrow \emptyset$;
2 $U \leftarrow \mathcal{T}$;
3 $PQ \leftarrow$ an empty priority queue that sorts time series by their upper bound (ub) of marginal coverage in non-increasing order;
4 **while** $|\mathcal{T} \setminus U| < |\mathcal{T}| \cdot \beta$ **do**
5 **if** $\mathcal{X} = \emptyset$ **then**
6 **foreach** $T \in \mathcal{T}$ **do**
7 $S_T \leftarrow$ similar set of T based on τ;
8 Insert T into PQ with $T.ub = |S_T|$;
9 $T^r \leftarrow PQ$.pop();
10 **else**
11 $max\_coverage = 0$;
12 **while** $|PQ| > 0$ **do**
13 $T \leftarrow PQ$.pop();
14 $S_T \leftarrow$ similar set of T based on τ;
15 $T.ub = |S_T \cap U|$;
16 **if** $T.ub > max\_coverage$ **then**
17 $max\_coverage = T.ub$;
18 $T^r = T$;
19 **if** $|PQ| > 0$ **and** $max\_coverage \geq PQ[0].ub$ **then**
20 break;
21 Update PQ with visited T excluding T^r;
22 $\mathcal{X} \leftarrow \mathcal{X} \cup \{T^r\}$;
23 $U \leftarrow U \setminus S_{T^r}$;
24 **return** $\mathcal{X}$;

computes $\sigma_\Delta(T|\emptyset)$ for each time series $T \subseteq \mathcal{T}$ and selects $\mathcal{X}_1$ (lines 5-9). Then, $\sigma_\Delta(T|\emptyset)$ can be used as the upper bound of $\sigma_\Delta(T|\mathcal{X}_i)$ in the next iteration. For each iteration j where $2 \leq j \leq k$, `GreedyET` processes each time series $T \subseteq \mathcal{T} \setminus \mathcal{X}_{j-1}$ in a non-increasing order of their upper bounds of $\sigma_\Delta(T|\mathcal{X}_j)$ and computes $\sigma_\Delta(T|\mathcal{X}_{j-1})$ (lines 12-15). Instead of processing all time series, `GreedyET` triggers an early termination when the maximum upper bound of unprocessed time series is smaller than the maximum $\sigma_\Delta(T|\mathcal{X}_{j-1})$ of processed ones (lines 19-20). Subsequently, `GreedyET` updates the upper bound of each unprocessed time series T as $\sigma_\Delta(T|\mathcal{X}_{j-1})$ (line 21) and proceeds to the next iteration. While `GreedyET` does not improve the worst-case time complexity of `Greedy`, it is empirically much more efficient than `Greedy`. Moreover, it maintains the same theoretical guarantee as `Greedy`.

3.3 `PreGreedy` and `PreGreedyET` Methods

For any given similarity threshold τ, both `Greedy` and `GreedyET` require computing similar sets to calculate the marginal coverage of time series. However, this computation typically dominates running time and becomes a bottleneck for the `Greedy` and `GreedyET` selection processes. This limitation motivates us to further accelerate the computation time by precomputing the similar sets for each time series, thereby avoiding repeated computations in each greedy iteration and significantly reducing overall running time. Instead of computing the similar sets on-the-fly during the selection process (as in `Greedy` and `GreedyET`), we precompute them before the selection process (as in `PreGreedy` and `PreGreedyET`). This involves

constructing a lookup table to store similar set S_T for each time series T prior to selection (before line 3 of Algorithm 1 and line 4 of Algorithm 2). This table enables direct use of similar sets during the greedy selection process without the need to compute them on-the-fly (line 6 of Algorithm 1 and lines 7 and 14 of Algorithm 2).

3.4 Complexity Analysis

Time Complexity. Computing similar set for a single time series on-the-fly requires $O(|\mathcal{T}|)$ time, so the total worst-case time complexity for both `Greedy` and `GreedyET` is $O(\beta|\mathcal{T}|^3)$. On the other hand, constructing a lookup table for S_T for each T requires $O(|\mathcal{T}|^2)$ time. Hence, the total worst-case time complexity for both `PreGreedy` and `PreGreedyET` is $O(|\mathcal{T}|^2 + \beta|\mathcal{T}|^2)$.

Space Complexity. Computing similar sets on-the-fly in each greedy iteration requires $O(|\mathcal{T}|)$ space, which is the worst-case space complexity for both `Greedy` and `GreedyET`. On the other hand, constructing a lookup table for S_T for each T requires $O(|\mathcal{T}|^2)$ space, which is the total worst-case space complexity for both `PreGreedy` and `PreGreedyET`.

4 LEARNING-BASED REPRESENTATIVE TIME SERIES SELECTION

Limitations of Non-learning based Methods. While our non-learning-based methods leverage the monotonicity and submodularity of our objective function to produce solutions with an approximation ratio, the challenge lies in the computation of marginal coverage of time series. The memory-intensive process of precomputing similar sets and the time-consuming nature of computing similar sets on-the-fly pose limitations on the scalability for all variants of Greedy selection methods. The primary bottleneck is the computation of similar sets, which is essential for computing the exact marginal coverage of time series. Adding to the challenge, specifying a new similarity threshold τ requires to recompute of similar sets. Currently, there is a lack of an approach that is effective, efficient and scalable simultaneously.

Given the aforementioned limitations, we develop a novel time and memory efficient approach that accurately predicts the exact marginal coverage of time series for representative time series selection. Specifically, we propose a self-supervised learning approach known as `MLGreedyET`. `MLGreedyET` aims to maintain high accuracy in estimating the exact marginal coverage while accommodating any specified similarity threshold τ without requiring expensive computation of similar sets. By incorporating greedy algorithm design into both the training and selection phases, `MLGreedyET` is able to offer an effective solution that simultaneously addresses memory and time issues.

At a high level, `MLGreedyET` involves three primary processes. Firstly, it prepares high-quality data by generating informative features based on the `GreedyET` framework. Next, the self-supervised learning model leverages these informative features and learns a function to estimate the marginal coverage $\hat{\sigma_\Delta}(T|\mathcal{X}_i)$, which approximates the exact marginal coverage $\sigma_\Delta(T|\mathcal{X}_i)$ as described in Equation 1. Finally, it searches and selects the representative time series based on the `GreedyET` framework. Unlike all variants of the Greedy selection method, it utilizes the model-estimated marginal coverage rather than the exact marginal coverage to select the representative time series, hence addressing the limitations previously outlined. In the subsequent sections, we will delve into each of these processes, starting by describing *how the learning function facilitates representative time series selection* (Section 4.1), followed by the *training workflow* (Section 4.2) and the *model architecture* (Section 4.3). Then, we will conduct a *time and space complexity analysis* (Section 4.4).

