

PSEUDO: Interactive Pattern Search in Multivariate Time Series with Locality-Sensitive Hashing and Relevance Feedback

Yuncong Yu, Dylan Kruffy, Jiao Jiao, Tim Becker, and Michael Behrisch

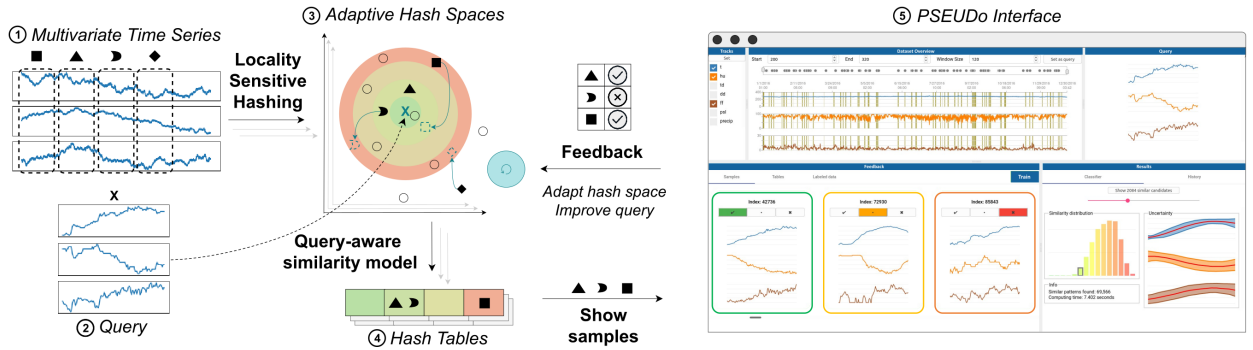


Fig. 1. PSEUDO creates a representation model for multivariate time series based on locality-sensitive hashing, conducts scalable pattern retrieval with few initial labels, and evolves with interpretable relevance feedback to capture subjective pattern similarity.

Abstract— We present PSEUDO, a visual pattern retrieval tool for multivariate time series. It aims to overcome the uneconomic (re-)training problem accompanying deep learning-based methods. Very high-dimensional time series emerge on an unprecedented scale due to increasing sensor usage and data storage. Visual pattern search is one of the most frequent tasks on time series. Automatic pattern retrieval methods often suffer from inefficient training data, a lack of ground truth labels, and a discrepancy between the similarity perceived by the algorithm and required by the user or the task. Our proposal is based on the query-aware locality-sensitive hashing technique to create a representation of multivariate time series windows. It features sub-linear training and inference time with respect to data dimensions. This performance gain allows an instantaneous relevance-feedback-driven adaption to converge to users' similarity notion. We demonstrate PSEUDO's performance in terms of accuracy, speed, steerability, and usability through quantitative benchmarks with representative time series retrieval methods and a case study. We find that PSEUDO detects patterns in high-dimensional time series efficiently, improves the result with relevance feedback through feature selection, and allows an understandable as well as user-friendly retrieval process.

Index Terms—time series, pattern search, locality-sensitive hashing, relevance feedback

1 INTRODUCTION

Searching for patterns similar to a given query in a time series database is one of the most frequent problems in time series analysis [38]. In the literature, it is called pattern search [28, 36], time series indexing [12], similarity search [22, 47], query by content, sub-sequence matching [19], and twin search [21]. It is an abstraction of many real-world problems, e.g., identifying brightness transients relating to astronomical objects like supernovae or quasars [35], searching for regulatory elements in genomic sequences [36], tracking recurrent events in data collected from inertial measurement units in mobile devices [34], and detecting similar behaviors in stock price [19, 39, 47]. It remains an interesting and important question to efficiently and accurately search for patterns in unlabeled multivariate time series. Our automotive engineers search for patterns spontaneously in measurement from various sensors and control units and wish for an answer

as promptly as possible. This task is challenging, not only because of the high dimensions (a large number of tracks), meager labels, and the efficiency requirement, but also the subjective and use-case-dependent similarity notion. Whereas model-free similarity measures lack trainable parameters and the power to model potentially complex similarity rules catering to the user's similarity notion, machine learning may suffer from insufficient labels and inefficient training. Furthermore, our application engineers ask for an interpretable process, i.e., which tracks count most for the event behind the pattern, to assist the subsequent domain-specific analysis.

We propose PSEUDO (Pattern Search, Exploration and Understanding in multivariate time series Data), a tool for visual pattern retrieval in Multivariate Time Series (MTS), especially very high-dimensional time series. It is powered by Locality-Sensitive Hashing (LSH) [60]. In a nutshell, LSH linearly maps all tracks into one with groups of hash functions, making subsequent processing scalable with respect to the data dimensions. Our major contribution is making LSH trainable and extending it with an efficient, steerable, and interpretable relevance feedback mechanism. Researchers have also introduced relevance feedback for tabular [6], text [57], and image data [15]. It is first introduced to time series data in [31] and appears recently in [36]. Finally, we implemented a prototypical user interface to assist the algorithm. Such UIs for time series retrieval are often called Visual Query Systems (VQSs) [35, 36, 55].

As shown in Fig. 1, the overall pipeline works as follows: 1) preprocessing the time series with sliding windows and window normalization, ① in Fig. 1; 2) marking by the user a pattern in the time series as the

- Yuncong Yu is with Utrecht University and IAV GmbH. E-mail: yuncong.yu@outlook.com.
- Dylan Kruffy is with NAVARA. E-mail: dylankruffy@gmail.com.
- Jiao Jiao is with Utrecht University and Fraunhofer ISI. E-mail: jiao.jiao@isi.fraunhofer.de.
- Tim Becker is with IAV GmbH. E-mail: tim.becker@iav.de.
- Michael Behrisch is with Utrecht University. E-mail: m.behrisch@uu.nl.

Manuscript received 31 March 2022; revised 1 July 2022; accepted 8 August 2022.
Date of publication 28 September 2022; date of current version 2 December 2022.
Digital Object Identifier no. 10.1109/TVCG.2022.3209431

Authorized licensed use limited to: University Library Utrecht. Downloaded on September 29, 2023 at 11:18:42 UTC from IEEE Xplore. Restrictions apply.

1077-2626 © 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See <https://www.ieee.org/publications/rights/index.html> for more information.

query searched for in the time series database, ②; 3) initial search based on the standard LSH, from ①② to ④; 4) sampling results for relevance feedback, from ④ to ⑤; 5) inspecting results and providing relevance feedback by the user, ⑤; 6) updating the LSH model and rerunning search, from ⑤ back to ③; 7) iterating the steps 4) to 6) until the user is satisfied with the result.

We benchmarked its “open-loop” accuracy and speed without relevance feedback against four representative benchmarking methods on four labeled datasets with different characteristics. Disappointingly, we found that the methods work differently on different datasets, and there is no universally best method in terms of accuracy. In the speed benchmark, PSEUDO is slightly worse than Mueen’s Algorithm for Similarity Search (MASS) (the state-of-the-art tool for time series pattern search in terms of speed) on univariate and low-dimensional datasets. Still, it starts to overtake the latter on a relatively high-dimensional dataset. To verify this finding, we benchmarked the scalability with respect to the number of tracks with an unlabeled very high-dimensional dataset. Next, we evaluate the effectiveness of the relevance feedback mechanism, where we use an agent simulating sensible user feedback as the experimental group and two agents simulating extreme user feedback as control groups. We witnessed an increasing accuracy within five feedback rounds, which happened instantaneously. Furthermore, we noticed that PSEUDO attaches higher weights to informative tracks and lower to less helpful ones, confirming its feature selection mechanism. Finally, we invited an expert to test PSEUDO in his use case.

2 REQUIREMENT ANALYSIS

After discussion with our engineers, we collected the following major requirements and analyzed them.

R1: Quick response. The engineers wish for an answer as promptly as possible. This requirement is also necessary if the query system is interactive or uses active learning. Consequently, optimization-based model training is not preferred. On the one hand, MASS suggests that Euclidean Distance (ED) is the fastest time series similarity measure [43, 59]. On the other hand, LSH excels in dealing with high-dimensional time series [60]. Therefore, we use LSH as feature representation and ED as the similarity/distance measure.

R2: Adaptive similarity measure. The interpretation of “similarity” is subject to the use case. For instance, one engineer wants to find a signal related to the start of the engine in CAN Bus data. Depending on the task, sometimes, he needs to search for all such signals and sometimes only a specific subcategory that is particularly similar to the query. However, he can begin with the same query. We deal with the problem by introducing a relevance feedback mechanism to the LSH model. According to **R1**, this mechanism should also be efficient.

R3: Interpretable process. A typical task of our calibration engineers is spontaneously tracing a signal related to an anomaly. Locating the recurrence of the target signal is the first step. After that, they need to analyze the root of the problem. It often means finding the tracks that have a causal relationship with the target signal. Accordingly, we introduced a feature selection-based feedback mechanism.

