NOOPS: Nonmyopic Feedback-Driven Pattern Search in Multivariate Time Series

Yuncong Yu[a,b,*], Tim Becker[a], Peter Schichtel[a], Wolfgang Aigner[c], Alexandru Telea[b], Michael Behrisch[b]

[a] *Department of Data Science and AI Projects, IAV GmbH, Rockwellstraße 16, Gifhorn, 38518, Lower Saxony, Germany*
[b] *Department of Information and Computing Sciences, Utrecht University, Princetonplein 5, Utrecht, 3584 CC, Utrecht, Netherlands*
[c] *Department of Computer Science and Security / Media and Digital Technologies, St. Pölten, Campus-Platz 1, St. Pölten, A-3100, Lower Austria, Austria*

Abstract

We propose a data processing pipeline for feedback-driven pattern search in multivariate time series. Prevailing machine-learning-based pattern search methods scrutinize time series with a sliding window and classify the data fragment in the window. They retrieve patterns of fixed sizes and assess the pattern relevance based on its internal characteristics oblivious of the pattern's context. Furthermore, they confine user feedback to acceptance, rejection, and abstention, an ineffective approach when a it target pattern is located imprecisely. We propose "nonmyopic" (literally meaning "non-nearsighted") multivariate time series pattern search. Instead of classifying patterns directly, nonmyopic search classifies each time series step as "in

*Corresponding author
Email addresses: yuncong.yu@iav.de (Yuncong Yu), tim.becker@iav.de (Tim Becker), peter.schichtel@iav.de (Peter Schichtel), wolfgang.aigner@fhstp.ac.at (Wolfgang Aigner), a.c.telea@uu.nl (Alexandru Telea), m.behrisch@uu.nl (Michael Behrisch)

Graphical Abstract

NOOPS: Nonmyopic Feedback-Driven Pattern Search in Multivariate Time Series

Yuncong Yu, Tim Becker, Peter Schichtel, Wolfgang Aigner, Alexandru Telea, Michael Behrisch

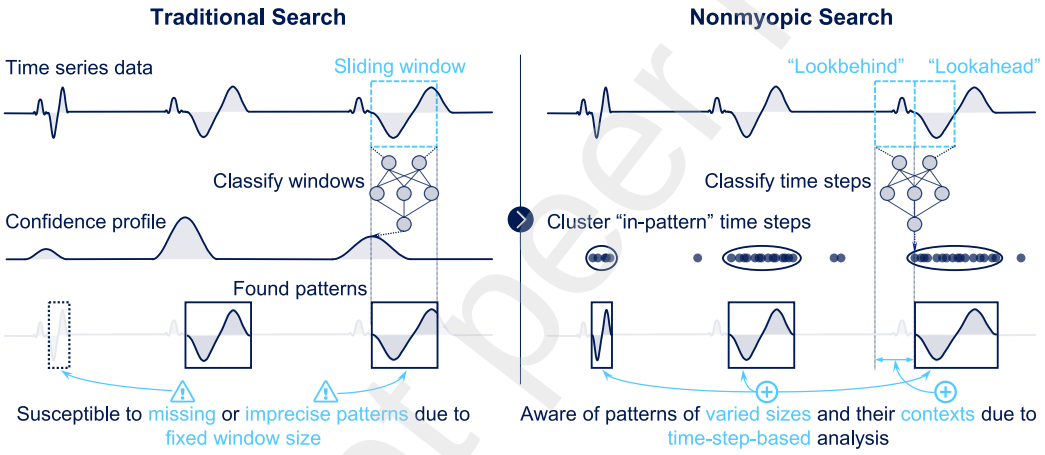

Figure 1: **Traditional Search vs. Nonmyopic Search.** We propose nonmyopic search for model-based time series pattern search. The traditional pattern search in time series scans the time series data with a sliding window to calculate a relevance profile, where top local maxima correspond to the found patterns. The traditional method has problems with patterns of variable duration and is oblivious to the pattern context. In contrast, nonmyopic search classifies whether each time step is in a target pattern or not and clusters the "in-target-pattern" time steps to retrieve pattern intervals. Nonmyopic search finds patterns of variable lengths in one scan and is aware of the pattern context.

Highlights

NOOPS: Nonmyopic Feedback-Driven Pattern Search in Multivariate Time Series

Yuncong Yu, Tim Becker, Peter Schichtel, Wolfgang Aigner, Alexandru Telea, Michael Behrisch

- Step-based classification + clustering outperforms scanning with sliding windows
- Nonmyopic search retrieves patterns of varying duration and considers their contexts
- Nonmyopic search is more accurate, efficient, and robust than traditional search
- Feedback on positions and sizes besides relevance increases accuracy significantly
- During evaluation, interval-based metrics work better than step-based ones

NOOPS: Nonmyopic Feedback-Driven Pattern Search in Multivariate Time Series

Yuncong Yu[a,b,*], Tim Becker[a], Peter Schichtel[a], Wolfgang Aigner[c], Alexandru Telea[b], Michael Behrisch[b]

[a] *Department of Data Science and AI Projects, IAV GmbH, Rockwellstraße 16, Gifhorn, 38518, Lower Saxony, Germany*
[b] *Department of Information and Computing Sciences, Utrecht University, Princetonplein 5, Utrecht, 3584 CC, Utrecht, Netherlands*
[c] *Department of Computer Science and Security / Media and Digital Technologies, St. Pölten, Campus-Platz 1, St. Pölten, A-3100, Lower Austria, Austria*

Abstract

We propose a data processing pipeline for feedback-driven pattern search in multivariate time series. Prevailing machine-learning-based pattern search methods scrutinize time series with a sliding window and classify the data fragment in the window. They retrieve patterns of fixed sizes and assess the pattern relevance based on its internal characteristics oblivious of the pattern's context. Furthermore, they confine user feedback to acceptance, rejection, and abstention, an ineffective approach when a it target pattern is located imprecisely. We propose "nonmyopic" (literally meaning "non-nearsighted") multivariate time series pattern search. Instead of classifying patterns directly, nonmyopic search classifies each time series step as "in

*Corresponding author
Email addresses: yuncong.yu@iav.de (Yuncong Yu), tim.becker@iav.de (Tim Becker), peter.schichtel@iav.de (Peter Schichtel), wolfgang.aigner@fhstp.ac.at (Wolfgang Aigner), a.c.telea@uu.nl (Alexandru Telea), m.behrisch@uu.nl (Michael Behrisch)

target pattern" or not. Then, it merges the "in-target-pattern" time steps into intervals of found patterns through density-based clustering. Nonmyopic search only scans the data once and can retrieve patterns of variable sizes. Moreover, it is aware of the temporal contexts of a pattern, which is crucial for wide application domains like automotive engineering. In addition, we propose "multi-feature feedback". It allows user feedback on the correctness of the found patterns as well as on their positions and sizes. Multi-feature feedback addresses the imprecision problem with conventional binary relevance feedback. Experiments verify that nonmyopic search outperforms traditional search in retrieval accuracy and speed, while multi-feature feedback further improves retrieval accuracy compared to binary relevance feedback. Our work helps automotive engineers trace events in various measurements, while remaining domain-agnostic and applicable to other fields. A version of this paper with supplemental materials and code is available at `https://osf.io/f6phs`.