4.1 `MLGreedyET` Method

In `MLGreedyET`, we introduce a learning function f. The objective of f is to predict the value $\hat{\sigma_\Delta}(T_i|\mathcal{X}_{i-1})$ as an approximation of the exact marginal coverage $\sigma_\Delta(T_i^c|\mathcal{X}_{i-1})$. Using f, `MLGreedyET` solves the RTSD problem using the `GreedyET` framework in Section 3.2. The representative time series selection process of `MLGreedyET` is similar to that of `GreedyET`. The only difference is that `MLGreedyET` employs f to predict $\hat{\sigma_\Delta}(T_i^c|\mathcal{X}_{i-1})$ rather than computing the costly exact marginal coverage $\sigma_\Delta(T_i^c|\mathcal{X}_{i-1})$ for each candidate time series T_i^c in each iteration i in `GreedyET`. In Algorithm 2, we replace the exact marginal coverage computations $|S_T|$ and $|S_T \cap U|$ in lines 11 and 18 with f. We explain how to derive the input features for f in Section 4.2.2. By replacing the time-consuming exact marginal coverage computation used in `Greedy` and `GreedyET`, and by avoiding the memory-intensive precomputation of similar sets in `PreGreedy` and `PreGreedyET`, our learning function f effectively addresses the bottleneck in computing the exact marginal coverage. Furthermore, f can predict $\hat{\sigma_\Delta}(T_i^c|\mathcal{X}_{i-1})$ for any given τ without the need for retraining, thus saving time and memory. Next, we will describe the training workflow and model architecture for f.

4.2 Training Workflow

To train a model capable of accurately estimating the marginal coverage of time series, it is essential to generate high-quality training samples containing useful and informative features.

4.2.1 Embedding Generation. Due to the large dimensionality of raw time series data, it is crucial to condense the data into a more manageable form to alleviate computational overhead. To achieve this, we must extract only necessary features and latent information from the high-dimensional raw time series, resulting in reduced-dimensional embeddings conducive to model training. Among numerous time series representation learning methods that have been proposed [32, 36, 51], we utilize SEAnet [51] to generate embeddings of dimension 16, following their default settings, for our raw input time series data. This choice is motivated by the fact that embeddings generated by SEAnet are optimized for similarity search and have been shown to better preserve original pairwise similarities in the lower-dimensional embedded space. This advantage translates to more accurate approximate similarity searches and aids our model in learning the similarities between time series within their embeddings.

4.2.2 Feature extraction. We present six input features of our data that will be fed into our model:

(1) Aggregated embeddings of the last selected representative time series $AGG(EMB(\mathcal{X}_{i-1})) \in \mathbb{R}^{16}$.

(2) Aggregated embeddings of the last residual time series $AGG(EMB(\overline{\mathcal{X}_{i-1}^{E_i^c}})) \in \mathbb{R}^{16}$.
(3) Embedding of the current candidate time series $E_i^c \in \mathbb{R}^{16}$.
(4) Count of last selected representative time series $|\mathcal{X}_{i-1}|$.
(5) Count of last residual time series $|\overline{\mathcal{X}_{i-1}^{E_i^c}}|$.
(6) Similarity threshold $\tau \in [0, 1]$.

Here, i denotes the current iteration, AGG denotes the aggregate function and EMB denotes the embedding. The last residual time series refer to the unselected time series, excluding the current candidate time series, i.e., $\overline{\mathcal{X}_{i-1}^{E_i^c}} = \mathcal{T} \setminus (\mathcal{X}_{i-1} \cup \{E_i^c\})$. The candidate time series $T_i^c \in \mathcal{T} \setminus \mathcal{X}_{i-1}$ is the time series for which we predict its exact marginal coverage $\sigma_\Delta(T_i^c|\mathcal{X}_{i-1})$, with its corresponding embedding denoted as E_i^c. We normalize the similarity threshold τ to ensure it remains within a consistent and reasonable range of 0 and 1 by dividing it by the maximum pairwise similarity among $\mathcal{T}$.

Rationale behind the choice of features extracted. The value of exact marginal coverage $\sigma_\Delta(T_i^c|\mathcal{X}_{i-1})$ for T_i^c varies depending on $\mathcal{X}_{i-1}$ and τ. Therefore, it is crucial to include the aggregated embeddings of $\mathcal{X}_{i-1}$, the embedding of T_i^c and the τ as input features of the training sample. Additionally, we incorporate the aggregated embeddings of $\overline{\mathcal{X}_{i-1}^{E_i^c}}$ to enable the model to capture the overall information of the remaining time series in $\mathcal{T}$. We also include the counts of $\mathcal{X}_{i-1}$ and $\overline{\mathcal{X}_{i-1}^{E_i^c}}$ to provide the model with information regarding their sizes, complementing their aggregated embeddings. Finally, these six input features are concatenated to form an input vector $Z_{(\mathcal{X}_{i-1}, T_i^c, \tau)} \in \mathbb{R}^{51}$, which is then scaled for training. The target label for training is the exact marginal coverage $\sigma_\Delta(T_i^c|\mathcal{X}_{i-1})$ of the candidate time series T_i^c.

Aggregate function. Due to the varying sizes of selected representative time series and residual time series, aggregating them into a fixed-size vector is essential for our model to process them as inputs. For the choice of AGG, we have considered mean and sum aggregate function. We opt for the sum aggregate function due to its greater expressive power [57], which enables it to preserve information about collective features across different embeddings. This method ensures that similar features in the resulting aggregation remain distinguishable. Mean aggregation tends to incur more information loss, particularly when discriminative features exist within individual embeddings, potentially leading to the cancellation of contrasting features across different embeddings.

4.2.3 Data generation. Given the vast space of potential training samples, we exploit the `GreedyET` framework to reduce the space to generate high-quality training samples. The data generation procedure is outlined in Procedure 3. Initially, we execute `GreedyET`, where in each iteration i, we explore a relatively small pool of T_i^c with high $\sigma_\Delta(T_i^c|\mathcal{X}_{i-1})$. From this pool, we include the T_i^c with the highest $\sigma_\Delta(T_i^c|\mathcal{X}_{i-1})$ in $\mathcal{X}_\rangle$. Concurrently, non-candidate T_i with low $\sigma_\Delta(T_i|\mathcal{X}_{i-1})$ are pruned through early termination (line 24 of Procedure 3). For each T_i^c in the i-th iteration, we extract their features and concatenate them to create a training sample alongside their corresponding $\sigma_\Delta(T_i^c|\mathcal{X}_{i-1})$ (line 21).

Procedure 3: Data Generation

```
1  foreach T ∈ 𝒯 do
2  |  Generate embedding E for T;
3  U ← 𝒯;
4  𝒳 ← ∅;
5  TrainSet ← ∅; // store pairs of features and labels
6  PQ ← an empty priority queue that sorts time series by their upper
   |  bound (ub) of marginal coverage in non-increasing order;
7  while U ≠ ∅ do
8  |  if 𝒳 = ∅ then
9  |  |  foreach T ∈ 𝒯 do
10 |  |  |  Insert T into PQ with T.ub = |S_T|;
11 |  |  T ← PQ.pop();
12 |  |  Compute (Z_(𝒳,T,τ), σ_Δ(T|𝒳)) and add to TrainSet;
13 |  else
14 |  |  max_coverage = 0;
15 |  |  while |PQ| > 0 do
16 |  |  |  T ← PQ.pop();
17 |  |  |  T.ub = |S_T ∩ U|;
18 |  |  |  if T.ub > max_coverage then
19 |  |  |  |  max_coverage = T.ub;
20 |  |  |  |  T^r = T;
21 |  |  |  Compute (Z_(𝒳,T,τ), σ_Δ(T|𝒳)) and add to TrainSet;
22 |  |  |  τ' ← increase or decrease τ;
23 |  |  |  Compute (Z_(𝒳,T,τ'), σ_Δ(T|𝒳)) and add to TrainSet;
24 |  |  |  if |PQ| > 0 and max_coverage ≥ PQ[0].ub then
25 |  |  |  |  Sample some remaining candidates T' using
   |  |  |  |  |  K-means++;
26 |  |  |  |  Compute (Z_(𝒳,T',τ), σ_Δ(T'|𝒳)) and add to
   |  |  |  |  |  TrainSet;
27 |  |  |  |  Compute (Z_(𝒳,T',τ'), σ_Δ(T'|𝒳)) add add to
   |  |  |  |  |  TrainSet;
28 |  |  |  |  break;
29 |  |  |  Update PQ with visited T excluding T^r;
30 |  |  𝒳 ← 𝒳 ∪ {T^r};
31 |  |  U ← U \ S_T^r;
32 return TrainSet;
```