R4: Accurate retrieval. This self-explanatory requirement often conflicts with **R1** and sometimes **R3**. It is unfeasible to achieve the best performance from every aspect, and we choose to be defensive at this point. Accuracy depends mainly on the representation (LSH) and the similarity measure (ED). While [60] has examined LSH’s accuracy sacrifice, we verified it with our own experiments. As for the similarity measure, Bagnall et al. benchmarked 20 methods and found Dynamic Time Warping (DTW) hard to beat for time series classification [4]. Nonetheless, the result of the quicker ED tends to converge to that of DTW, providing more data [5]. These findings consolidate our choice of ED as the primary similarity measure.

Please note that we postpone related work to Sect. 6 to avoid interrupting the information flow.

3 RELEVANCE-FEEDBACK-DRIVEN LOCALITY-SENSITIVE HASHING

This section describes PSEUDO’s relevance-feedback-driven learning algorithm based on LSH. LSH allows querying large MTS efficiently

for real-time user interaction. We extended it with an also efficient and interpretable relevance feedback algorithm.

We derive our conceptual model, depicted in Fig. 2, from FDive, the feedback-driven preference learning method by Behrisch et al. [6]. Inspired by the conceptual model, we assume that the randomly initialized parameters in the LSH hash functions are trainable to combat the ambiguity of time series similarity.

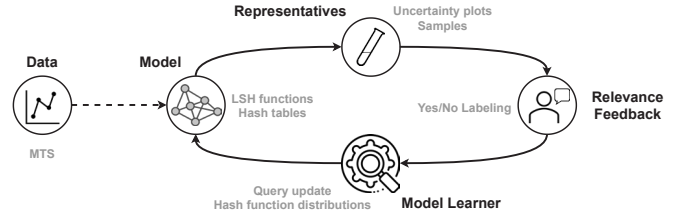


Fig. 2. **Conceptual Model:** PSEUDO is (1) an instantiation of FDive [6] for MTS retrieval. (2) It models data with LSH, (3) samples the outcomes, (4) invites the user to review the model’s similarity understanding, and (5) optimizes the LSH functions based on user feedback.

In the conceptual model, input MTS data are modeled with LSH functions and stored in query-aware hash tables, significantly speeding up the processing. Striving for a sensible data modeling on top of LSH’s probabilistic nature and converging to the subjective similarity, we draw a sample set from the hashed time series. The user then checks the samples as well as their “average” shape and variance per bucket. This mechanism actively contributes to the querying process in that PSEUDO learns to understand the user’s similarity notion and interprets it as feature/track importance. Due to the size and complexity of MTS data, such an open and adaptable exploration process was unfeasible before within our self-imposed performance limits. Moreover, we can see that such a user-in-the-loop active learning approach improves the overall retrieval performance and remains explainable.

For a detailed description, we introduce the following notations. We use lower case letters for scalars and upper case for matrices; i for time steps, j for track indices, and r for feedback rounds; vectors are denoted with arrows, like \vec{a} ; complex element-wise operations vectors or matrices are denoted with their entries like $\{a_j\}$; we use \mathbb{C} to denote sampled candidates, with $\mathbb{C}^{+/-}$ being positively or negatively labeled items. Next, we will explain how to transform the conceptual model into the concrete PSEUDO pipeline in Fig. 3.

3.1 Initial Modeling

The initial modeling follows the procedures proposed in [60]. As Fig. 3 shows, the input data include a query $Q = \{\vec{q}_i \in \mathbb{R}^d\}$, $i = 1, 2, \dots, t$ and a d -dimensional MTS $S = \{\vec{s}_i \in \mathbb{R}^d\}$, $i = 1, 2, \dots, n$, of length n . The output data are the time series windows filtered by LSH and their similarity to Q . Because the method is extensive and established, we describe only the essentials in this section.

As the red box in Fig. 3 indicates, the method preprocesses S by partitioning it with a sliding window of size t (length of Q). Optionally, we can use a range of differently sized sliding windows for patterns with variable duration. Subsequently, the windows and the query are normalized and prepared for hashing.

In the next step, LSH initializes l compound hash functions. Each compound hash function consists of k hash functions $h(\vec{x})$, $\vec{x} \in \mathbb{R}^d$. Each hash function h is independently initialized with a vector $\vec{a} = \{a_j\} \in \mathbb{R}^d$ containing d elements drawn independently from the standard normal distribution ($a_j \sim \mathcal{N}(0, 1)$).

Fig. 4 illustrates the details during the modeling process. Each h calculates dot product between its \vec{a} and every time step \vec{x}_i in a time series window $X = \{\vec{x}_i\}$, thus merging d tracks in X to a univariate hash code of length t . A projection collision for \vec{x}_i happens when $|h(\vec{q}_i) - h(\vec{x}_i)| \leq \frac{\omega}{2}$ holds with a given error band ω . Further, a hash collision for X happens when the number of projection collisions between the query Q and X under an h exceeds a threshold t_s . A group hash collision for X

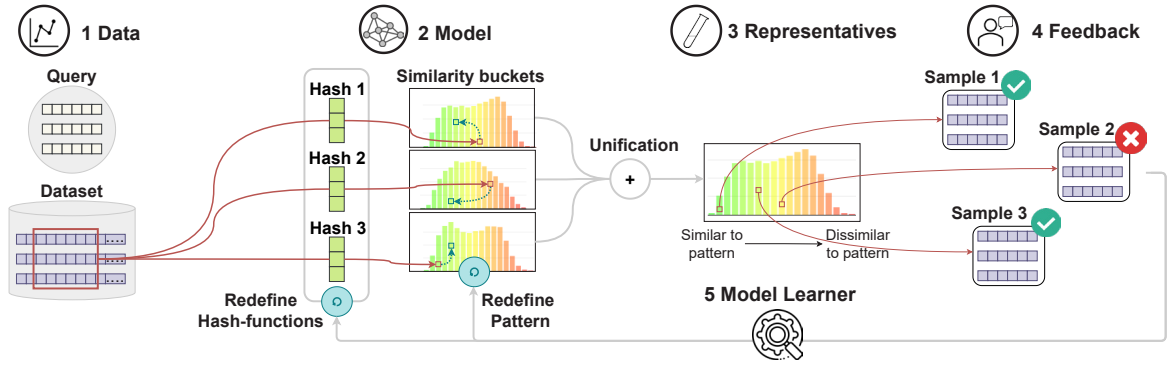


Fig. 3. **PSEUDO Processing Pipeline:** (1) PSEUDO receives a query and preprocessed time series windows as inputs, (2) hash them into hash table buckets representing the distribution of similarity to the query (green depicts similar, red dissimilar), (3) draw representatives from both similar and dissimilar buckets, (4) receive user feedback, (5) and update the hash functions and the query pattern accordingly.

happens when X and Q have hash collisions under all k hash functions in a compound hash function. Finally, X is considered a candidate similar to Q when X and Q have compound hash collision under at least one of the l compound hash functions. How l , k , and other parameters can be optimally set is referred to [60].

In the next step, the pipeline uses a similarity measure like ED or DTW to calculate the similarity of all candidates to the query. PSEUDO modifies the MTS modeling scheme [60] by applying the similarity measure on the hash code of the candidates rather than the time series windows in this stage. This modification has two advantages: (1) the complexity with respect to the number of tracks is reduced from linear to constant; (2) the similarity measure is based on a representation from LSH, which is updated later, reflecting the user's emphasis rather than pure algorithmic. Although the model building appears complex, it can be computed instantaneously and allows rapid adaption.

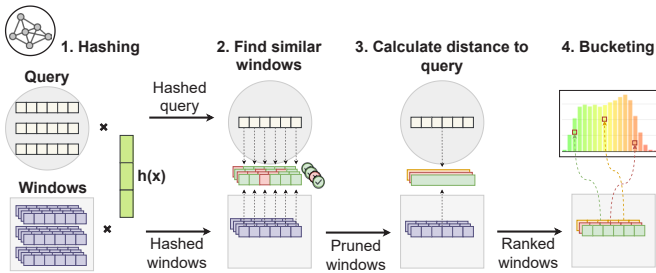


Fig. 4. **Modeling Process (detail):** (1) PSEUDO hashes all multivariate time series windows including the query to univariate hash codes, (2) prunes the windows outside the hash bucket containing the query, (3) rank the similarity of the remaining windows to the query based on a similarity measure like ED or DTW.





Retrieval Invariances: Our modeling scheme allows us to account for various retrieval invariances. While the sliding window-based preprocessing covers horizontal translational invariance (translation along the time axis) (Fig. 4.1), it fails to capture efficiently. Vertical translational invariance and scaling invariance (amplitude scaling and bias in the y-axis) are handled by normalizing all windows in the first step (Fig. 4.1). If necessary, the distortion invariance, i.e., time shifts in the pattern, is approached with an elastic measure like DTW.

3.2 Relevance Feedback

We achieve the model steering in our conceptual model (Fig. 2) through the representative selection and relevance feedback steps. They allow users to incorporate their domain expertise and guide how the model should alter its current state. This process has two components: (1) it is inherently a visual-interactive (interface) problem and will therefore be elaborated on in Sect. 4, and (2) we need to decide which *representative*

samples to show the user. In PSEUDO, users can give feedback on samples and hash tables.