Keywords: time series analysis, pattern search, active learning

2000 MSC: 37M10, 68T10, 68U99

---

1. Introduction

Searching for recurrences of an event in chronologically recorded real-valued observations, known as time series [1, 2], is a ubiquitous task in various domains, such as retrieving interesting events in IoT measurements [3, 4] detecting pathogenic fragments in genomic sequences [5, 6], identifying similar market behavior in historical stock prices [7, 8], etc. Our automotive engineers frequently trace event-induced patterns in sensor measurements as

preliminary work for root cause diagnosis. These practical issues can be formulated as time series pattern search. It addresses the problem of finding patterns in a time series similar to given examples.

This problem is challenging owing to the obscurity of "similarity", which varies in use cases and can be subjective [9]. To address this problem, machine learning models are trained to capture the desired notion of similarity [10, 11]. The time series is fed piece by piece with a sliding window to the model that outputs the pattern relevance. Unfortunately, there are rarely sufficient ground truths to support model training in practice [12]. Accordingly, researchers invite users to verify patterns typically suggested by a tentative rule-based (meaning not involving model training) initial search [6, 13, 14]. We call such model-based user-in-the-loop time series pattern search "feedback-driven pattern search".

Existing methods for feedback-driven pattern search suffer from three weaknesses. **(W1) Fixed pattern length**: The size of the sliding window dictates that of the found patterns. To retrieve patterns of variable duration, the methods require multiple differently sized kernels, each scouring the data once. **(W2) Context-obliviousness**: The sliding window appraises the relevance of a pattern based on its internal characteristics, ignoring the context. The target event sometimes, unfortunately, lacks rich features on its own, but may be preceded or succeeded by distinctive signals that the user may not even notice. In that case, considering the data behavior in temporal proximity contributes to the retrieval. **(W3) Imprecise feedback**: The state-of-the-art allows the user to accept, reject, or eschew a found pattern, from which a positive, a negative, or no label for model (re)training is ex-

tracted. However, none of the options is satisfactory when a found pattern overlaps a target pattern partially, illustrated in Figure 2.

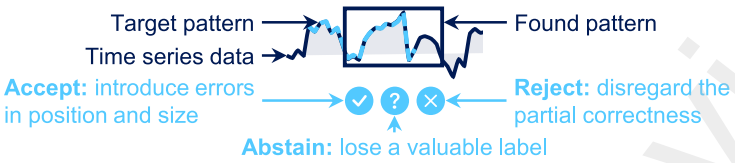

Figure 2: **Problem with Traditional Relevance Feedback.** User feedback is confined to acceptance, rejection, and abstention, which may place the user in a dilemma, where a found pattern is partially correct, and all options are controversial.

We present `NOOPS`, a data processing pipeline for feedback-driven pattern search. It features two techniques to overcome the aforementioned weaknesses. First, instead of classifying the patterns as relevant or not directly, we propose to classify if each time step in the time series is located in a target pattern. The "in-target-pattern" time steps are then merged into found pattern intervals. We call this approach **nonmyopic search** because, as we will show, it recognizes patterns of different sizes and even peeks into their context. This improvement overcomes W1 and W2, increasing retrieval accuracy by 19% and decreasing execution time by 68% in our experiments. Second, to overcome W3, we propose to solicit feedback on more features than relevance/correctness, specifically, the positions and sizes of the found patterns. We name this approach **multi-feature feedback**. We will verify that the minor extra labeling effort pays off, raising the retrieval accuracy by 35% in our experiment.

As an auxiliary work, we developed a Visual Query System (VQS) demonstrated in the video `https://youtu.be/7-MPiC7asBg`.

2. Related Work

Similarity Measures. Time series pattern search algorithms, especially the rule-based ones, often rely on a similarity measure (alternatively a distance measure) to gauge the relevance of a pattern with respect to the query (a given example of the target event). The similarity measure plays a decisive role in retrieval accuracy [15]. Basic options include cosine similarity [16], Mahalanobis distance [17], and variants of the L^p norm, such as Euclidean Distance (ED) [18], and Chebyshev distance [19]. To capture temporally warped patterns, researchers proposed elastic similarity measures, where Dynamic Time Warping (DTW) and its variants [20, 21] are the most widely accepted and proven hard to beat for time series classification [22]. Sadly, "no single algorithm accounts for human judgments of time series similarity" [9]. Hence, many works employs a model in place of the rule-based similarity measure to assess the pattern relevance [10, 11]. Our data processing pipeline begins with a rule-based initial search and proceeds with the model-based feedback-driven search. The initial search uses ED as the similarity measure, specifically with Mueen's Algorithm for Similarity Search (MASS), the fastest pattern search algorithm so far [18], due to its speed, the primary concern of the initial search. The feedback-driven search requires a model and aims at a higher retrieval accuracy.

Scanning Approaches. The similarity measure or the model discussed in the previous paragraph requires a scanning approach to traverse the time series data. The classic approach runs a sliding window to check every position in the time series. At each position, it estimates the relevance of the pattern that starts from the position and spans the window size. Faloutsos et

al. named this approach "sequential scanning" [23]. This approach cannot retrieve patterns of variable sizes efficiently. To address this problem, researchers proposed various solutions. Some works match pieces in the query with pieces in the time series with the same characteristics (e.g., monotonicity), and a matching pair of pieces may differ in size [24]. It is also possible to formulate the query as a regular expression searching a symbolic representation of the time series, where quantifiers in the regex describe scaling along the time axis [25, 26]. There are also works proposing computational techniques [27]. However, these approaches only apply to certain rule-based similarity measures. For model-based pattern search, sequential scanning is the only option in the literature [6, 14], resulting in W1 and W2. Our scanning approach breaks this convention and embraces nonmyopic search. It cooperates virtually with all classification models and overcomes the two weaknesses, achieving higher accuracy and speed.