Enhanced data quality. To ensure a balanced distribution of the target label, it is necessary to include training samples with low exact marginal coverage. Since the pruned T_i with low $\sigma_\Delta(T_i|\mathcal{X}_{i-1})$ typically constitute a significant portion, effective sampling of a portion of them as training samples is essential. Drawing inspiration from K-means clustering [37, 41], we employ the advanced clustering initialization strategy, K-means++ [6] to sample a small portion of pruned T_i, along with their corresponding $\sigma_\Delta(T_i|\mathcal{X}_{i-1})$ to form the additional training samples (lines 25-26 of Procedure 3). K-means++ operates by sampling T_i such that each new sample is chosen with a probability proportional to its squared similarity from the closest existing sampled T_i. This method ensures that the sampled T_i are evenly distributed in terms of their similarities, thereby contributing to a more balanced distribution of training data and enhancing the generalization of the model. In addition to this strategy, for each training sample generated, we augment them by slightly adjusting the feature τ to a neighboring value within an appropriate range of ±0.1, while keeping the remaining features fixed (lines 22-23 and 27). This augmentation introduces

Procedure 4: Model Training

```
while not converged do
    Shuffle TrainSet;
    foreach batch ∈ TrainSet do
        loss ← 0;
        foreach (Z_(X_{i-1},T_i,τ), σ_Δ(T_i|X_{i-1})) ∈ batch do
            σ̂_Δ(T_i|X_{i-1}) ← f(Z_(X_{i-1},T_i,τ));
            loss ← loss + |σ_Δ(T_i|X_{i-1}) − σ̂_Δ(T_i|X_{i-1})|;
        Minimize loss / |batch|;
```

more variety into the training data, providing the model with contrasting samples to better understand the effects of τ and improve its generalization capabilities.

4.3 Model Architecture

Next, our learning function f utilizes the features $Z_{(\mathcal{X}_{i-1},T_i,\tau)}$ to estimate the value of marginal coverage $\hat{\sigma_\Delta}(T_i|\mathcal{X}_{i-1})$. It is implemented using a fully connected multi-layer neural network optimized with mini-batch stochastic gradient descent, which has demonstrated high effectiveness in our experiments. The network consists of five layers, including the input layer, three hidden layers and the output layer. Formally, f is defined as

$$\begin{aligned} f(Z_{(\mathcal{X}_{i-1},T_i,\tau)}) &= W_4 \cdot \mathrm{ReLU}(W_3 \cdot \mathrm{ReLU}(W_2 \cdot \mathrm{ReLU}(\\ &\quad W_1 \cdot Z_{(\mathcal{X}_i,T,\tau)} + b_1) + b_2) + b_3) + b_4 \\ &= \hat{\sigma_\Delta}(T_i|\mathcal{X}_{i-1}) \end{aligned} \tag{2}$$

where ReLU denotes the Rectified Linear Unit activation function [24, 25], $W_1 \in \mathbb{R}^{51\times D_1}$, $W_2 \in \mathbb{R}^{D_1\times D_2}$, $W_3 \in \mathbb{R}^{D_2\times D_3}$, $W_4 \in \mathbb{R}^{D_3\times 1}$, $b_1 \in \mathbb{R}^{D_1}$, $b_2 \in \mathbb{R}^{D_2}$, $b_3 \in \mathbb{R}^{D_3}$ and $b_4 \in \mathbb{R}^1$ are learnable parameters. We use mean absolute error as our loss function $\mathcal{L}$. Given a set N of training pairs, where each pair consists of the ground truth exact marginal coverage $\sigma_\Delta(T_i|\mathcal{X}_{i-1})$ and predicted marginal coverage $\hat{\sigma_\Delta}(T_i|\mathcal{X}_{i-1})$, our goal is to minimize the mean absolute error across all training pairs, i.e.,

$$\mathcal{L} = \frac{\sum_{(\sigma_\Delta(T_i|\mathcal{X}_{i-1}),\hat{\sigma_\Delta}(T_i|\mathcal{X}_{i-1}))\in N} |\sigma_\Delta(T_i|\mathcal{X}_{i-1}) - \hat{\sigma_\Delta}(T_i|\mathcal{X}_{i-1})|}{|N|} \tag{3}$$

The training procedure is outlined in Procedure 4.

4.4 Complexity Analysis

Time Complexity. For each candidate time series T_i^c, predicting the value for marginal coverage $\hat{\sigma_\Delta}(T_i^c|\mathcal{X}_{i-1})$ takes constant $O(|Z_{(\mathcal{X}_{i-1},T_i^c,\tau)}|)$ time. At the end of each iteration i, we compute the exact marginal coverage $\sigma_\Delta(T_i^r|\mathcal{X}_{i-1})$ of the representative time series selected T_i^r to check for termination condition, taking $O(|\mathcal{T}|)$ time. Thus, each iteration takes $O(|Z_{(\mathcal{X}_{i-1},T_i^c,\tau)}|\cdot|\mathcal{T}|+|\mathcal{T}|) = O(|\mathcal{T}|)$ time and the total worst-case time complexity for `MLGreedyET` is $O(\beta|\mathcal{T}|^2)$.

Space Complexity. Apart from the model parameters, predicting the value for marginal coverage $\hat{\sigma_\Delta}(T_i^c|\mathcal{X}_{i-1})$ takes constant $O(|Z_{(\mathcal{X}_{i-1},T_i^c,\tau)}|)$ space. At the end of each iteration i, we compute the exact marginal coverage $\sigma_\Delta(T_i^r|\mathcal{X}_{i-1})$ of the representative time series selected T_i^r to check for termination condition, taking $O(|\mathcal{T}|)$ space. Since we generate an embedding $E \in \mathcal{E}$ for each time series, the total worst-case space complexity for `MLGreedyET` is $O(|\mathcal{E}| + |\mathcal{T}|)$.

Table 3: Statistics summary of the datasets (# denotes number and TS stands for time series).

Dataset	# of TS	# of sampled TS	TS length
ECG [30]	97M	10K	320
Seismic [23]	100M	50K	256
SALD [47]	200M	100K	128
DEEP [49]	1B	150K	96
METR-LA [34]	207	-	288

5 EXPERIMENT

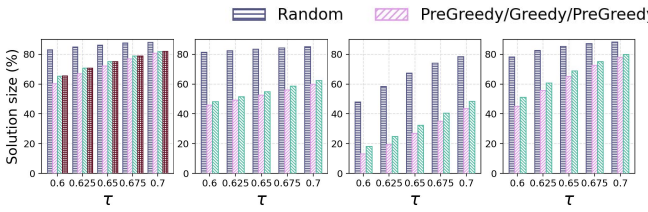
We conduct extensive ablation experiments to evaluate the performance of our proposed non-learning-based and learning-based methods (Section 5.2). Our objective is to demonstrate that our methods can achieve high effectiveness while ensuring outstanding efficiency, low memory consumption, or a balanced combination of both. While time series clustering [2, 35] and object diversification [14, 48] address different problems, they share commonalities with our problem. Therefore, we also compare our methods with theirs to evaluate the performance (Section 5.3). In addition, we present a visualization case study to illustrate the practicality of our method in real-world scenarios (Section 5.4).