Relevance Feedback on Candidate Samples: We draw samples from the candidates (windows surviving filtering by LSH as explained in Sect. 3.1) and invite the user to label them. The number of candidates can be large, even after filtering with LSH. Users can not process more than possibly a few dozen without being stuck in a tedious labeling process. Following the central idea of (visual) active learning [3, 15, 17], we choose representatives based on the trade-off between exploitation and exploration. On the exploitation side, PSEUDO includes the top-5 candidates with the highest similarity in the samples. However, if we restrict ourselves to top hits, the learning process tends to reinforce the current knowledge while refusing to learn something new. Hence, we also draw five random samples from the candidates that may not have top similarity scores.

Relevance Feedback on Hash Tables: Besides predicted candidate samples, we visualize the hash tables and allow the user to provide feedback on them. The details are kept in Sect. 4.2. As mentioned, the whole LSH model uses a histogram for similarity ranking. We represent the individual hash table likewise. The histogram visualizes the similarity distribution perceived by a hash table. We interpret the commonly occurring histogram patterns in the following manner: a positively skewed shape  points to a hash function, which observes many windows similar to the query, while a negatively skewed shape  means the opposite. Unitary  or bi-modal distributions  refer to undecided or decisive hash functions.

3.3 Updating Model

Conventional LSH does not contain trainable parameters and cannot be optimized directly by relevance feedback. However, the randomly initialized parameters in the hash functions do not have to persist. As described in Sect. 3.1, a hash function h performs dot product $\vec{a} \cdot \vec{x}_i$ with its hashing vector \vec{a} and i -th time step in a window $X = \{\vec{x}_i \in \mathbb{R}^d\}$. This operation can be interpreted as a weighted merge of d tracks in X with the track weights in \vec{a} .

To capture relevance feedback, we propose to modify \vec{a} with a learned weighting vector \vec{w} and use the vector $\vec{b} = \vec{w} \odot \vec{a}$ in place of \vec{a} , where \odot denotes element-wise product.

To avoid vanishing or exploding parameters, we want to retain the expectation of the magnitude of \vec{a} , namely $E(\|\vec{b}\|) = E(\|\vec{a}\|)$. It is achieved by normalizing the magnitude of \vec{w} to \sqrt{d} . We normalize \vec{w} instead of \vec{b} directly because there are multiple \vec{b} but only one \vec{w} .

Recall that $a_j \sim \mathcal{N}(0, 1)$, where a_j is the j -th element of \vec{a} corresponding to the j -th track. The squared magnitude of \vec{a} is

$$\begin{aligned} E(\|\vec{a}\|^2) &= E\left(\sum_{j=1}^d a_j^2\right) = \sum_{j=1}^d E(a_j^2) = \sum_{j=1}^d E((a_j - 0)^2) \\ &= \sum_{j=1}^d E((a_j - E(a_j))^2) = \sum_{j=1}^d \text{Var}(a_j) = \sum_{j=1}^d 1 = d \end{aligned}$$

Likewise, since $b_j = w_j \cdot a_j \sim \mathcal{N}(0, w_j^2)$, it follows

$$E(\|\vec{b}\|^2) = \sum_{j=1}^d \text{Var}(b_j) = \sum_{j=1}^d w_j^2 = E(\|\vec{w}\|^2)$$

Aiming at $E(\|\vec{b}\|^2) = E(\|\vec{a}\|^2)$, we set

$$E(\|\vec{w}\|^2) = E(\|\vec{b}\|^2) = E(\|\vec{a}\|^2) = d$$

As a result, we need to normalize $\|\vec{w}\|$ to \sqrt{d} .

PSEUDO allows two types of relevance feedback, namely, feedback on the candidate samples and the hash tables. Accordingly, we maintain two weight vectors \vec{w}_s and \vec{w}_h for samples and hash tables respectively.

Sample Relevance Adaption: The feedback on the positively labeled samples, $\mathbb{C}^+ = \{C_{pos}\}$, is transformed to track importance. We calculate the DTW distances or ED between the tracks of the positively labeled samples and that of the query Q . Let \vec{q}_j be the j -th track of Q , \vec{c}_j the j -th track of C_{pos} . We define $\vec{z}_j = \sum_{C_{pos} \in \mathbb{C}^+} DTW(\vec{q}_j, \vec{c}_j)$ as the aggregate distance between \vec{c}_j and \vec{q}_j . Next, we normalize the entries in \vec{z}_j between $[0, 1]$ with $\vec{z}^* = \{z_j^*\} = \{z_j / \sum_{j=1}^d z_j\}$, then convert distance to its negatively correlated weight vector yields $\vec{w}_s^* = \{1 - z_j^*\}$, which is subsequently normalized to $\vec{w}_s = \vec{w}_s^* \frac{\sqrt{d}}{\|\vec{w}_s^*\|}$.

Hash Table Relevance Adaption: The feedback on the hash tables can be implemented likewise. Let $A = \{\vec{a}_{pos}\}$ be the parameter vectors of all hash functions labeled as positive by the user. Then, we can define $\vec{a}_h = \{a_{h,j}\} = \{\sum_{\vec{a}_{pos} \in A} a_{pos,j}^2\}$. After normalization to magnitude \sqrt{d} we get $\vec{w}_h = \vec{a}_h \frac{\sqrt{d}}{\|\vec{a}_h\|}$.

Finally, we merge $\vec{w}_{s,r}$, $\vec{w}_{h,r}$ in r -th feedback round and \vec{w}_{r-1} into \vec{w}_r through a linear combination with a learning rate α , which can be gradually modified to enforce exploration stability [6].

$$\begin{aligned} \vec{w}_r^* &= (1 - \alpha)\vec{w}_{r-1} + \frac{\alpha}{2}(\vec{w}_{s,r} + \vec{w}_{h,r}) \\ \vec{w}_r &= \vec{w}_r^* \frac{\sqrt{d}}{\|\vec{w}_r^*\|} \end{aligned}$$

Query Adaption: As shown by the backward arrow in Fig. 3, we update not only the LSH hash functions but also the query because the initial query does not necessarily represent the generally desired shape of the pattern. In each training iteration, we integrate Q (search query) and \mathbb{C}^+ (set of positively labeled samples). This operation is known to be non-trivial [18] and impacts future exploration direction. After the trial with the naive element-wise average yields unsatisfactory results, because the average often resembles none of the original windows, we decided for Dynamic Time Warping Barycenter Averaging (DBA) [50], as it takes distortion and time shift during averaging into consideration [18, 20, 50].

4 USER INTERFACE

We designed PSEUDO's visual interface depicted in Fig. 5. It comprises five interlinked views facilitating data exploration, query definition, process monitoring, result inspection, relevance feedback, and state management. A REST API connects the web-based user interface to PSEUDO's backend algorithm. The backend (in Python and C++) and frontend (with Angular) code are available under <https://git.science.uu.nl/vig/sublinear-algorithms-for-va/pseudo>. The video <https://www.youtube.com/watch?v=0JfXoDyZRPY> demonstrates the user interface as well as a concise workflow.




4.1 Dataset Overview, Track View, and Query View.



The *Dataset Overview*, *Tracks View*, and *Query View* collaborate closely. The *Dataset Overview* (Fig. 5.b) plots tracks selected in the *Tracks View* (Fig. 5.a) together with the window labels (dots Fig. 5.b3 in the range slider Fig. 5.b1 and marked intervals in the line chart Fig. 5.b2). Besides direct panning and zooming, a range slider (Fig. 5.b1) above the track curves (Fig. 5.b2) serves as a mini-map for navigation and an overview of predictions and labels.


The *Query View* (Fig. 5.c) shows the user-defined multivariate query. The user defines the query in a query-by-example manner by selecting a region in the *Dataset Overview*. The user can change the query on the fly, but PSEUDO must repeat the hashing process whenever the query size varies. Typically users have to wait several seconds for LSH parameter estimation. While not focused on in this project, our query definition interface currently lacks the option to define patterns shifted across tracks. We plan to investigate better query interfaces for MTS data following recent examples, such as [11].

4.2 Feedback View

The *Feedback View* (Fig. 5.c) shows representatives of all predictions, visualizes hash tables, and keeps track of labeled data. We differentiate three respective tabs for different purposes:

The *Samples Tab* (Fig. 5.d1) lists samples of the classified windows. They are surrounded by frames color-encoded from green over yellow to red, indicating decreasing similarity to the query. Right above the samples, PSEUDO invites the user to label the windows by clicking  for similar,  for indecisive, and  for dissimilar, as described in Sect. 3.3 (Sample Relevance Adaption).

PSEUDO visualizes the hash tables in the *Tables Tab* (Fig. 5.d2). Each hash table is visualized as a histogram, showing the similarity distribution perceived by the hash function. Like the frames surrounding the samples in the *Samples Tab*, the bars in the histograms here are also color-encoded from green to red, indicating decreasing similarity to the current query. To help understand how well the hash functions work, we plot each time step's mean, minimum, and maximum values among the top-20 similar windows for each hash function. The mean value curves portray the pattern shape perceived as similar by the hash function, while the minimum and maximum values form the lower and upper bound of the pattern. The band's tightness implies the certainty or importance of the track during classification. Based on this visual encoding, the user can modify the hash tables' importance by clicking  for important and  for indecisive, as described in Sect. 3.3 (Hash Table Relevance Adaption). In the future, we plan to implement a ranking or sampling mechanism to avoid showing all hash tables.