Active Learning. Model training requires adequate labels. In practice, labels are scarce. Researchers have proposed relevance feedback / active learning [28], which invites users to label the outcomes, possibly in multiple iterations. Since its introduction in time series analysis in [29], this technique has also gained attention in time series pattern search [6, 13, 14]. However, the state of the art only allows yes/no/indecisive as feedback on the relevance of the found patterns, exposing W3. We propose multi-feature feedback on the positions and sizes of the found patterns, which solves W3, yielding substantially higher retrieval accuracy.

Our proposed algorithm and the user interface constitute a VQS. Please refer to Appendix A for related work on VQSs.

3. Method

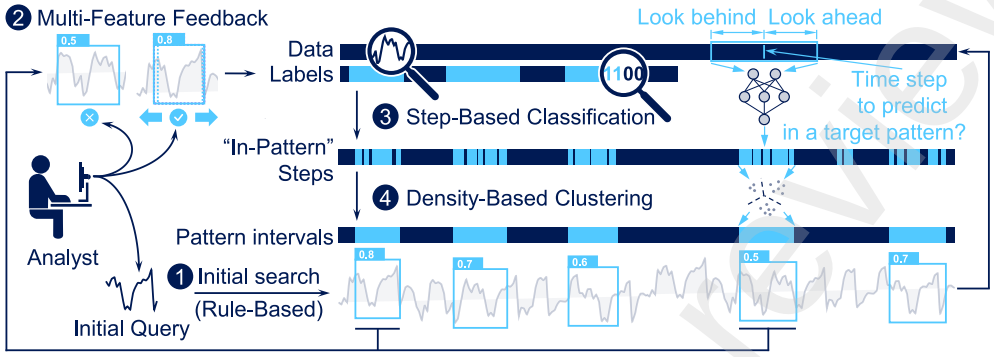

Figure 3: **Data Processing Pipeline.** NOOPS combines time-step-based classification and density-based clustering to retrieve patterns of variable duration and be aware of their context. Moreover, NOOPS extends traditional relevance feedback by feedback on positions and sizes of the found patterns, boosting retrieval accuracy even further.

A time series $S = \left[s_i \in \mathbb{R}^d\right]_{0 \le i < n}$ of length n (number of time steps) and dimension d (number of tracks/channels) is an arrangement of n real-valued vectors of size d. We adopt the matrix notation with a pair of square brackets for an array of indexed entries. The sub-script indicates the range of the element indexes. We use the set notation with a pair of curly brackets for a group of not necessarily ordered entries without duplicates, such as all found patterns. We define 1) a subsequence $S\,(a, b)$ of S, starting at a-th time step inclusive and ending at b-th time step exclusive as $S(a, b) = [s_i]_{a \le i < b}$; 2) a query of length w as $Q = \left[q_i \in \mathbb{R}^d\right]_{0 \le i < w}$; and 3) a relevance function $f_{rel}\colon \mathbb{R}^{(b-a)\times d} \to [0, 1]$, which takes $S\,(a, b)$ as input and returns its relevance/confidence. Please refer to Appendix B for a list of symbols.

We define the task time series pattern search as follows: given a time series S and one or more target pattern examples, return all subsequences $S\,(a, b)$ in S whose confidence is no less than a threshold $\tau \in (0, 1)$, namely,

return $\{S(a,b) \mid f_{rel}(S(a,b)) \geq \tau\}$.

We propose a four-stage data processing pipeline for feedback-driven pattern search in multivariate time series, illustrated in Figure 3.

Stage ❶ searches for the query Q based on an established rule-based method MASS [18] to obtain initial results. Though we focus on feedback-driven search, this stage is indispensable since there are usually insufficient labels to begin with model training immediately. If the found patterns in this stage fail to reach the user's desired quality, they still serve as label candidates for the subsequent feedback-driven search.

Stage ❷ gathers user feedback. During this stage, domain knowledge flows into the pipeline, which reflects the user's notion of relevance. Compared to traditional feedback on the correctness of the found patterns alone, we allow the user to edit the positions and sizes of the found patterns and call this type of augmented feedback "multi-feature feedback". With multi-feature feedback, the user can confidently endorse a found pattern that matches a target pattern partially, after correcting its position and size. Otherwise, the user may accept it hesitantly, fearing a deterioration of the label quality; or reject it regrettably, as a desired pattern is touched at least.

Stage ❸ and Stage ❹ form "nonmyopic" search.

Stage ❸ uses the user labels obtained in Stage ❷ to train a model. In traditional search, the model classifies patterns directly as relevant or not. The input of the model is a pattern P, and the raw output of the model is the probability that P is relevant $f_{rel}(P)$. In contrast, nonmyopic search trains a model to classify if each time step in the time series resides in a target pattern. The input of the model is $S(i-h,\, i+h+1)$ around the

time step s_i to classify. The raw output of the model is the probability that s_i is located in a target pattern $\widetilde{f}_{rel}\left(S\left(i-h,\, i+h+1\right)\right)$, where we use $\widetilde{f}_{rel}\colon \mathbb{R}^{2h+1} \times d \to [0,1]$ instead of f_{rel} to indicate the usage of the model for estimating the relevance of time steps, rather than relevance of patterns. This alteration brings many benefits to be elaborated in Section 3.1.

As the user expects relevant patterns in whole pieces instead of time steps in relevant patterns, Stage ❹ merges (the indexes of) the "in-target-pattern" time steps into found patterns intervals $\{(a,b) \mid S(a,b) \in \{F\}\}$ with density-based clustering, where $\{F\}$ denotes all found patterns.

Because Stage ❶ is not our focus, we keep its detailed explanation in Appendix C. Subsequently, we will first explain Stage ❸ and Stage ❹, followed by Stage ❷ because we consider nonmyopic search (Stage ❸ + Stage ❹) the major contribution. Moreover, the exposition of multi-feature feedback (Stage ❷) presumes the understanding of nonmyopic search.

3.1. Time-Step-Based Classification: The Adaptive Core

In `NOOPS`'s pipeline in Figure 3, Stage ❸ serves as the adaptive core. Traditional model-based pattern search classifies time series patterns straightforwardly. To attain patterns of different sizes, the model has to go through multiple resampled versions of the time series S. Otherwise, the input size of the model would dictate the length of the found patterns.