5.1 Experimental Setup

Datasets. To demonstrate the generality of our solutions, we conduct experiments (Section 5.2 and 5.3) on four large-scale real-world datasets from diverse domains and they are widely used in time series similarity search [18, 19, 51, 55]: ECG [30] from electrocardiography, Seismic [23] from seismology, SALD [47] from neuroscience and DEEP [49] from image processing. Due to the computational complexity in terms of time and memory for computing similar sets in the non-learning-based Greedy variants, we uniformly sample 10K, 50K, 100K and 150K time series from these datasets respectively for evaluation purposes. We also conduct a visualization case study (Section 5.4) on the METR-LA dataset [34], which comprises traffic speed readings collected from 207 sensors at 5-minute intervals along the highways of Los Angeles County on May 1st, 2012. The statistics of the datasets are summarized in Table 3.

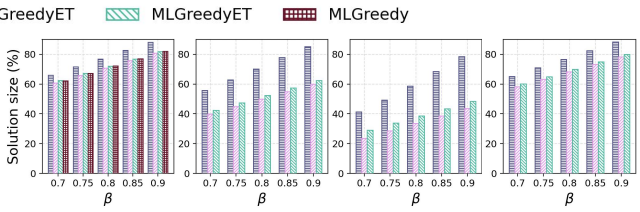
Methods For Comparison. For our ablation experiments, we compare the following variants of the `Greedy` approach with a random selection method as a baseline. We have excluded `Greedy` from this comparison due to its extended runtime, exceeding a day even on our smallest dataset.

- `Random`: Randomly selects time series as representatives until they cover at least β proportion of the entire time series database.
- `PreGreedy` (Section 3.3): Our non-learning based method that precomputes similar sets and selects representative time series based on exact marginal coverage using the greedy algorithm.
- `PreGreedyET` (Section 3.3): Our enhanced version of `PreGreedy` featuring the early termination technique.
- `GreedyET` (Section 3.2): Similar to `PreGreedyET`, but without precomputing similar sets.



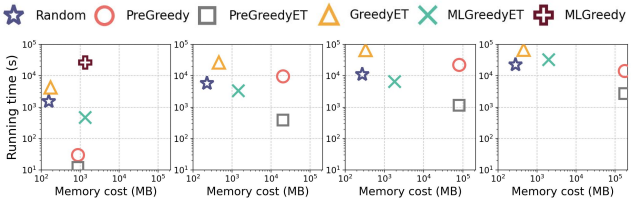
(a) ECG (b) Seismic (c) SALD (d) DEEP

Figure 3: Solution size by each method across different similarity thresholds τ.



(a) ECG (b) Seismic (c) SALD (d) DEEP

Figure 4: Solution size by each method across different coverage thresholds β.



(a) ECG (b) Seismic (c) SALD (d) DEEP

Figure 5: Trade-off between running time and memory cost for each method.

- `MLGreedyET` (Section 4): Our learning-based method that selects representative time series based on our model's predicted marginal coverage using the greedy algorithm with early termination.
- `MLGreedy`: Similar to `MLGreedyET`, but without the early termination technique.

For comparison with existing works from time series clustering and diversification, we include the following approaches.

- `k-medoids` [31]: A clustering algorithm that partitions a set of time series into k clusters by minimizing the sum of dissimilarities between time series and their nearest medoid.
- `k-shape` [45]: A partitional clustering algorithm with shape-based distance measure that preserves the shapes of time series.
- `DTCR` [40]: An unsupervised temporal representation learning approach optimized for k-means time series clustering.
- `MaxMin` [16]: A diversification method that selects items by maximizing the minimum distance between selected items.
- `MaxSum` [16]: A diversification method that selects items by maximizing the sum of distance between selected items.
- `DisC` [15]: A diversification algorithm that selects a subset of diversified items to cover the entire dataset.

Evaluation Metrics. We evaluate the performance of the methods using the following metrics.

- *Solution size* is calculated as the percentage of representative time series selected over the total number of time series. Since our objective is to minimize the number of representative time series selected, a smaller size or lower percentage reflects greater effectiveness, indicating that the data can be represented with fewer representatives. This is critical for system usage because it reduces similar data and facilitates analysis by focusing on the smaller representative set of time series.
- *Total coverage* is calculated as the percentage of time series covered by the selected representatives over the total number of time series. Higher coverage is preferable, as it indicates that a greater number of time series are represented by the selected representatives, ensuring fewer time series are overlooked and left unrepresented.
- *Time efficiency* is evaluated by measuring the running time of the method.
- *Memory footprint* is evaluated by measuring the memory consumption of the method.

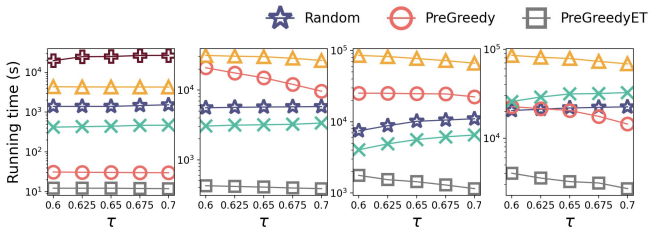
Parameter Settings. All models are trained for 300 epochs with three hidden layer sizes of 128, 64, and 16 respectively, and a batch size of 256. The training/validation split is 80:20. For testing, we normalize the similarity threshold $\tau \in [0, 1]$ and vary it with values of {0.6, 0.625, 0.65, 0.675, 0.7} and the coverage threshold β with values of {0.7, 0.75, 0.8, 0.85, 0.9}, where the underlined values are the default settings. These values span a wide range of representative time series selected and facilitate a comprehensive evaluation of the method across different scenarios.

Environment. All experiments are conducted on a Linux server with Intel Xeon E5 CPUs, 512GB RAM and Tesla P100 PCIe 16GB. The code, implemented in Python and PyTorch, is available at [1].

5.2 Ablation Experimental Results

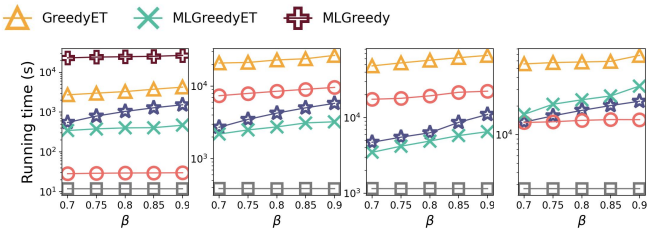
Exp 1 - Effectiveness Comparison. Figures 3 and 4 compare the solution size, i.e., the percentage of representative time series selected over the total number of time series, by each method across different similarity thresholds τ and coverage thresholds β respectively. We present all non-learning-based Greedy variants in one bar since they yield the same result. Since our objective is to minimize the number of representative time series selected, a smaller size or lower percentage indicates greater effectiveness. The results show that `MLGreedyET` not only consistently outperforms the `Random` baseline, but also competes effectively with non-learning-based Greedy variants. Specifically, it achieves a percentage difference as low as 1% in the number of representatives selected when compared with the non-learning-based Greedy variants on ECG at $\tau = 0.7$ and $\beta = 0.9$. Although `MLGreedy` achieves similar performance to `MLGreedyET`, it lacks the early termination feature. As a result, it can only process the smallest ECG dataset, while runtimes for larger datasets exceed a day. This limitation is further reflected in Exp 2 and 3.