The labeled windows for the current round are kept in the *Labeled Data Tab* (Fig. 5.d3). Users can revise decisions before clicking the  button. PSEUDO will consider the labeled sample windows and hash tables in the next training round.

4.3 Results View

The *Results View* (Fig. 5.e) shows the outcome statistics and provides git-like version management.

In the *Classifier Tab* (Fig. 5.e1), we use a result histogram to visualize the distribution of similarity between the query and all windows that survive LSH's pruning. Clicking a bin in the histogram shows the reconstructed visual pattern analogous to the ones in the *Feedback View*. Rather than using the top-20, this pattern result view summarizes all windows in the chosen bin. The mean curves also show the average form within this bin, bounded by each time step's minimum and maximum values to illustrate the variance within tracks. The histogram and the reconstructed shape help the user better understand the classification result and provide guidance on the strictness during the labeling process. Besides, users can set the number of top candidates and display them in the dataset overview.

5 EVALUATION

We evaluate PSEUDO against the requirements described in Sect. 2 through three distinct evaluation threads. First, we benchmark accuracy (**R4**) and speed/scalability (**R1**) with representative techniques for

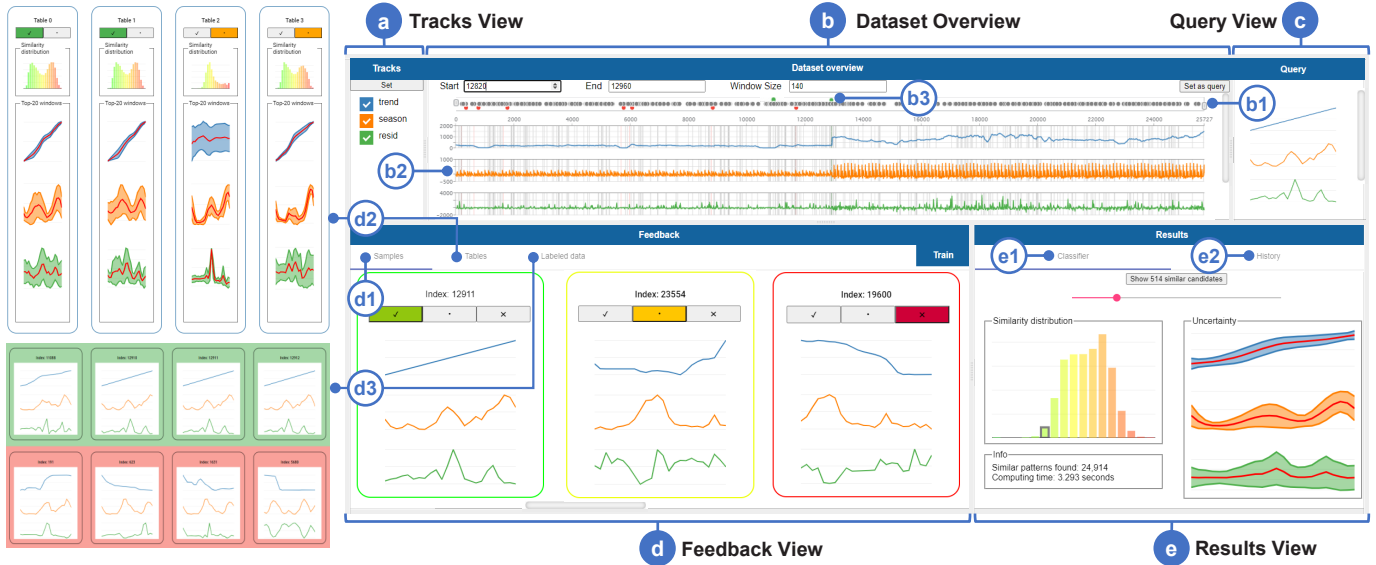


Fig. 5. **Interface Overview:** We plot the selected tracks from the *Tracks View* (a) in the *Dataset Overview* (b) along with the user-defined query in (c). The *Feedback View* (d) depicts classification result samples and information about the hash tables and is used to receive relevant feedback. The *Results View* (e) displays the result distribution and provides search history management.

time series pattern retrieval. Second, we verify the steerability of the relevance feedback mechanism (R2). Third, we validate the usability, including understandability (R3), through an interview with an expert from the energy transition domain. Please note that, accuracy and speed evaluation have already been conducted in [60]. We extend the accuracy evaluation with labeled datasets and extend the speed evaluation with a scalability test with respect to data dimensions.

We use six datasets with different characteristics, namely 1) Deep Valve: electrical current through a solenoid valve, 2) EEG Eye State, 3) Filling Prediction: relative air filling in an Otto engine, 4) Variable Displacement: rotational speed and acceleration of an Otto engine 5) EEG Schizophrenia and 6) Intelliekon: household energy consumption. 1) to 4) have ground truth labels. Therefore, they are used in the accuracy benchmark. While the speed benchmark also uses these four datasets, we would like to evaluate PSEUDO's scalability with respect to higher data dimensions more thoroughly. Hence, we include the unlabeled EEG Schizophrenia dataset with 70 effective tracks because the speed/scalability test does not require labels. 6) is the dataset used by the interviewed expert. We keep a detailed description of the datasets and the physical background of the queries in Appendix A.

All experiments are conducted on a standard laptop running on 64-bit Windows 10 Enterprise with Intel i7-8650U CPU, 16GB RAM, and 1TB HDD. We keep the complete experiment setup, including hardware, software, and parameters in Appendix B.

5.1 Accuracy and Speed

We benchmark PSEUDO's accuracy with deactivated relevance feedback to the representative methods correlation, DTW [7,44], MASS [43, 45], and Symbolic Aggregate approxXimation (SAX) [37, 38].

SAX is a representation of time series. However, it comes with a distance measure [38], making it a full-fledged similarity measure. SAX uses Piecewise Aggregate Approximation (PAA) as a necessary preprocessing step, which reduces data volume. To ensure a fair comparison, we apply PAA with the same resolution for other methods.

Besides step-wise classification metrics accuracy, balanced accuracy, precision, recall, and F1 score, we borrow the metric mean Average Precision (mAP) from object detection in computer vision in favor of a segment-wise perspective. This metric requires a threshold for Intersection over Union (IoU) between a prediction (predicted pattern) and the closest ground truth label to judge the prediction as a true positive. In this context, the IoU between a prediction and a ground truth label means their overlapping time span divided by their joint time span. If

there is no overlapping, IoU is null. We have used 30% and 50% as the IoU-threshold. They are denoted mAP30 and mAP50, respectively. For other metrics, we tune the confidence/distance threshold to achieve the best F1 score.

According to the benchmark in Table 1, each method performs well on some datasets while poorly on others. This result implies that the notion of similarity may vary in different use cases, which leads to the need for adaptive similarity measures.

Next, we measure the average elapsed time over five repetitions for all methods on the same four labeled datasets and report the results in Table 2. PSEUDO, in our *slow* DTW configuration, is comparable with MASS, the fastest similarity search algorithm so far, and even surpasses the latter in high-dimensional cases like the EEG eye state dataset. In low-dimensional datasets, the pruning effect of LSH plays a major role by reducing the number of candidates. Whereas in high-dimensional cases, LSH's weighted track merging provides sub-linear scalability. Please note that we conduct all experiments with Python. However, the Python version of MASS is slower than the MATLAB version, which is still hard to beat.

To further verify this scalability, we measure the elapsed time with an increasing number of tracks in the high-dimensional EEG Schizophrenia dataset. The result in Fig. 6 confirms PSEUDO's good scalability for high-dimensional data. Please refer to Appendix C for details of the scalability analysis and experiment.

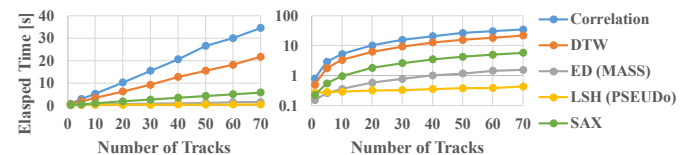


Fig. 6. **Scalability Benchmark:** PSEUDO's speed stands out as the data dimensions increase. While the other methods scale linearly with the data dimensions, PSEUDO achieves sub-linear scalability. Left: linear scale; right: logarithmic scale.

5.2 Steerability

While Sect. 5.1 measures the "open-loop" performance without relevance feedback, this section verifies the effectiveness and mechanics of relevance feedback through accuracy and track importance evolution.