We propose training the model to estimate the probability that a time step is "in-target-pattern" or "not-in-target-pattern" (hereinafter called positive and negative time step, respectively). The confidence of all time steps is

$$\widetilde{C} = \left[\widetilde{c}_i = \begin{cases} \widetilde{f}_{rel}\left(S\left(i-h_1,\, i+h_2+1\right)\right), & \text{if } h_1 \le i < n-h_2 \\ 0, & \text{otherwise} \end{cases}\right]_{0 \le i < n} \tag{1}$$

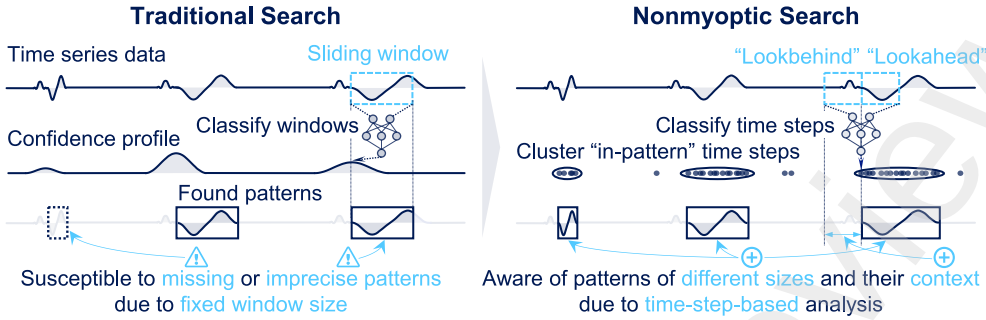

Figure 4: **Traditional Search vs. nonmyopic Search.** Traditional pattern search scans the time series with a sliding window to get a confidence profile, where top local maxima correspond to the found patterns. It has problems with patterns of variable duration and is oblivious to the pattern context. Nonmyopic search classifies whether each time step is within a target pattern and clusters the "in-target-pattern" time steps to retrieve pattern intervals. It finds patterns of variable lengths in one scan and is aware of the pattern context.

where we pad the confidence with 0 (especially for $0 \leq i < h_1$) so that the time step indexes do not shift. The indexes of the positive time step are

$$B = \left\{ i \mid \widetilde{f}_{rel}\left(S\left(i - h_1,\, i + h_2 + 1\right)\right) \geq \tau \right\} \tag{2}$$

Recall that $\widetilde{f}_{rel}$ introduced at the beginning of Section 3 estimates the probability that a time step belongs to a target pattern or not, based on the behavior of the neighboring data. h_1 and h_2 are hyperparameters that we call "lookbehind" and "lookahead", respectively, because they resemble the positive lookbehind and positive lookahead in regular expression. In practice, we set $h_1 = h_2 = h$ and call h "lookaround". It is set with the average length of the positively labeled found patterns from Stage ❷.

The concrete implementation of $\widetilde{f}_{rel}$ differs slightly for each model. For instance, some models, like `CNN` and `LSTM`, support multivariate time series out of the box and can take $S\left(i - h_1, i + h_2 + 1\right)$ as their input unmodified. Other models, like `Random Forest` and `XGBoost`, require flattening $S\left(i - h_1, i + h_2 + 1\right)$ before feeding it to the model. How to properly han-

dle each model is out of the scope of this work. Our model configuration is documented in Appendix D.3.

It is worth noting that time-step-based classification is a standard task for the famous EEG Eye State dataset [30] and has witnessed a great success [31, 32]. It inspired us to transfer the technique to time series pattern search.

The proposed change from pattern-interval-based to time-step-based classification brings three benefits. First, it can retrieve patterns of variable sizes in one scan, while the traditional search requires a series of differently-sized sliding window kernels. The time-step-based classification breaks the connection between the model input size and the found pattern size. It still processes the data with a sliding window of size $h_1 + h_2 + 1$, however, the window size does not dictate the found pattern size. Thanks to this decoupling, time-step-based classification only looks at the data once.

Second, our approach is aware of the pattern context. As illustrated in Figure 4, depicted by the arrow pointing out from the symbol ⊕ above the text "context", when classifying the initial time steps of a target pattern, the lookbehind extends into the historical data on the left side of the pattern. Likewise, the lookahead will gradually probe into the future data beyond the end time step of the pattern, as the sliding window approaches the right boundry of the pattern. Therefore, when retrieving a found pattern $F = S(a, b)$, the search takes an extended data fragment $S(a - h_1, b + h_2)$ into consideration, including the context $S(a - h_1, a) \cup S(b, b + h_2)$. This context awareness is helpful when there is characteristic information preceding or succeeding the target patterns. This is for instance the case when the target event represents a phase in a multi-phase process surrounded by

more distinctive phases, e.g. in our `Deep Valve Linear Phase` use case illustrated in Figure 8. Whereas, if the context is uninformative, the model learns to ignore it anyway.

Third, model training becomes more robust. In traditional search, one labeled pattern $S(a, b)$ serves as a single training instance. In the time-step-based classification, it corresponds to $b-a$ training instances $[S(i-h_1,\ i+h_2+1)]_{a\leq i<b}$. These training instances depict the pattern from diverse temporal perspectives, increasing the robustness of the training, reflected in the much smaller standard deviation of the retrieval accuracy over multiple runs in our experiments in Section 4.2. Moreover, it enables us to perform a data augmentation to further increase the number of labeled training instances. If $S(a, b)$ is a positively labeled pattern and its start time step index a and end time step index b are precise, then, it should be surrounded by negative time steps. Namely, we can use $[S(i-h_1,\ i+h_2+1)]_{a-\widetilde{h}\leq i<a \text{ or } b\leq i<b+\widetilde{h}}$ as negative training instances, where $\widetilde{h}$ is a hyperparameter which we set as h. Not only does this data augmentation increase training instances, but it also releases the user from the burden of providing negative labels. Traditional feedback-driven search, on the other hand, forces the user to provide negatively labeled patterns because the model cannot be trained only with positive training instances. On the downside, training requires more time due to the increase of training instances. This, however, is less of a concern compared to inference time, as we will see in Section 4.2. Training time depends on the number of labels, which is typically small; while the inference time scales linearly with the length of the time series, which is generally large. Therefore, as data volume increases, inference time dominates.