Exp 2 - Time-Memory Trade-off Comparison. Figure 5 shows the trade-off between running time and memory cost for each method. Ideally, a method should minimize both factors. Notably, `GreedyET` demonstrates remarkably low memory consumption, while `PreGreedy` and `PreGreedyET` excel in terms of running time. In contrast, `MLGreedyET` strikes a favorable balance between running time and memory cost, as shown across Seismic, SALD and



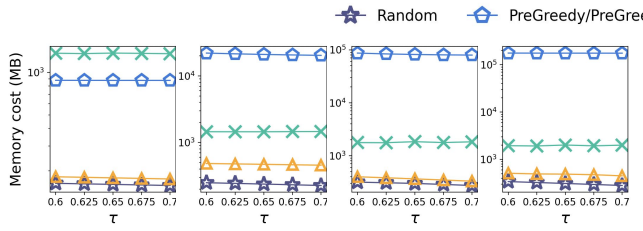
(a) ECG (b) Seismic (c) SALD (d) DEEP

Figure 6: Running time comparison for each method across different similarity thresholds τ.



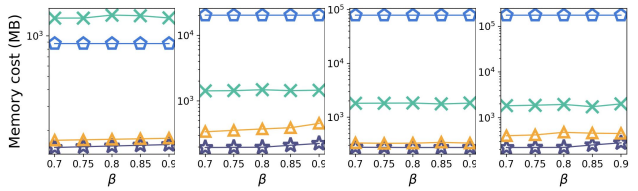
(a) ECG (b) Seismic (c) SALD (d) DEEP

Figure 7: Running time comparison for each method across different coverage thresholds β.



(a) ECG (b) Seismic (c) SALD (d) DEEP

Figure 8: Memory cost comparison for each method across different similarity thresholds τ.



(a) ECG (b) Seismic (c) SALD (d) DEEP

Figure 9: Memory cost comparison for each method across different coverage thresholds β.

DEEP. However, MLGreedyET may not be the best choice for situations where minimizing either running time or memory cost is prioritized over balancing both. The memory space required by MLGreedyET primarily depends on its model parameters, which remain consistent across datasets. Consequently, for small datasets such as ECG, the space required by model parameters may be dominant, but the scalability advantage of MLGreedyET becomes evident with larger datasets.

Exp 3 - Efficiency Comparison. Figures 6 and 7 compare the running time of each method across different similarity thresholds τ and coverage thresholds β. We observe that: (1) without precomputing similar sets, MLGreedyET shows high efficiency while ensuring effectiveness, achieving up to 21× speedups over GreedyET when compared on SALD at $\tau = 0.6$ and $\beta = 0.9$. This demonstrates the advantage of predicting the marginal gain over computing the exact marginal gain; (2) MLGreedyET is 47× to 67× faster than MLGreedy on ECG. Additionally, PreGreedyET is 212× to 9196× faster than PreGreedy in selecting time series, but the total running time of both methods are limited by the precomputation of similar sets. This highlights the superiority of the early termination technique introduced in PreGreedyET, GreedyET and MLGreedyET.

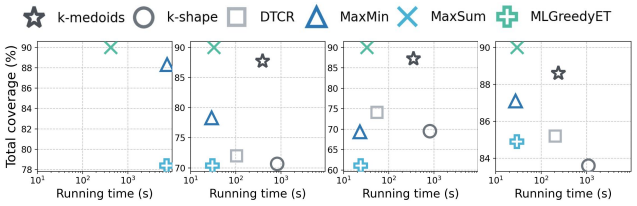
Exp 4 - Memory Cost Comparison. Figures 8 and 9 compare the memory consumption of each method across different similarity thresholds τ and coverage thresholds β. Based on these two figures, both PreGreedy and PreGreedyET incur memory costs primarily due to the precomputation of similar sets, which notably increases as the dataset scale grows. Furthermore, the memory cost for precomputing similar sets escalates with lower similarity thresholds τ, as the size of similar sets increases. On the other hand, MLGreedyET exhibits consistent memory costs across varying τ values and saves up to 101× memory space when compared to PreGreedy and PreGreedyET on DEEP at $\tau = 0.7$ and $\beta = 0.85$, demonstrating its high scalability.

5.3 Effectiveness Study on Extensions from Clustering and Diversification

Comparison with Time Series Clustering. It is important to note that direct comparison between time series clustering and our methods may be challenging due to the differing problems and constraints we address, as detailed in Sections 1 and 6. Despite these differences, we compare the total coverage of the selected representative time series and running time between our methods and clustering. Specifically, we compare our method with k-medoids [31], as it is one of the popular clustering methods and its medoids are actual time series, which we treat as our representative time series in this case. We also compare our method with DTCR [40], a recent representation learning approach optimized for clustering and k-shape [45], a well-known approach for time series clustering. DTCR applies k-means clustering to its learned cluster-specific temporal representations generated by its unsupervised learning model, while k-shape introduces a shape-based distance measure with a tailored centroid computation method. These methods are considered as variants of k-means clustering.

To ensure a fair comparison with our method, we make the following adjustments. (1) We set the number of clusters k equal to the number of representative time series selected by MLGreedyET, since the optimal value for k is unknown. (2) Unlike k-medoids, where the cluster centers are actual time series, the centers in DTCR and k-shape are not actual. To address this, we select the time series nearest to each cluster center as the representative (for k-shape, we use its proposed shape-based distance measure). In cases of empty clusters, we select the next nearest unselected time series as representative. (3) We determine the coverage of the selected time series by its similar set rather than its cluster.

We compare the total coverage of representative time series selected with the running time of clustering methods and MLGreedyET and show the results in Figure 10. In our smallest dataset ECG,



(a) ECG (b) Seismic-1K (c) SALD-1K (d) DEEP-1K

Figure 10: Comparison of total coverage against running time for different methods with similarity threshold $\tau = 0.6$ and coverage threshold $\beta = 0.9$.

`k-medoids`, `k-shape` and `DTCR` exceeded our time limit (over a day) due to large dataset size and high value of k. Therefore, we uniformly downsampled the remaining datasets to 1K time series to mitigate the extended runtimes of these methods. Based on the figure, we observe that `MLGreedyET` (along with all our proposed methods) guarantees full coverage of at least β proportion of the total number of time series (e.g., at least 90% of time series are covered if $\beta = 0.9$) for any similarity thresholds τ. Additionally, it is the fastest among all clustering methods (e.g., 7× to 37× faster on DEEP-1K). In contrast, the representatives selected by the clustering methods fail to fully cover and represent the required coverage of β in each dataset (e.g., `k-shape` covers only 69.5% of the 90% required on SALD-1K). Since the cluster centers in `k-shape` and `DTCR` are not actual time series, empty cluster may occur. In such cases, we select the next nearest unselected time series as representative. This adjustment likely explains why the coverage results for `k-shape` and `DTCR` tend to be worse than `k-medoids`, thus making direct comparisons to our method less reliable.