Datasets	Methods	Accuracy	Balanced accuracy	Precision	Recall	F1 score	mAP30	mAP50
Deep valve - pattern contains pulses - large horizontal scaling	Correlation	0.9	0.77	0.8	0.57	0.66	0.78	0.22
	DTW	0.92	0.81	0.92	0.63	0.75	0.59	0.26
	ED (MASS)	0.18	0.5	0.18	1	0.31	0.06	0.01
	LSH (PSEUDO)	0.92	0.87	0.78	0.79	0.78	0.9	0.74
	SAX	0.96	0.94	0.89	0.91	0.9	0.78	0.36
EEG Eye State - pattern very fuzzy - extreme horizontal scaling	Correlation	0.64	0.66	0.57	0.8	0.67	0.29	0.1
	DTW	0.78	0.79	0.72	0.84	0.77	0.61	0.5
	ED (MASS)	0.72	0.72	0.66	0.78	0.71	0.29	0.16
	LSH (PSEUDO)	0.7	0.7	0.66	0.67	0.67	0.25	0.17
	SAX	0.45	0.5	0.45	1	0.62	0.1	0.09
Filling prediction - pattern mainly stationary - pattern variation concentrate on both ends	Correlation	0.97	0.79	0.95	0.59	0.73	0.79	0.79
	DTW	0.94	0.9	0.54	0.85	0.66	0.77	0.77
	ED (MASS)	0.97	0.8	0.9	0.6	0.72	0.78	0.78
	LSH (PSEUDO)	0.95	0.67	0.9	0.35	0.51	0.55	0.55
	SAX	0.98	0.87	0.93	0.74	0.82	0.94	0.94
Variable Displacement - vibrating pattern	Correlation	1.00 ± 0.00	0.96 ± 0.02	0.98 ± 0.01	0.91 ± 0.03	0.95 ± 0.01	0.83 ± 0.34	0.83 ± 0.34
	DTW	1.00 ± 0.00	0.92 ± 0.07	0.88 ± 0.10	0.84 ± 0.15	0.85 ± 0.10	0.74 ± 0.31	0.67 ± 0.34
	ED (MASS)	0.94 ± 0.04	0.72 ± 0.10	0.14 ± 0.09	0.49 ± 0.23	0.19 ± 0.09	0.07 ± 0.07	0.02 ± 0.02
	LSH (PSEUDO)	0.99 ± 0.01	0.74 ± 0.12	0.69 ± 0.34	0.48 ± 0.24	0.54 ± 0.24	0.31 ± 0.21	0.30 ± 0.21
	SAX	1.00 ± 0.00	0.90 ± 0.07	0.95 ± 0.04	0.80 ± 0.14	0.86 ± 0.10	0.66 ± 0.33	0.66 ± 0.33

Table 1. **Accuracy Benchmark:** Different methods work well on different datasets, indicating varying similarity notions in different use cases. PSEUDO's relevance feedback mechanism is deactivated in this experiment.

Datasets	Deep valve	EEG Eye	Filling pred.	Variable Disp.
Correlation	5.53 ± 0.45	9.89 ± 0.40	3.32 ± 0.06	51.16 ± 1.02
DTW	3.74 ± 0.06	20.67 ± 0.32	3.81 ± 0.10	34.72 ± 0.86
ED (MASS)	0.68 ± 0.03	0.56 ± 0.04	0.36 ± 0.03	4.45 ± 0.06
LSH (PSEUDO)	0.93 ± 0.04	0.30 ± 0.02	0.70 ± 0.05	9.21 ± 0.16
SAX	1.62 ± 0.10	4.16 ± 0.07	1.67 ± 0.04	11.95 ± 0.44

Table 2. **Speed Benchmark:** Search time is in seconds and averaged over five repeats. PSEUDO is comparable in speed with the so far fastest similarity search tool MASS and surpasses the latter in the high-dimensional case (the EEG eye state dataset). Please note that the used Python implementation of MASS is significantly slower than the MATLAB version, which is still hard to beat.

In general, it is challenging to evaluate active learning systems objectively. Inspired by [57], we evaluate PSEUDO's steerability with three agents simulating user behavior. The first agent simulates normal user feedback and labels the samples according to the ground truth labels. If a sample has at least 50% IoU with a ground truth label, the sample is accepted, otherwise rejected. The second agent accepts all samples, and the third rejects all. They form the control groups. We use the EEG Eye State dataset to demonstrate PSEUDO's steerability because this dataset is high-dimensional and has ground truth labels. A plot with all tracks can be found in Appendix D. The target patterns correspond to the periods when the subject's eyes are closed. This dataset contains 14 tracks. They are not equally informative for the pattern search. Fig. 8 plots four representative tracks with target patterns in gray. We can notice that the patterns in the upper two tracks, "F8" and "AF4", are much more prominent, while the lower two, "T7" and "P7", are not as helpful.

We run PSEUDO on the dataset with five feedback rounds and record the accuracy as well as the track weights evolution. Fig. 7 shows the accuracy metrics mAP30 and mAP50 in five feedback rounds with the three agents. We witness an accuracy increase with the normal agent and no improvement in the control groups. It confirms that sensible feedback helps improve PSEUDO's accuracy.

To further verify PSEUDO's feature selection-based relevance feedback mechanism, we plot the evolution of the track weights (\bar{w} in Sect. 3.3) in Fig. 3 (complete plot in Fig. 3), next to the plot of data. As expected, the more instructive tracks are attached more weights, and the less informative tracks get down-weighted. In contrast, the track weights from the "all accepted" agent group evolve randomly because the feedback contains no useful information. The track weights from

the "all rejected" agent group stay the same because PSEUDO currently cannot exploit the rejected samples, which is one of its limitations.

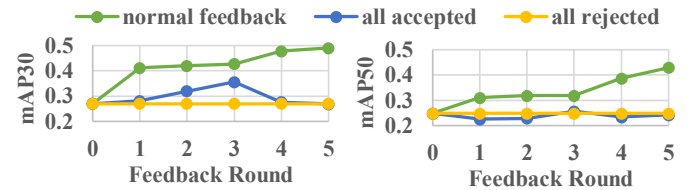


Fig. 7. **Accuracy Evolution:** During relevance feedback, accepting or rejecting all samples yields no accuracy improvement, while faithful user feedback contributes to the accuracy. Left: mAP30; right: mAP50.

5.3 Case Study

We conducted an in-person case study to show PSEUDO's usability in a real-world use case in the energy domain. The invited expert leads the business unit of demand response and smart grids. He has over 19 years of experience with smart meter data. In an individual one-hour session, we first introduced PSEUDO's functionality, discussed the primary use cases for smart meter data, and provided a brief demonstration. The expert expressed great interest in PSEUDO: "It is a quite interesting tool to analyze time series." and mentioned: "I can imagine we can use it to try to recognize specific characteristics." We brainstormed multiple use cases for smart meter data analysis. Then, we selected one of the brainstormed application scenarios to show how PSEUDO can help him identify shifts in energy consumption behaviors in households. Energy consumption behavior is the response to complex environments that should be analyzed on several temporal scales. However, our expert currently lacks a tool to identify the changes in energy consumption behaviors in high resolution. Instead, his team calculates the means of smart meter data in specific periods, which is inflexible and cannot capture minor changes in the periods.

For his use case (depicted in Fig. 5), we conducted experiments on a smart meter dataset collected from a field study between 2009 and 2010 within the German research project Intelliekon [52]. For two of 1720 randomly picked households, the expert decomposed the one-year hourly energy consumption into the trend, seasonal, and residual components based on a Bayesian structural TS model. These three tracks are loaded as our MTS input (Fig. 5 (a)). During the visual exploration with PSEUDO, the shifts in user behaviors can be

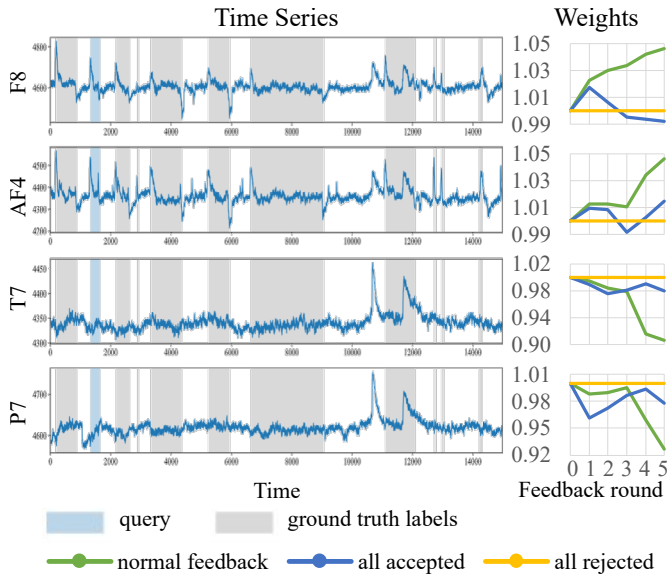


Fig. 8. **Track Weight Evolution:** The target patterns are not equally prominent in all tracks. Visually, the patterns in the upper two tracks, “F8” and “AF4”, are more pronounced, while the lower two tracks, “T7” and “P7”, are less informative. PSEUDO attaches higher weights to more relevant features/tracks during relevance feedback.

identified by searching for characteristic patterns in various temporal scales (monthly, weekly, and daily).

During the study, the expert operated PSEUDO. It helped the expert identify shifts in user behavior and inspect the behavior differences between user groups. Overall, the expert spoke highly of PSEUDO. The expert highlighted the interaction during querying and exploration: “It is very useful to have a view of similarities among searched results to help pick up the thresholds of similarity.”; i.e., Fig. 5(e). In the above use case, the expert did not address any significant usability issues. Only when we imagined the broader use cases with multiple households, the expert suggested that it would be more convenient to support visualizing multiple files as track groups. From his perspective, a nice add-on for PSEUDO would be a textual explanation of the hashing algorithm and its visualizations because it can help him avoid explaining the results to customers without a deep technical background. We keep details of the interview in Appendix E.