3.2. Density-Based Clustering: From Steps to Intervals

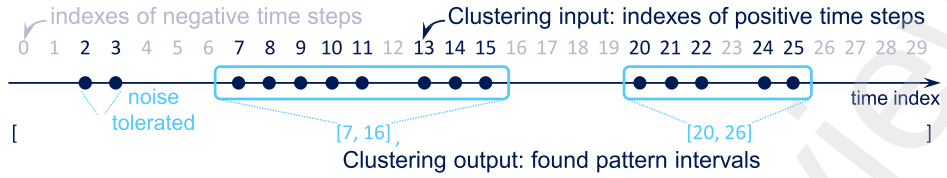

Figure 5: **A Canonical Example of Clustering Time Steps.** We merge the positive time steps to found pattern intervals because the user expects found patterns in whole pieces. We can achieve the merging with density-based clustering because the positive time steps from the same pattern form a cluster along the time axis.

Identifying time steps in target patterns from Stage ❸ is one step closer to the desired output. However, the user expects the found patterns, not individual time steps. Accordingly, Stage ❹ converts the positive time steps B from Equation (2) into found pattern intervals $\{(a, b) \mid S(a, b) \in \{F\}\}$, where $\{F\}$ denotes all found patterns.

We cannot blindly merge consecutive positive time steps due to inevitable misclassifications. For instance, in Figure 5, it is likely that Stage ❸ misclassified Time Step 2 and 3 as positive, while Time Steps 12 and 23 as negative. False positive time steps can result in incorrect found patterns, and false negative time steps split a target pattern into several smaller found patterns. The merging process must tolerate such errors.

We address this problem using one-dimensional density-based clustering. This approach is suitable because, along the time axis, a target pattern appears as a cluster of densely packed positive time steps. In contrast, the positive time steps between target patterns are much sparser and will be disregarded by the density-based clustering method out of the box.

Density-based clustering distinguishes itself from partition-based and hierarchical clustering by not necessitating prior knowledge of the number of

clusters or levels. Nevertheless, this property is not necessarily an advantage due to other parameters affecting the number of clusters / found patterns.

In practice, we adopt Density-Based Spatial Clustering of Applications with Noise (DBSCAN) with parameters in Appendix D.3. The input of DBSCAN contains the indexes of the positive time steps B, and the output gives the found pattern intervals $\{(a,b) \mid S(a,b) \in \{F\}\}$, where $\{F\}$ denotes all found patterns. Finally, the confidence of a found pattern $F = S(a,b)$ is the average confidence over all time steps in F, i.e., $f_{rel}(F) = \frac{1}{b-a}\sum_{i=a}^{b-1} \widetilde{c}_i$, where $\widetilde{c}_i \in \widetilde{C}$ from Equation (1) is the confidence of i-th time step.

3.3. Multi-Feature Feedback: Procuring Training Labels

Stage ❷ in Figure 3 closes the proposed pipeline. It provides labels for model training while infusing the user's domain knowledge into the pipeline.

Conventional active learning allows user feedback on relevance alone. For each selected found pattern, it asks the user to accept/reject/skip it. Then, it uses the accepted found patterns as positive labels and the rejected ones as negative labels during model training, while omitting the skipped ones. This can lead to a dilemma illustrated in Figure 2. If accepted, the found pattern as a positive label may confuse the model with its imperfection; if rejected, its partial correctness could be misleading; if skipped, the partial correctness of the found pattern is unutilized.

Combined with nonmyopic search, this imprecision in user feedback introduces mislabeled training instances directly. Imagine that the user accepts a found pattern $F = S(a,b)$. This single user label would provide $b-a$ positive training instances $[S(i-h_1,\, i+h_2+1)]_{a \le i < b}$. However, the user may actually desire the target pattern $T = S(c,d)$ that overlaps F. Suppose

$a < c < b < d$, i.e., F starts and ends too early. To be precise, F overlaps with T on $S(c, b)$, but also wrongly detects the fragment $S(a, c)$ and misses $S(b, d)$. This user feedback correctly labels all time step in $S(c, b)$ (corresponding to training instances $[S(i - h_1,\, i + h_2 + 1)]_{c \le i < b}$) as positive, but also incorrectly labels all time step of $S(a, c)$ (corresponding to training instances $[S(i - h_1,\, i + h_2 + 1)]_{a \le i < c}$) positive.

To address this problem, we propose multi-feature feedback, not only on the relevance of the found patterns but also on their positions and sizes. Pertaining to the same example, if the algorithm suggests F, the user should be able to notice T in the UI as well, otherwise, the user would not have accepted F. The user can adjust the position and size of F to align with T, so that $a_{new} := c$, $b_{new} := d$, i.e., $F_{new} := T$. In this way, we resolve the predicament caused by the imprecision of the traditional relevance feedback.

With this adjustment, the user may even accept a found pattern of low quality that overlaps only a little with a target pattern and should have been rejected by traditional relevance feedback. As long as the user notices the target pattern, the user can virtually use the target pattern instead of the imprecise found pattern as a positive label.

We have designed a user interface to assist the user with query definition, result inspection & analysis, and feedback provision. The UI is necessary because we are dealing with feedback-driven pattern search involving user interaction. However, the UI is not our focus in this work and we omit it due to the space limit. Please refer to Appendix E for a comprehensive explanation of the UI features and to Appendix I for a cohesive workflow.

4. Evaluation

We make two contributions to feedback-driven pattern search: 1) nonmyopic search combining time-step-based classification and density-based clustering; 2) multi-feature feedback on the positions and sizes of the found patterns besides their correctness. Accordingly, we benchmarked 1) nonmyopic search against traditional search on retrieval accuracy and speed; 2) multi-feature feedback against binary relevance feedback on retrieval accuracy.

We used three labeled datasets with different characteristics in the experiments. Among them, the `Deep Valve` dataset has two use cases, each with unique target patterns. Please refer to Appendix F for plots of the data, the queries, the ground truths, as well as their physical background.

We conducted all experiments on a standard laptop with 64-bit Windows 10 Enterprise, Intel i7-8650U, 16 GB RAM, and 1 TB HDD. Details of hardware, software, and parameter settings can be found in Appendix D.

4.1. Proposing A Novel Metric on Time Series Retrieval Accuracy

Numerous methods for time series pattern search have been proposed. However, there is a lack of a standardized evaluation approach or metric to assess the quality of the identified patterns. This is partially understandable because few labeled public datasets exist for time series pattern search. One common evaluation method is visually inspecting the found patterns [6, 33]. Another possibility is to measure performance decrease against a more accurate baseline method [27, 5], which is suitable when the proposed algorithm is an accelerated version of the baseline. Luckily, our engineers have provided ground truths. Nonetheless, we need to establish an evaluation metric given

Figure 6: **Time-Step-Based vs. Pattern-Interval-Based Metrics.** There are cases where the found patterns do not match the target patterns well, yet the time steps are all correctly classified, as in ⓑ, where the algorithm fails to recognize a target pattern as a whole piece. The time-step-based metrics will arrive at high scores despite this problem. The pattern-interval-based metric Average Precision (AP) avoids such pitfalls.

ground truths.