Comparison with Object Diversification. Similar to clustering, direct comparison with diversification may pose challenges due to the inherent differences in the problems and constraints, as outlined in Sections 1 and 6. However, to evaluate the effectiveness of our method against diversification, we compare the solution size, i.e., the percentage of representative time series selected to fully cover the required coverage of β. Specifically, we choose `DisC` [15] for comparison because its problem definition is close to ours, as it also requires selected items to fully cover the data. We apply `DisC` to time series data and treat the diversified items as representatives. We incorporate the key constraint of `DisC` into our methods, both non-learning-based Greedy variants (represented by `Greedy` as they produce the same result) and the learning-based method `MLGreedyET`. This constraint ensures that the similarity between any pair of representative time series selected is less than the similarity threshold τ. We refer to the modified methods as `Greedy-DisC` and `MLGreedyET-DisC` respectively. Table 4 shows the solution size of `Greedy` and `MLGreedyET` along with their respective diversification counterparts, `Greedy-DisC` and `MLGreedyET-DisC`. The results indicate that our methods, `Greedy` and `MLGreedyET`, have a lower percentage of representative time series selected compared to `Greedy-DisC` and `MLGreedyET-DisC`. This suggests that our methods are more effective, as they require fewer representatives to cover and represent the given data.

In addition to `DisC`, we also include two widely used fundamental models for diversification, `MaxMin` and `MaxSum` [16] for comparison. `MaxMin` and `MaxSum` aim to select a subset of diverse items,

Table 4: Solution size (%) with similarity threshold $\tau = 0.6$ and coverage threshold $\beta = 0.9$.

Methods	ECG	Seismic	SALD	DEEP
`Greedy-DisC`	61.11	49.87	17.38	48.91
`Greedy`	**60.40**	**45.88**	**13.19**	**45.00**
`MLGreedyET-DisC`	66.15	52.37	20.32	52.07
`MLGreedyET`	**65.56**	**48.12**	**18.26**	**51.15**

such that either the minimum or the sum of pairwise distances between selected items is maximized. More formally, the objective functions of `MaxMin` and `MaxSum` are defined as $f_{MIN}(S) = \max_{\substack{S \subseteq O \\ |S|=k}} \min_{\substack{o_i, o_j \in S \\ o_i \neq o_j}} (1 - SIM(o_i, o_j))$ and $f_{SUM}(S) = \max_{\substack{S \subseteq O \\ |S|=k}} \sum_{\substack{o_i, o_j \in S \\ o_i \neq o_j}} (1 - SIM(o_i, o_j))$. Both `MaxMin` and `MaxSum` address the k-Diversity problem [16], which involves selecting a fixed number k of diverse items. In our context, we treat each selected item as a representative time series. Since k-Diversity problem is known to be NP-hard [20], we use greedy heuristics for their implementation, which have been shown to produce good solutions [14]. We then evaluate the total coverage of the representative time series selected with the running time of both `MaxMin` and `MaxSum`. Similar to how we handle clustering methods, (1) we set k equal to the number of representative time series selected by `MLGreedyET`, and (2) we determine the coverage of a selected time series by its similar set. The results are reported in Figure 10, where we see that despite comparable running times with `MLGreedyET` on the downsampled datasets, both diversification methods fail to fully cover the required coverage β (e.g., `MaxSum` covers only 61.1% of the 90% required on SALD-1K), indicating their coverage limitations compared to `MLGreedyET`.

5.4 Visualization Case Study

We present a visualization case study employing our `MLGreedyET` to select representative time series from METR-LA, as illustrated in Figure 11. We set the similarity threshold $\tau = 0.85$ to obtain a sufficiently small size for clear visualization, and the coverage threshold $\beta = 1$ to ensure every sensor is covered by a representative so that no traffic patterns are overlooked. We observe that the representative time series selected by `MLGreedyET` are distinct from one another and exhibit various interesting patterns across the sensors. The minimum number of representative time series selected also ensures that the representative traffic patterns stand out and clearly summarize the given sensors, providing a comprehensive and distinct overview of the traffic patterns without redundancy. For instance, in Figure 11b, the representative time series and its time series represented indicate consistently fast traffic flow throughout the day. In contrast, Figures 11c and 11d show heavy traffic during the morning and evening peaks respectively. Each representative only represents traffic patterns that meet the similarity threshold, and they collectively cover and represent different traffic patterns recorded by each sensor in the region.

6 RELATED WORK

In this section, we review the literature on time series clustering and object diversification, which are related to our RTSD problem. Time series clustering identifies cohesive groups of similar

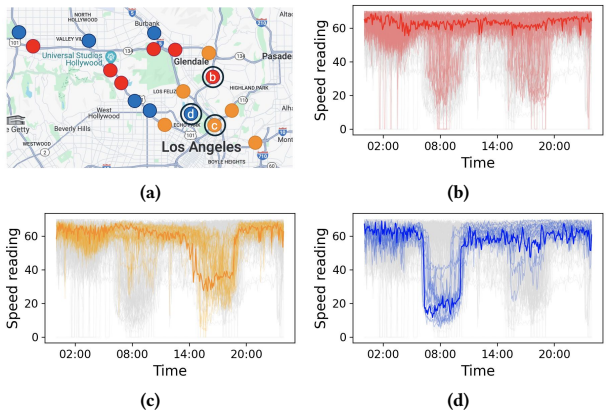


Figure 11: (a) shows sensors on road segments, each colored by their representative traffic patterns in (b), (c) and (d). Representative time series are shown in thick lines, with corresponding sensors circled on the map. Time series represented by these representatives are shown in thinner lines of the same color, corresponding to the uncircled sensors. Remaining time series in gray are those not covered by the representatives. To reduce visual clutter, only 5 sensors represented by each representative are shown, e.g., Sensor b represents other sensors in red since their patterns are highly similar.

time series, while object diversification selects diverse objects to enhance the quality of query results for users. These approaches differ fundamentally from RTSD problem, which we discuss next.

Time Series Clustering. Time series clustering is a well-researched topic with extensive literature in the field [2, 5, 8, 13, 21, 27, 29, 35, 56]. Its objective is to maximize data similarity within clusters while minimizing it across clusters. This unsupervised technique groups similar time series into clusters based on certain patterns or features. Partitioning clustering divides time series into k clusters by grouping similar time series together. Popular algorithms such as k-means [41], k-medoids [31], and Fuzzy C-means [7] iteratively compute centroids and assign time series to their nearest centroid until termination. k-shape [45] is a well-known time series clustering method that introduces a shape-based distance measure with a tailored centroid computation method. DTCR [40] is a recent representation learning approach that applies k-means clustering to cluster-specific temporal representations generated by its unsupervised learning model. Additionally, recent studies on time series clustering tend to focus on specific contexts, such as clustering multivariate time series data [8] or incomplete time series data [39], which differ from our problem setting.

While the cluster center may be loosely perceived as a representative time series, several differences exist compared to our problem. Our representative time series are similarity-bounded, ensuring that the similarity between the representative time series and the time series it represents is at least the user-defined threshold. This allows users to control the similarity threshold and, hence, the representativeness of each representative. Moreover, under this property, we ensure that every time series is covered and represented by at least one of the minimum number of representative time series. This stands in contrast to clustering, as we cannot guarantee that the similarity between time series and its centroids within a cluster meets the threshold, and centroids may not fully cover and represent all time series. In particular, clustering algorithms that require specifying the number of clusters pose more challenges, as it is difficult to determine the minimum number of clusters such that the centroids can fully cover all time series. Additionally, centroids are typically not actual time series (e.g., the mean of all feature vectors within a cluster in k-means clustering), which does not align with our problem.