6 RELATED WORK

To avoid interrupting the information flow, we move the related work here, providing a broader context.

Time Series Analysis: Time series is a sequence of observations recorded chronologically [9, 56]. If each observation forms a high-dimensional vector, the time series is multivariate [60]. Each dimension in the time series is called a channel, track, trace, attribute, or feature. [30, 56] list around ten typical tasks that analysts perform on such data, like anomaly detection, classification, and segmentation. Time series indexing best describes our work, which addresses the problem of finding time series in a database that are most similar to a given query. With sliding windows, we convert the problem to time series indexing, which is common among similar works [34, 36, 42]. It may result in inefficiency though, when the pattern length varies. Recently, [42] combats this issue by matching trends of sub-sequences in the pattern as the first step, and [55] approaches the problem with quantifiers in regex. Currently, we use a series of sliding windows of logarithmically increasing size to capture patterns with varying duration.

Time Series Representation: A time series representation transforms time series into another form. It compresses data volume, reduces dimensionality, or extracts latent features. Figure 1 in [38] presents a hierarchy of 17 time series representations. Most common ones include Discrete Fourier Transform (DFT), Fast Fourier transform (FFT),

Symbolic Aggregate approxImation (SAX), and Singular Value Decomposition (SVD). They continue to develop in recent years. For instance, SAX assumes a normal distribution of the values in the time series, which is not always satisfied. [25] and [8] solve this problem with inverse normal transformation and a data-driven kernel density estimator respectively. By focusing on a portion of Fourier coefficients, Partial Fourier Transform (PFT) speeds up FFT further by an order of magnitude [48]. Recently, deep learning models have been introduced to represent time series, like pretrained Convolutional Neural Networks (CNNs) [2], convolutional autoencoder [36], and TS2Vec [61]. Our baseline method LSH can also be regarded as a time series representation, which we will elaborate in the following paragraphs.

Locality-sensitive Hashing: Generally, hashing-based algorithms aim for a significant speed boost with tolerable information loss. First introduced in [27], LSH inherits this concept while differing from the other hashing algorithms in that it maps similar objects to close hash codes. This feature enables its application in a wide range of data mining problems, like nearest neighbor search [26, 27], hierarchical clustering [13, 32], and near-duplicate detection [14]. Conceptually, LSH conceives data objects in the database as points in a high-dimensional space. This space is cut into sub-regions by a group of random hyperplanes. Each hyperplane is described by a hash function [27]. Each object points in the same sub-region fall into the same bucket in the hash table. To reduce false negatives, multiple hyperplane groups or multiple hash function groups are used. All objects colliding with the query under at least one hash function group are treated as candidates. Subsequently, the candidates can be filtered with a similarity measure to suppress false positives. Since LSH is considerably fast, it can be repeated for every new query with query-centered bucketing, namely Query-Aware Locality Sensitive Hashing (QALSH), yielding better accuracy [26]. Recently, LSH received attention for time series indexing, where it achieves up to 20 times faster processing speeds with a minor accuracy sacrifice [41]. Then, [60] introduced a query-aware adaptation for MTS. We extend the idea by making LSH trainable.

Time Series Similarity Measure: Measuring similarity/distance between two time series is a fundamental problem in time series analysis. Besides the L^p -norms, including ED (L^2 -norm), tolerance band (L^∞ -norm), and Manhattan distance (L1-norm), Dynamic Time Warping (DTW) [7] is one of the most popular elastic similarity measures that tackle time shifts. It shows superior accuracy even compared with machine learning methods, as the extensive benchmark in [4] indicates. Numerous other distance measures also exist. [40] benchmarked eleven model-free similarity measures for time series classification. Recent representative model-based similarity measures include siamese networks [33, 49] and NeuralWrap [29]. Some representations bring their own similarity measures, like SAX’s distance measure between symbols [38] and ShapeSearch’s scoring system defined upon their shape primitives [55]. Starting from [31], researchers introduce relevance feedback mechanisms to similarity measures to capture users’ similarity notions [16, 36]. PSEUDO supports various similarity measures. In experiments, we adopt DTW on the adaptive LSH representation to address the issue of subjective similarity.

Visual Query Systems: The term Visual Query System (VQS) is introduced by [51] and refers to tools that allow visual pattern retrieval via a user interface [35]. One essential component is query definition. Two prevailing methods are query-by-example and query-by-sketch. The former defines a query by providing an example, e.g., marking an interval in the time series plot, like in TimeSearcher [23] and PEAX [36]. This method is preferable if the query is complex and an example is accessible [36]. Query-by-sketch specifies a query by drawing it. Examples range from QuerySketch [58] over QueryLines [51] to Zenvisage++ [35]. It gives users more freedom, especially when the initial example is hard to find [46]. It is an active research area as capturing the unbiased concept from the user’s drawing is challenging [35]. In addition, there are other query definition methods, e.g., ShapeSearch [55] supports natural language and regex besides sketch. We list recent representative tools for time series pattern search together with PSEUDO in Table 3. Rigorously speaking, MASS is a similarity

Tool	Query Definition	Feature Representation	Similarity Measure / Classifier	Multivar.	Relevance Feedback	Focus / contributions
MASS [43,59]	Provide query directly	None	ED	No	No	Speed
Qetch [42]	Query by sketch	Trend	Own local distortion + shape error	No	No	Query by sketch
STSS [28]	Provide query directly	Value and trend	Match/unmatch regex pattern	No	No	Semantic shape description
ShapeSearch [55]	Query-by-sketch / natural lang. / regex	Position, trend and dedicated operators	Scoring system defined on the representation	No	No	Semantic shape description
PEAX [36]	Query-by-example	Conv. autoencoder	Random Forest	No	Yes	Accuracy repr., adapt. similarity
Zenvisage++ [54]	Query-by-sketch	None	ED / DTW / Segmentation / MVIP	No	No	Query-by-sketch
PSEUDO	Query-by-example	LSH	ED / DTW	Yes	Yes	Speed, adaptive similarity

Table 3. **Comparison of recent VQSs:** PSEUDO mainly focuses on efficient and interpretable pattern search in high-dimensional time series with a use-case-dependent similarity notion.

search algorithm and not a VQS. However, it is the fastest similarity search algorithm and thus worth mentioning. Compared with PSEUDO, few VQSs deal with multivariate time series and only PEAX addresses the problem with the subjective similarity notion. PEAX’s use case in epigenomics is relatively stable, making it acceptable to train a convolutional autoencoder with top gear for several days. In comparison, PSEUDO focuses more on speed and interoperability, catering to the spontaneous needs, especially for high-dimensional time series.

7 DISCUSSION AND LIMITATIONS

PSEUDO goes beyond the state-of-the-art in interactive MTS analysis by incorporating three aspects to make MTS data exploration more tractable for real-world applications. First, it implements an adaptive classification making it a user-centric Visual Analytics approach, in contrast to static deep learning-based one-shot methods. Second, our method utilizes one of the most scalable and efficient data processing techniques: hashing-based algorithms. Third, the implemented concept of “buckets” is easy to understand, allowing for a fast adaption of PSEUDO in less ML-savvy application environments. However, during the project, we came across conceptual, design, and implementation challenges, that we would like to discuss in the following.

On the conceptual side, we found that a thorough task taxonomy for MTS analysis is missing. We can map PSEUDO’s high-level tasks into Brehmer and Munzner’s typology [10], e.g., our tool implements *browse*, *explore*, *locate*, and *lookup* tasks. However, we did not focus on MTS tasks like finding patterns with significant cross-track time shifts or tasks that assess the invariance properties of specific patterns as these tasks are conceptually on a different level of abstraction.

We invested much effort in the scalable and fast search backend but did not focus much on the frontend. Our rather simplistic use of standard visualizations and the implemented query-by-example system, though work flawlessly and effortlessly, demonstrate this aspect distinctively. We admit that the visualization does not scale to very high-dimensional data. We are considering reactive switching between line charts, horizon diagrams [24], and color-encoded pixels [1]. We are also considering distorting the time series in the plot. For instance, using different heights for tracks with varied importance and distorting the time to compress unimportant time regions. In the future, we plan to extend our Visual Analytics contributions in two directions: First, we will tackle query-definition challenges, like *How can users specify a) multi-track queries or b) queries with a temporal relationship between them?*, with new query definition panels and plan to apply interactive augmentation, like Shadow Draw [53], to help with this process.

A limitation within PSEUDO is that we model MTS data as numerical vectors with a fixed temporal resolution and assume the tracks in the target patterns are synchronized. However, we can envision more complex application scenarios, such as in crime analysis, where MTS tracks are a) not synchronized and b) contain categorical or even complex data types, such as surveillance webcam images.

Another interesting challenge for VA is tracking *biases* and *convergence* in exploration processes. Currently, we include negative and indecisive labels to promote target class separation, which inevitably adds to a potential confirmation bias in every iteration. We could, however, also regard every new positive label, which is distinct from the current set of positive labeled items, as a novel exploration thread or

fork. This enables *quality metrics to quantify task change(s)*, i.e., a strong difference between positive labels could signal a transition from exploitation to exploration. We are currently also examining feedback beyond binary labels, such as a similarity score and feedback regarding the shape, size, and position, e.g., allowing tuning the start and end time of the samples.