Suppose a time series $S = [s_i]_{0 \le i < n}$ comes with ground truths $\{G\}$, and the pattern search algorithm retrieves found patterns $\{F\}$. The evaluation metric on retrieval accuracy should gauge how well $\{F\}$ aligns with $\{G\}$.

We conceive two approaches to designing the metric: time-step-based and pattern-interval-based.

For the time-step-based metric, we first need to derive whether each time step in S is located in a found pattern, i.e.,

$$Y = \left[y_i = \begin{cases} 1, & \text{if } a \le i < b,\ S(a,b) \in \{F\} \\ 0, & \text{otherwise} \end{cases} \right]_{0 \le i < n} \tag{3}$$

Likewise, we can derive whether each time step in S is in a ground truth

$$Z = \left[z_i = \begin{cases} 1, & \text{if } a \le i < b,\ S(a,b) \in \{G\} \\ 0, & \text{otherwise} \end{cases} \right]_{0 \le i < n} \tag{4}$$

Subsequently, we can count 1) the number of time steps both in a ground truth and in a found pattern (true positives) $TP = |\{i \mid y_i = z_i = 1\}|$, 2) the number of time steps in a found pattern but not in a ground truth (false posi-

tives) $FP = |\{i \mid y_i = 1,\, z_i = 0\}|$, and 3) the number of time steps in a ground truth but not in a found pattern (false negatives) $FN = |\{i \mid y_i = 0,\, z_i = 1\}|$ Finally, we can calculate the $precision = \frac{TP}{TP+FP}$, $recall = \frac{TP}{TP+FN}$, and $F_1 = \frac{2 \cdot precision \cdot recall}{precision+recall}$.

Time-step-based metrics may lead to pitfalls like in Figure 6 ⓑ, where the algorithm fails to recognize the target pattern as a single piece, despite the correct classification of every time step and hence perfect scores.

We propose, therefore, to transfer the metric Average Precision (AP) [34] from object detection in computer vision to pattern search in time series. It operates on pattern intervals instead of time steps.

AP requires an Intersection over Union (IoU) threshold υ to judge whether a found pattern is relevant based on its matching ground truth. The IoU of two patterns P_1 and P_2 is $IoU\,(P_1, P_2) = \frac{|P_1 \cap P_2|}{|P_1 \cup P_2|}$. AP pronounces a found pattern F^* true positive if 1) F^* overlaps a ground truth G by at least υ; and 2) G does not overlap another found pattern F' of higher confidence by at least υ (otherwise, the more confident F' has already consumed G). Together, F^* is added as a new true positive if $\exists G \in \{G\}$ s.t. $IoU\,(F^*, G) \geq \upsilon$ and $\forall F' \in \{F' \in \{F\} \mid IoU\,(F', G) \geq \upsilon\}$, $f_{rel}\,(F^*) \geq f_{rel}\,(F')$.

The IoU threshold υ regulates the strictness of the metric. Figure 6 ⓐ shows a series of IoUs. We used mainly AP30 and AP50 with IoU threshold 30% and 50% respectively and interpreted AP30 as "roughly caught" and AP50 as "precisely located". Please note that in computer vision, the IoU threshold for AP is usually at least 50% and can be as high as 95%. According to our findings, however, this thresholding can be too strict for time series pattern search, and the scores of most results (including and especially results

from baseline methods) would be too low and not well separated in $[0, 1]$, making result comparison difficult. Figure 6 ⓒ provides an example, where the found pattern has 33% IoU with a ground truth that does not overlap with another found pattern. The found pattern is, consequently, a true positive according to AP30, but a false positive according to AP50.

In conclusion, while advocating time-step-based classification, we prefer a pattern-interval-based metric to measure retrieval accuracy, because it aligns better with the desired outcome format of the task pattern search, i.e., found pattern (intervals) in whole pieces rather than individual time steps.

4.2. Nonmyopic Search vs. Traditional Search

This section evaluates the proposed nonmyopic search in terms of retrieval accuracy and speed in a quantitative comparison with traditional search.

We included four models in this experiment: 1) `Random Forest`: the champion of retrieval accuracy in traditional search according to Lekschas et al. [6]; 2) `XGBoost`: the winner in our preliminary model selection both in retrieval accuracy and speed; 3) `LSTM`: one of the most popular models for time series analysis; 4) `1D-CNN`: a common alternative to RNNs for time series analysis [35]. Nonetheless, we did not intend to find the best model since the goal is to benchmark two pipelines. Rather, we included multiple models to avoid judgments biased by artifacts resulted from model choices.

Model training requires labels. We used the first (sorted in ascending order of start time) several ground truths as positive labels for both nonmyopic and traditional search. For negative labels, due to the different mechanisms of label usage by nonmyopic and traditional search, we provided different negative labels for them. For nonmyopic search, we simply provided two

or three relatively long negative labels. This is the desired way of providing negative labels for nonmyopic search, basically to show the background. Otherwise, negative labels are optional anyway due to the data augmentation specific to nonmyopic search explained in the thrid benefit of nonmyopic search in Section 3.1. For traditional search, negative labels are mandatory. Meaningful negative labels should be of comparable size to the positive labels or target patterns. Therefore, we provided at least five and up to twenty negative labels chosen randomly, depending on the total number of ground truths. The labels can be found in Appendix G.3.

Figure 7 shows the experiment results, one diagram per use case. We omitted `LSTM` because it is significantly slower than the other models, causing a skewed visual representation for other models. Diagrams with `LSTM` can be found in Appendix G.2. In the diagrams, the abscissae represent wall-clock time, and the ordinates retrieval accuracy measured by AP30. Thus, the overall performance increases from bottom right to top left in a diagram. We chose AP30 because it spreads the metric scores in the diagrams relatively well between 0 and 1. The light points represent the models in nonmyopic search, while the dark linked points the models in traditional search. Similar to the rule-based initial search introduced in Appendix C, the performance of traditional model-based pattern search depends on the number of sliding window kernels to retrieve patterns of different sizes. Hence, we measured the performance of a model in traditional search with different numbers of sliding window kernels and used linked points or lines to represent them. We conducted each accuracy measurement five times, averaged the results, and included vertical error bars in the diagrams to represent the standard

deviations across these five trials. Each speed measurement is averaged over ten repetitions. We did not conduct accuracy and speed measurements simultaneously because the former incurs more costs like generating plots and saving results during retrieval, which were excluded from the latter.