Object Diversification. Object diversification aims to select a subset of diverse objects to ensure a wide representation of different aspects within the data [48, 60]. It is a broad topic that covers a wide range of domains and data types such as search query results [48], images retrieval [38] and training data for neural machine translation [43]. However, we found no existing diversification methods specifically tailored to time series data. Various definitions of diversity have been proposed [14], focusing on the content or similarity [60], the novelty [11], and the semantic coverage [3] of the objects. Diversification based on the content or similarity of the objects is particularly relevant to our RTSD problem. For instance, MaxMin [16] and MaxSum [10] select a subset of objects such that the minimum or the sum of pairwise distances among the selected objects is maximized. DisC [15] selects a subset of diverse objects with a dissimilarity constraint to cover the entire result set.

Similar to clustering, if we consider an object as a time series, the objects selected by algorithms like MaxMin and MaxSum are not similarity-bounded and, thus the selected objects are not guaranteed to fully cover the required proportion of the result set. Additionally, these algorithms require the number of objects to be selected as input, which poses similar challenges as in clustering. In the case of DisC, the dissimilarity constraint mandates that the distance between selected objects must exceed a user-defined distance. This may result in more objects being selected and does not guarantee the minimum number of representatives to be selected as defined in our problem.

7 CONCLUSION AND FUTURE WORK

In this paper, we introduce the problem of finding the smallest set of representative time series to summarize a dataset. We prove the problem to be NP-hard and propose several greedy variants to approximate the optimal solution, but they fail to ensure both time and memory efficiency. Therefore, we introduce a self-supervised learning approach integrated with greedy algorithm to yield effective solutions while ensuring efficiency and low memory consumption. Extensive experiments on four real-world datasets demonstrate the effectiveness, efficiency and scalability of our method. We also present a visualization case study to highlight its practical value. While our methods show strong performance, future work will focus on scaling to larger datasets, reducing human interpretation of the representatives and exploring integration with large language models by feeding representatives into prompts to evaluate its potential for enhancing time series analysis tasks.

ACKNOWLEDGMENT

This project is supported in part by ARC DP240101211 and FT240100832. Ge Lee is a recipient of the Data61 top-up scholarship.

REFERENCES

[1] 2024. Source code. https://github.com/rmitbggroup/RTSD.

[2] Saeed Aghabozorgi, Ali Seyed Shirkhorshidi, and Teh Ying Wah. 2015. Time-series clustering–a decade review. *Information systems* 53 (2015), 16–38.

[3] Rakesh Agrawal, Sreenivas Gollapudi, Alan Halverson, and Samuel Ieong. 2009. Diversifying search results. In *Proceedings of the second ACM international conference on web search and data mining*. 5–14.

[4] Wolfgang Aigner, Silvia Miksch, Heidrun Schumann, and Christian Tominski. 2011. *Visualization of time-oriented data*. Vol. 4. Springer.

[5] Ali Alqahtani, Mohammed Ali, Xianghua Xie, and Mark W Jones. 2021. Deep time-series clustering: A review. *Electronics* 10, 23 (2021), 3001.

[6] David Arthur, Sergei Vassilvitskii, et al. 2007. k-means++: The advantages of careful seeding. In *Soda*, Vol. 7. 1027–1035.

[7] James C Bezdek. 2013. *Pattern recognition with fuzzy objective function algorithms*. Springer Science & Business Media.

[8] Angela Bonifati, Francesco Del Buono, Francesco Guerra, and Donato Tiano. 2022. Time2Feat: learning interpretable representations for multivariate time series clustering. *Proceedings of the VLDB Endowment (PVLDB)* 16, 2 (2022), 193–201.

[9] Paul Boniol, John Paparrizos, Themis Palpanas, and Michael J Franklin. 2021. SAND: streaming subsequence anomaly detection. *Proceedings of the VLDB Endowment* 14, 10 (2021), 1717–1729.

[10] Allan Borodin, Aadhar Jain, Hyun Chul Lee, and Yuli Ye. 2017. Max-sum diversification, monotone submodular functions, and dynamic updates. *ACM Transactions on Algorithms (TALG)* 13, 3 (2017), 1–25.

[11] Charles LA Clarke, Maheedhar Kolla, Gordon V Cormack, Olga Vechtomova, Azin Ashkan, Stefan Büttcher, and Ian MacKinnon. 2008. Novelty and diversity in information retrieval evaluation. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*. 659–666.

[12] Hui Ding, Goce Trajcevski, Peter Scheuermann, Xiaoyue Wang, and Eamonn Keogh. 2008. Querying and mining of time series data: experimental comparison of representations and distance measures. *Proceedings of the VLDB Endowment* 1, 2 (2008), 1542–1552.

[13] Rui Ding, Qiang Wang, Yingnong Dang, Qiang Fu, Haidong Zhang, and Dongmei Zhang. 2015. Yading: Fast clustering of large-scale time series data. *Proceedings of the VLDB Endowment* 8, 5 (2015), 473–484.

[14] Marina Drosou and Evaggelia Pitoura. 2010. Search result diversification. *ACM SIGMOD Record* 39, 1 (2010), 41–47.

[15] Marina Drosou and Evaggelia Pitoura. 2012. DisC Diversity: Result Diversification based on Dissimilarity and Coverage. *Proceedings of the VLDB Endowment* 6, 1 (2012).

[16] Marina Drosou and Evaggelia Pitoura. 2013. Diverse set selection over dynamic data. *IEEE Transactions on Knowledge and Data Engineering* 26, 5 (2013), 1102–1116.

[17] Karima Echihabi, Kostas Zoumpatianos, and Themis Palpanas. 2020. Big sequence management: on scalability. *IEEE BigData* (2020).

[18] Karima Echihabi, Kostas Zoumpatianos, Themis Palpanas, and Houda Benbrahim. 2018. The Lernaean Hydra of Data Series Similarity Search: An Experimental Evaluation of the State of the Art. *Proceedings of the VLDB Endowment* 12, 2 (2018), 112–127.

[19] Karima Echihabi, Kostas Zoumpatianos, Themis Palpanas, and Houda Benbrahim. 2019. Return of the Lernaean Hydra: Experimental Evaluation of Data Series Approximate Similarity Search. *Proceedings of the VLDB Endowment* 13, 3 (2019), 403–420.

[20] Erhan Erkut. 1990. The discrete p-dispersion problem. *European Journal of Operational Research* 46, 1 (1990), 48–60.

[21] Philippe Esling and Carlos Agon. 2012. Time-series data mining. *ACM Computing Surveys (CSUR)* 45, 1 (2012), 1–34.

[22] Christos Faloutsos, Mudumbai Ranganathan, and Yannis Manolopoulos. 1994. Fast subsequence matching in time-series databases. *ACM Sigmod Record* 23, 2 (1994), 419–429.

[23] I. R. I. for Seismology with Artificial Intelligence. 2018. Seismic Data Access. http://ds.iris.edu/data/access/.

[24] Kunihiko Fukushima. 1975. Cognitron: A self-organizing multilayered neural network. *Biological cybernetics* 20, 3-4 (1975), 121–136.

[25] Kunihiko Fukushima. 1980. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics* 36, 4 (1980), 193–202.

[26] Vehbi C Gungor and Gerhard P Hancke. 2009. Industrial wireless sensor networks: Challenges, design principles, and technical approaches. *IEEE Transactions on industrial electronics* 56, 10 (2009), 4258–4265.