Finally, we will address the evaluation of more PSEUDO components in the future. Specifically, we plan to run a dedicated study on the proposed relevance feedback on the hash tables (Sect. 3.2) with the focus to prove the usability and effectiveness of this feedback mechanism. Second, we calculate the similarity based on the hash codes rather than the original time series windows as described in Sect. 3.1. Although it shows no problem in the accuracy evaluation, it is sensible to measure the potential accuracy sacrifice brought by this modification.

8 CONCLUSION

This work has proposed PSEUDO, an efficient, adaptive, and interpretable tool for visual pattern retrieval in multivariate time series based on LSH and relevance feedback. We found PSEUDO impressively efficient for very high-dimensional time series. It works well in use cases where initial labels are meager and the promptness of the result counts. These properties make it particularly useful for user interaction in VQSs. Furthermore, we found that PSEUDO improves results with an also efficient relevance feedback mechanism based on feature selection. This property helps capture subjective task-dependent similarity and hints for further domain-specific analysis. In the future, we expect an increasing collaboration between hashing algorithms and machine learning due to the explosion of data size, e.g., for massive video processing. As future work for PSEUDO, we are especially interested in visualizing high-dimensional time series with different track importance and examining more possibilities for relevance feedback.

REFERENCES

- [1] A. Gogolou, T. Tsandilas, T. Palpanas, and A. Bezerianos. Comparing similarity perception in time series visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):523–533, 2019. doi: 10.1109/TVCG.2018.2865077
- [2] G. Anand and R. Nayak. Unsupervised visual time-series representation learning and clustering. In H. Yang, K. Pasupa, A. C.-S. Leung, J. T. Kwok, J. H. Chan, and I. King, eds., *Neural Information Processing*, pp. 832–840. Springer International Publishing, Cham, 2020.
- [3] D. Arendt, E. Saldanha, R. Wesslen, S. Volkova, and W. Dou. Towards rapid interactive machine learning: evaluating tradeoffs of classification without representation. In W. Fu, S. Pan, O. Brdiczka, P. Chau, and G. Calvary, eds., *Proc. 24th Int. Conf. on Intelligent User Interfaces, IUI 2019*, pp. 591–602. ACM, 2019. doi: 10.1145/3301275.3302280
- [4] 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.
- [5] N. Begum, L. Ulanova, A. Dau, J. Wang, and E. Keogh. A general framework for density based time series clustering exploiting a novel admissible pruning strategy. *CoRR*, abs/1612.00637, 2016.
- [6] M. Behrisch, F. Korkmaz, L. Shao, and T. Schreck. Feedback-driven interactive exploration of large multidimensional data supported by visual classifier. In *2014 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pp. 43–52, 2014. doi: 10.1109/VAST.2014.7042480
- [7] D. J. Berndt and J. Clifford. Using dynamic time warping to find patterns in time series. In *KDD workshop*, vol. 10, pp. 359–370. Seattle, WA, USA., 1994.
- [8] Bountrogiannis Konstantinos, Tzagkarakis George, and Tsakalides Panagiotis. Data-driven kernel-based probabilistic sax for time series dimensionality reduction. In *2020 28th European Signal Processing Conference (EUSIPCO)*, pp. 2343–2347, 2021. doi: 10.23919/Eusipco47968.2020.9287311
- [9] G. E. P. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung. *Time Series Analysis: Forecasting and Control*. John Wiley & Sons, 2015.
- [10] M. Brehmer and T. Munzner. A multi-level typology of abstract visualization tasks. *IEEE Trans. Vis. Comput. Graph.*, 19(12):2376–2385, 2013. doi: 10.1109/TVCG.2013.124
- [11] B. C. M. Cappers, P. N. Meessen, S. Etalle, and J. J. van Wijk. Eventpad: Rapid malware analysis and reverse engineering using visual analytics. In D. Staheli, C. L. Paul, J. Kohlhammer, D. M. Best, S. Trent, N. Prigent, R. Gove, and G. Sauer, eds., *15th IEEE Symposium on Visualization for Cyber Security, VizSec 2018, Berlin, Germany, October 22, 2018*, pp. 1–8. IEEE, 2018. doi: 10.1109/VIZSEC.2018.8709230
- [12] K. Chakrabarti, E. Keogh, S. Mehrotra, and M. Pazzani. Locally adaptive dimensionality reduction for indexing large time series databases. *ACM Trans. Database Syst.*, 27(2):188–228, 2002. doi: 10.1145/568518.568520
- [13] M. Cochez and H. Mou. Twister tries: Approximate hierarchical agglomerative clustering for average distance in linear time. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, SIGMOD '15*, pp. 505–517. Association for Computing Machinery, New York, NY, USA, 2015. doi: 10.1145/2723372.2751521
- [14] A. S. Das, M. Datar, A. Garg, and S. Rajaram. Google news personalization: Scalable online collaborative filtering. In *Proceedings of the 16th International Conference on World Wide Web, WWW '07*, pp. 271–280. Association for Computing Machinery, New York, NY, USA, 2007. doi: 10.1145/1242572.1242610
- [15] F. L. Dennig, T. Polk, Z. Lin, T. Schreck, H. Pfister, and M. Behrisch. Ffive: Learning relevance models using pattern-based similarity measures. In *2019 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pp. 69–80. IEEE, 2019.
- [16] X. Du, L. Deng, and K. Qian. Current market top business scopes trend—a concurrent text and time series active learning study of nasdaq and nyse stocks from 2012 to 2017. *Applied Sciences*, 8(5), 2018. doi: 10.3390/app8050751
- [17] J. J. Dudley and P. O. Kristensson. A review of user interface design for interactive machine learning. *ACM Trans. Interact. Intell. Syst.*, 8(2):8:1–8:37, 2018. doi: 10.1145/3185517
- [18] F. Petitjean, G. Forestier, G. I. Webb, A. E. Nicholson, Y. Chen, and E. Keogh. Dynamic time warping averaging of time series allows faster and more accurate classification. In *2014 IEEE International Conference on Data Mining*, pp. 470–479, 2014. doi: 10.1109/ICDM.2014.27
- [19] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos. Fast subsequence matching in time-series databases. In *Proceedings of the 1994 ACM SIGMOD International Conference on Management of Data, SIGMOD '94*, pp. 419–429. Association for Computing Machinery, New York, NY, USA, 1994. doi: 10.1145/191839.191925
- [20] G. Forestier, F. Petitjean, H. A. Dau, G. I. Webb, and E. Keogh. Generating synthetic time series to augment sparse datasets. In *2017 IEEE International Conference on Data Mining (ICDM)*, pp. 865–870, 2017. doi: 10.1109/ICDM.2017.106
- [21] Georgios Chatzigeorgakidis, Dimitrios Skoutas, Kostas Patroumpas, Themis Palpanas, Spiros Athanasiou, and Spiros Skiadopoulos. Twin subsequence search in time series. *CoRR*, abs/2104.06874, 2021.
- [22] B. C. Gao and D. T. Anh. Similarity search for numerous patterns over multiple time series streams under dynamic time warping which supports data normalization. *Vietnam Journal of Computer Science*, 3(3):181–196, 2016. doi: 10.1007/s40595-016-0062-4
- [23] Harry Hochheiser and Ben Shneiderman. Dynamic query tools for time series data sets: Timebox widgets for interactive exploration. *Information Visualization*, 3(1):1–18, 2004. doi: 10.1057/palgrave.ivs.9500061
- [24] J. Heer, N. Kong, and M. Agrawala. Sizing the horizon: The effects of chart size and layering on the graphical perception of time series visualizations. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '09*, pp. 1303–1312. Association for Computing Machinery, New York, NY, USA, 2009. doi: 10.1145/1518701.1518897
- [25] Hoonseok Park and Jae-Yoon Jung. Sax-arm: Deviant event pattern discovery from multivariate time series using symbolic aggregate approximation and association rule mining. *Expert Systems with Applications*, 141:112950, 2020. doi: 10.1016/j.eswa.2019.112950
- [26] Q. Huang, J. Feng, Y. Zhang, Q. Fang, and W. Ng. Query-aware locality-sensitive hashing for approximate nearest neighbor search. *Proceedings of the VLDB Endowment*, 9(1):1–12, 2015.
- [27] P. Indyk and R. Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing, STOC '98*, p. 604–613. Association for Computing Machinery, New York, NY, USA, 1998. doi: 10.1145/276698.276876
- [28] João Rodrigues, Duarte Folgado, David Belo, and Hugo Gamboa. Ssts: A syntactic tool for pattern search on time series. *Information Processing & Management*, 56(1):61–76, 2019. doi: 10.1016/j.ipm.2018.09.001
- [29] Josif Grabocka and Lars Schmidt-Thieme. Neuralwarp: Time-series similarity with warping networks. *CoRR*, abs/1812.08306, 2018.
- [30] E. Keogh. Indexing and mining time series data. In S. Shekhar and H. Xiong, eds., *Encyclopedia of GIS*, pp. 493–497. Springer US, Boston, MA, 2008. doi: 10.1007/978-0-387-35973-1_598
- [31] E. J. Keogh and M. J. Pazzani. An enhanced representation of time series which allows fast and accurate classification, clustering and relevance feedback. In *Kdd*, vol. 98, pp. 239–243, 1998.
- [32] H. Koga, T. Ishibashi, and T. Watanabe. Fast agglomerative hierarchical clustering algorithm using locality-sensitive hashing. *Knowledge and Information Systems*, 12(1):25–53, 2007. doi: 10.1007/s10115-006-0027-5
- [33] L. Hou, X. Jin, and Z. Zhao. Time series similarity measure via siamese convolutional neural network. In *2019 12th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*, pp. 1–6, 2019. doi: 10.1109/CISP-BMEI48845.2019.8966048
- [34] E. Laftchiev and Y. Liu. Finding multidimensional patterns in multidimensional time series. In *KDD workshop on MiLeTS 2018, London, 2018*.
- [35] D. J. L. Lee, J. Lee, T. Siddiqui, J. Kim, K. Karahalios, and A. G. Parameswaran. You can't always sketch what you want: Understanding sensemaking in visual query systems. *IEEE Trans. Vis. Comput. Graph.*, 26(1):1267–1277, 2020. doi: 10.1109/TVCG.2019.2934666
- [36] F. Lekschas, B. Peterson, D. Hahn, E. Ma, N. Gehlenborg, and H. Pfister. Peax: Interactive visual pattern search in sequential data using unsupervised deep representation learning. In *Computer Graphics Forum*, vol. 39(3), pp. 167–179. Wiley Online Library, 2020.
- [37] J. Lin, E. Keogh, S. Lonardi, and B. Chiu. A Symbolic Representation of Time Series, with Implications for Streaming Algorithms. In *Proc. of 8th ACM SIGMOD Workshop on Research Issues in DM and Knowledge Discovery, DMKD '03*, pp. 2–11. Assoc. for Comp. Mach., New York, NY, USA, 2003. doi: 10.1145/882082.882086
- [38] J. Lin, E. Keogh, L. Wei, and S. Lonardi. Experiencing sax: a novel