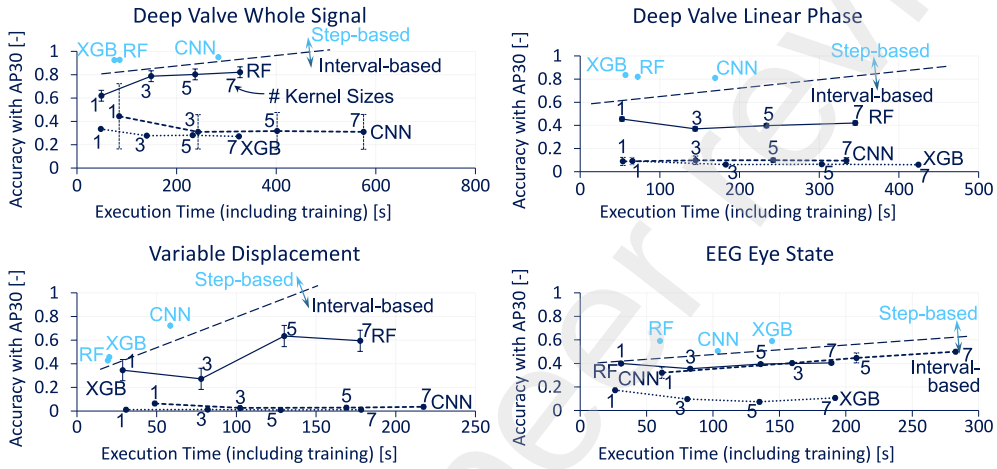

Figure 7: **Comparison on Retrieval Accuracy and Speed.** Compared to the linked dark points denoting traditional search with different numbers of sliding window kernels, the light points for nonmyopic search are further to the upper left in each plot, indicating an improvement in retrieval accuracy and speed. (RF: `Random Forest`, XGB: `XGBoost`)

With the same model, the data points representing nonmyopic search are generally higher than those representing traditional search, indicating consistently higher retrieval accuracy with nonmyopic search. Indeed, many of the data points for traditional search are relatively low, suggesting that it is ineffective with certain models in specific scenarios.

Furthermore, nonmyopic search is more robust because 1) we can hardly discern the error indicator bars around the light points, while the error indicator bars are prominent around the dark points, implying that traditional search may produce more inconsistent results in different runs; 2) the light points of different models in the same diagram are relatively of the same

height while the dark lines of different models in the same diagram are distributed vertically more widely, hinting at the robustness of nonmyopic search with regard to the model choice. We attribute the improvement in robustness primarily to the increase in the number of training instances elaborated in the third benefit of nonmyopic search in Section 3.1.

In the `Deep Valve Linear Phase` use case, all models in traditional search performed poorly because the internal characteristics of the target event (simply a linear phase) lack distinctive features. However, the target pattern is a phase in a multi-phase process illustrated in Figure 8. Together with its preceding and succeeding phases, the pattern becomes easier to detect. As a result, nonmyopic search aware of the pattern context maintains a high retrieval accuracy for all models.

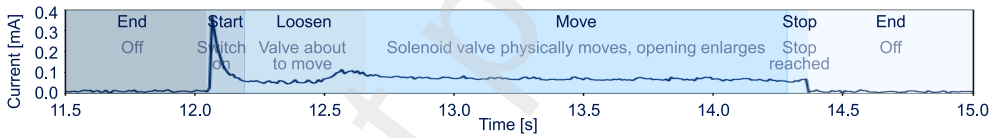

Figure 8: **An Operation Cycle in the `Deep Valve Linear Phase` Use Case.** Hidden in a five-phase process, the target event is the linear phase titled "Move". Though it has few conspicuous features per se, its context (the phases titled "Start", "Loosen", and "Stop") has distinctive shapes and can contribute to identifying the target pattern.

Not only is nonmyopic search more accurate and robust, but also faster because it needs to scan the data only once, while traditional search multiple times to find patterns of different sizes. As Figure 7 verifies, the light points are generally positioned further to the left than the dark lines.

Figure 9 shows the training and inference time in nonmyopic search and traditional search with various numbers of sliding window kernels, revealing more insights. We show only the result for `Random Forest`. The qualita-

tive relationships between different parts of execution time are similar in other use cases. In the `Deep Valve Whole Signal` use case, the inference time for nonmyopic search, at 39.92 ± 0.73 seconds, is comparable to that of traditional search using a single sliding window kernel, which can only identify patterns of a fixed size, at 48.74 ± 1.47 seconds. Both methods scan the data only once. Compared to traditional search with five sliding window kernels, which finds patterns of variable duration in a coarse manner and takes 237.15 ± 4.31 seconds, nonmyopic search significantly reduces the time required. On the other hand, nonmyopic search requires considerable training time (45.73 ± 0.91 seconds) due to the massive increase of training instances. In comparison, training time is negligible for traditional search, only 0.16 ± 0.01 seconds. However, increased training time in nonmyopic search is of less concern for larger data, because it is mainly determined by the (usually small) number of labels, while inference time scales linearly with (the generally large size of) the time series data. Therefore, for larger datasets, the inference time will dominate and should be the major concern in terms of execution time.

If restricting the experiment to `Random Forest` (the best model for traditional search in terms of retrieval accuracy and speed) and the number of sliding window kernels to five (which should be coarse enough; we use eight sliding window kernels in production in rule-based initial search), nonmyopic search brought 19% retrieval accuracy boost (absolute difference) in 68% less execution time, as shown in Table G.6.

In addition to the primary discovery, we report the following secondary findings like model comparison in Appendix G.2.