[27] Dino Ienco and Roberto Interdonato. 2020. Deep multivariate time series embedding clustering via attentive-gated autoencoder. In *Advances in Knowledge Discovery and Data Mining: 24th Pacific-Asia Conference, PAKDD 2020, Singapore, May 11–14, 2020, Proceedings, Part I 24*. Springer, 318–329.

[28] Hassan Ismail Fawaz, Benjamin Lucas, Germain Forestier, Charlotte Pelletier, Daniel F Schmidt, Jonathan Weber, Geoffrey I Webb, Lhassane Idoumghar, Pierre-Alain Muller, and François Petitjean. 2020. Inceptiontime: Finding alexnet for time series classification. *Data Mining and Knowledge Discovery* 34, 6 (2020), 1936–1962.

[29] Ali Javed, Byung Suk Lee, and Donna M Rizzo. 2020. A benchmark study on time series clustering. *Machine Learning with Applications* 1 (2020), 100001.

[30] Alistair EW Johnson, Tom J Pollard, Lu Shen, Li-wei H Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. 2016. MIMIC-III, a freely accessible critical care database. *Scientific data* 3, 1 (2016), 1–9.

[31] Leonard Kaufman and Peter J Rousseeuw. 2009. *Finding groups in data: an introduction to cluster analysis*. John Wiley & Sons.

[32] Eamonn Keogh, Kaushik Chakrabarti, Michael Pazzani, and Sharad Mehrotra. 2001. Dimensionality reduction for fast similarity search in large time series databases. *Knowledge and information Systems* 3 (2001), 263–286.

[33] Jure Leskovec, Andreas Krause, Carlos Guestrin, Christos Faloutsos, Jeanne VanBriesen, and Natalie Glance. 2007. Cost-effective outbreak detection in networks. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*. 420–429.

[34] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. 2018. Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting. In *International Conference on Learning Representations*.

[35] T Warren Liao. 2005. Clustering of time series data—a survey. *Pattern recognition* 38, 11 (2005), 1857–1874.

[36] Jessica Lin, Eamonn Keogh, Li Wei, and Stefano Lonardi. 2007. Experiencing SAX: a novel symbolic representation of time series. *Data Mining and knowledge discovery* 15 (2007), 107–144.

[37] Stuart Lloyd. 1982. Least squares quantization in PCM. *IEEE transactions on information theory* 28, 2 (1982), 129–137.

[38] Wei Lu, Mengqi Luo, Zhenyu Zhang, Guobiao Zhang, Heng Ding, Haihua Chen, and Jiangping Chen. 2019. Result diversification in image retrieval based on semantic distance. *Information Sciences* 502 (2019), 59–75.

[39] Qianli Ma, Chuxin Chen, Sen Li, and Garrison W Cottrell. 2021. Learning Representations for Incomplete Time Series Clustering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 8837–8846.

[40] Qianli Ma, Jiawei Zheng, Sen Li, and Gary W Cottrell. 2019. Learning representations for time series clustering. *Advances in neural information processing systems* 32 (2019).

[41] James MacQueen et al. 1967. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, Vol. 1. Oakland, CA, USA, 281–297.

[42] Shigeru Masuyama, Toshihide Ibaraki, and Toshiharu Hasegawa. 1981. The computational complexity of the m-center problems on the plane. *IEICE TRANSACTIONS (1976-1990)* 64, 2 (1981), 57–64.

[43] Xuan-Phi Nguyen, Shafiq Joty, Kui Wu, and Ai Ti Aw. 2020. Data diversification: A simple strategy for neural machine translation. *Advances in Neural Information Processing Systems* 33 (2020), 10018–10029.

[44] Themis Palpanas. 2015. Data series management: The road to big sequence analytics. *ACM SIGMOD Record* 44, 2 (2015), 47–52.

[45] John Paparrizos and Luis Gravano. 2015. k-shape: Efficient and accurate clustering of time series. In *Proceedings of the 2015 ACM SIGMOD international conference on management of data*. 1855–1870.

[46] Jin Shieh and Eamonn Keogh. 2008. i SAX: indexing and mining terabyte sized time series. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. 623–631.

[47] Southwest University. 2018. Southwest University Adult Lifespan Dataset (SALD). http://fcon_1000.projects.nitrc.org/indi/retro/sald.html.

[48] Erik Vee, Utkarsh Srivastava, Jayavel Shanmugasundaram, Prashant Bhat, and Sihem Amer Yahia. 2008. Efficient computation of diverse query results. In *2008 IEEE 24th International Conference on Data Engineering*. IEEE, 228–236.

[49] Skoltech Computer Vision. 2018. Deep billion-scale indexing. http://sites.skoltech.ru/compvision/noimi.

[50] Dakuo Wang, Justin D Weisz, Michael Muller, Parikshit Ram, Werner Geyer, Casey Dugan, Yla Tausczik, Horst Samulowitz, and Alexander Gray. 2019. Human-AI collaboration in data science: Exploring data scientists' perceptions of automated AI. *Proceedings of the ACM on human-computer interaction* 3, CSCW (2019), 1–24.

[51] Qitong Wang and Themis Palpanas. 2021. Deep learning embeddings for data series similarity search. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*. 1708–1716.

[52] Tingting Wang, Shixun Huang, Zhifeng Bao, J Shane Culpepper, and Reza Arablouei. 2022. Representative Routes Discovery from Massive Trajectories. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 4059–4069.

[53] Xiaoyue Wang, Abdullah Mueen, Hui Ding, Goce Trajcevski, Peter Scheuermann, and Eamonn Keogh. 2013. Experimental comparison of representation methods and distance measures for time series data. *Data Mining and Knowledge Discovery* 26 (2013), 275–309.

[54] Yunhai Wang, Yuchun Wang, Xin Chen, Yue Zhao, Fan Zhang, Eugene Wu, Chi-Wing Fu, and Xiaohui Yu. 2023. OM3: An Ordered Multi-level Min-Max Representation for Interactive Progressive Visualization of Time Series. *Proceedings of the ACM on Management of Data* 1, 2 (2023), 1–24.

[55] Zeyu Wang, Qitong Wang, Peng Wang, Themis Palpanas, and Wei Wang. 2023. Dumpy: A compact and adaptive index for large data series collections. *Proceedings of the ACM on Management of Data* 1, 1 (2023), 1–27.

[56] Chenxiao Xu, Hao Huang, and Shinjae Yoo. 2021. A deep neural network for multivariate time series clustering with result interpretation. In *2021 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 1–8.

[57] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2018. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826* (2018).

[58] M Adil Yalçin, Niklas Elmqvist, and Benjamin B Bederson. 2016. Cognitive stages in visual data exploration. In *Proceedings of the Sixth Workshop on Beyond Time and Errors on Novel Evaluation Methods for Visualization*. 86–95.

[59] Yue Zhao, Yunhai Wang, Jian Zhang, Chi-Wing Fu, Mingliang Xu, and Dominik Moritz. 2021. KD-Box: Line-segment-based KD-tree for interactive exploration of large-scale time-series data. *IEEE Transactions on Visualization and Computer Graphics* 28, 1 (2021), 890–900.

[60] Cai-Nicolas Ziegler, Sean M McNee, Joseph A Konstan, and Georg Lausen. 2005. Improving recommendation lists through topic diversification. In *Proceedings of the 14th international conference on World Wide Web*. 22–32.