- symbolic representation of time series. *Data Mining and knowledge discovery*, 15(2):107–144, 2007.
- [39] R. A. K.-I. Lin and Shim, Harpreet S Sawhney Kyuseok. Fast similarity search in the presence of noise, scaling, and translation in time-series databases. In *Proceeding of the 21th International Conference on Very Large Data Bases*, pp. 490–501, 1995.
- [40] J. Lines and A. Bagnall. Time series classification with ensembles of elasticdistance measures. *Data Mining and Knowledge Discovery*, 29(3):565–592, 2015. doi: 10.1007/s10618-014-0361-2
- [41] C. Luo and A. Shrivastava. SSH (Sketch, Shingle, & Hash) for indexing massive-scale time series. In *NIPS 2016 Time Series Work.*, pp. 38–58, 2017.
- [42] M. Mannino and A. Abouzied. Expressive time series querying with hand-drawn scale-free sketches. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI '18, p. 1–13. Association for Computing Machinery, New York, NY, USA, 2018. doi: 10.1145/3173574.3173962
- [43] R. Mercer, S. Alaei, A. Abdoli, S. Singh, A. Murillo, and E. Keogh. Matrix profile xxiii: Contrast profile: A novel time series primitive that allows real world classification. In *2021 IEEE 16th International Conference on Data Mining (ICDM)*, 2021.
- [44] Mohammad Shokoohi-Yekta, Jun Wang, and Eamonn Keogh. On the Non-Trivial Generalization of Dynamic Time Warping to the Multi-Dimensional Case. In *Proc. of the 2015 SIAM Int. Conference on Data Mining*, pp. 289–297, 2015. doi: 10.1137/1.9781611974010.33
- [45] A. Mueen, Y. Zhu, M. Yeh, K. Kamgar, K. Viswanathan, C. Gupta, and E. Keogh. The fastest similarity search algorithm for time series subsequences under euclidean distance, August 2017. <http://www.cs.unm.edu/~mueen/FastestSimilaritySearch.html>.
- [46] N. Ruta, N. Sawada, K. McKeough, M. Behrisch, and J. Beyer. Sax navigator: Time series exploration through hierarchical clustering. In *2019 IEEE Visualization Conference (VIS)*, pp. 236–240, 2019. doi: 10.1109/VISUAL.2019.8933618
- [47] T. Negi and V. Bansal. Time series: Similarity search and its applications. In *Proceedings of the International Conference on Systemics, Cybernetics and Informatics: ICSCI-04, Hyderabad, India*, pp. 528–533, 2005.
- [48] Y.-c. Park, J.-G. Jang, and U. Kang. Fast and accurate partial fourier transform for time series data. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, KDD '21*, pp. 1309–1318. Association for Computing Machinery, New York, NY, USA, 2021. doi: 10.1145/3447548.3467293
- [49] W. Pei, D. M. J. Tax, and L. van der Maaten. Modeling time series similarity with siamese recurrent networks. *arXiv preprint arXiv:1603.04713*, 2016.
- [50] F. Petitjean, A. Ketterlin, and P. Gançarski. A global averaging method for dynamic time warping, with applications to clustering. *Pattern Recognit.*, 44(3):678–693, 2011. doi: 10.1016/j.patcog.2010.09.013
- [51] K. Ryall, N. Lesh, T. Lanning, D. Leigh, H. Miyashita, and S. Makino. Querylines: Approximate query for visual browsing. In *CHI '05 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '05, pp. 1765–1768. Association for Computing Machinery, New York, NY, USA, 2005. doi: 10.1145/1056808.1057017
- [52] J. Schleich, M. Klobasa, M. Brunner, S. Gözl, K. Götz, and G. Sunderer. Smart metering in germany—results of providing feedback information in a field trial. *Proceedings of the ECEEE 2011 Summer Study*, pp. 1667–1674, 2011.
- [53] L. Shao, M. Behrisch, T. Schreck, T. von Landesberger, M. Scherer, S. Bremm, and D. A. Keim. Guided sketching for visual search and exploration in large scatter plot spaces. In M. Pohl and J. C. Roberts, eds., *5th International EuroVis Workshop on Visual Analytics, EuroVA@EuroVis 2014, Swansea, UK, June 9-10, 2014*. Eurographics Association, 2014. doi: 10.2312/eurova.20141140
- [54] T. Siddiqui, J. Lee, A. Kim, E. Xue, X. Yu, S. Zou, L. Guo, C. Liu, C. Wang, K. Karahalios, and A. G. Parameswaran. Fast-forwarding to desired visualizations with zenvisage. In *8th Biennial Conference on Innovative Data Systems Research, CIDR 2017, Chaminade, CA, USA, January 8-11, 2017, Online Proceedings*. www.cidrdb.org, 2017.
- [55] T. Siddiqui, P. Luh, Z. Wang, K. Karahalios, and A. G. Parameswaran. Shapesearch: A flexible and efficient system for shape-based exploration of trendlines. In D. Maier, R. Pottinger, A. Doan, W. Tan, A. Alawini, and H. Q. Ngo, eds., *Proceedings of the 2020 International Conference on Management of Data, SIGMOD Conference 2020, online conference [Portland, OR, USA], June 14-19, 2020*, pp. 51–65. ACM, 2020. doi: 10.1145/3318464.3389722
- [56] Tak-chung Fu. A review on time series data mining. *Engineering Applications of Artificial Intelligence*, 24(1):164–181, 2011. doi: 10.1016/j.engappai.2010.09.007
- [57] R. van de Schoot, J. de Bruin, R. Schram, P. Zahedi, J. de Boer, F. Weijdemans, B. Kramer, M. Huijts, M. Hoogerwerf, G. Ferdinands, A. Harkema, J. Willemsen, Y. Ma, Q. Fang, S. Hindriks, L. Tummers, and D. L. Oberski. An open source machine learning framework for efficient and transparent systematic reviews. *Nature Machine Intelligence*, 3(2):125–133, 2021. doi: 10.1038/s42256-020-00287-7
- [58] M. Wattenberg. Sketching a graph to query a time-series database. In *CHI '01 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '01, pp. 381–382. Association for Computing Machinery, New York, NY, USA, 2001. doi: 10.1145/634067.634292
- [59] Yeh Chin-Chia Michael, Zhu Yan, Ulanova Liudmila, Begum Nurjahan, Ding Yifei, Dau Hoang Anh, Silva Diego Furtado, Mueen Abdullah, and Keogh Eamonn. Matrix profile i: All pairs similarity joins for time series: A unifying view that includes motifs, discords and shapelets. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*, pp. 1317–1322, 2016. doi: 10.1109/ICDM.2016.0179
- [60] C. Yu, L. Luo, L. L.-H. Chan, T. Rakthanmanon, and S. Nutanong. A fast lsh-based similarity search method for multivariate time series. *Information Sciences*, 476:337–356, 2019.
- [61] Zhihan Yue, Yujing Wang, Juanyong Duan, Tianmeng Yang, Congrui Huang, and Bixiong Xu. Learning timestamp-level representations for time series with hierarchical contrastive loss. *CoRR*, abs/2106.10466, 2021.