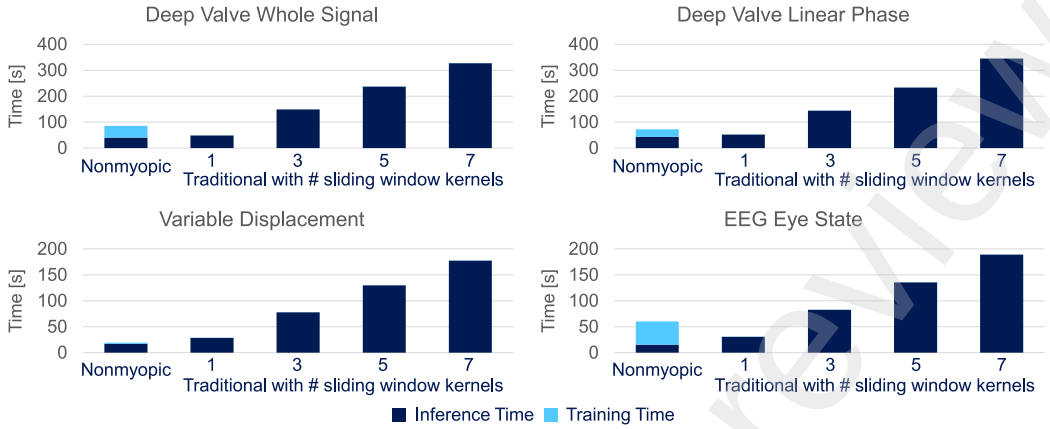

Figure 9: **Training and Inference Time with `Random Forest`.** The diagrams show the training and inference time in nonmyopic and traditional search. For one thing, the inference time of nonmyopic search is comparable with traditional search with one sliding window kernel, both scanning the data only once. Whereas, the inference time in traditional search scales roughly proportionally to the number of sliding windows, making it less attractive when pattern lengths vary significantly. For another, while the training time in traditional search is negligible compared to inference time, the training time in nonmyopic search is considerable due to the massive increase of training instances.

In summary, nonmyopic search improves retrieval accuracy and speed for feedback-driven pattern search. Please refer to Appendix G for more details.

4.3. Multi-Feature vs. Binary Relevance Feedback

This section evaluates the proposed multi-feature feedback on retrieval accuracy in a quantitative comparison with traditional relevance feedback. Conceptually, the former is advantageous as it encompasses the capabilities of the latter at least. However, we aim to show that the minor extra effort with multi-feature feedback pays off far more than expected. We used nonmyopic search with `Random Forest` due to its strong performance in Section 4.2.

Designing experiments involving user feedback is complicated due to the entailed manual labor and uncertainty. We propose to simulate user behavior, similar to [36, 14]. Specifically, we created two agents, the BR-Agent

and MF-Agent, to emulate Binary Relevance and Multi-Feature feedback, respectively, depicted in Figure 10.

The BR-Agent accepts a found pattern $F = S(a, b)$ as a positive label if: 1) F overlaps a ground truth $G = S(c, d)$ no less than an IoU threshold υ^* (set to 30%), i.e., $\exists G \in \{G\}$ s.t. $IoU(F, G) \geq \upsilon^*$, where $\{G\}$ denotes all ground truths; 2) To avoid duplicate labels, F should not overlap an already added positive label F' by υ^*, i.e., $\forall F' \in \{F\}$, $IoU(F, F') < \upsilon^*$, where $\{F\}$ denotes all found patterns before adding F as a new positive label. If F fails to fulfill the first criterion, the BR-Agent labels F negative. If F meets the first criterion but not the second, the BR-Agent ignores the pattern.

The MF-Agent adds a found pattern F as a positive label as long as it intersects with a ground truth G, regardless of how small their IoU is, i.e., $\exists G \in \{G\}$ s.t. $F \cap G \neq \emptyset$. This behavior assumes that the user can notice the touched target pattern upon seeing the found pattern. Moreover, MF-Agent adds G instead of F as a positive label if G is not already added, which simulates the user's corrections on the position and size of the found pattern. The MF-Agent only labels a found pattern negative when it does not come into contact with any ground truth, i.e., $\forall G \in \{G\}$, $F \cap G = \emptyset$.

The experiment begins with the imperfectly retrieved patterns from the rule-based initial search detailed in Appendix C. Each agent assessed the found patterns based on the ground truths and returned five labels. Because `Random Forest` and `XGBoost` do not support iterative training with new labels only, we retrained all models with both the new and the old labels. We performed ten iterations of feedback-driven pattern search, allowing results to converge to stable retrieval accuracy in all use cases.

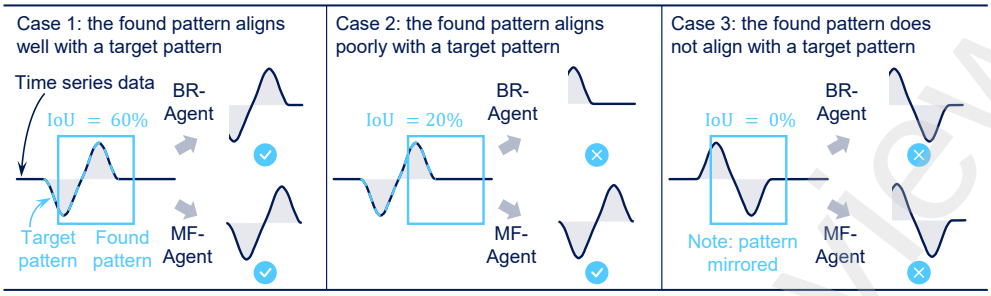

Figure 10: **BR-Agent vs. MF-Agent.** The BR-agent labels a found pattern positive when it overlaps a ground truth sufficiently, and negative when not. The MF-agent labels a found pattern positive, as long as it overlaps a ground truth (no matter how small the overlap is), and it adjusts the position and size of the found pattern to match the target pattern (use the ground truth instead of the found pattern as the positive label); the MF-agent only labels a found pattern negative when it does not overlap any ground truth.

Figure 11 shows the evolution of the retrieval accuracy according to AP30 and AP50 after the rule-based initial search and each of the ten feedback-driven search iterations. According to AP30, both feedback types can improve the search results from the rule-based initial search (Feedback Round 0 in Figure 11). In the `Deep Valve Whole Signal` use case, multi-feature feedback outperforms binary relevance feedback by a margin of 0.05 according to AP30. Other use cases witnessed a greater performance gain, with the light curves for multi-feature feedback much higher than the dark ones for binary relevance feedback. If measured by AP50, this discrepancy was even more evident. In fact, retrieval accuracy did not grow much with binary relevance feedback but still rose to a high level with multi-feature feedback.

To better compare the retrieval accuracy after convergence, we plot retrieval accuracy after ten feedback-driven search iterations with a series of APs in Figure 12. In all use cases, the light bars representing multi-feature feedback are higher than the dark bars representing binary relevance feedback, showing the general performance gain with the former. Moreover, the

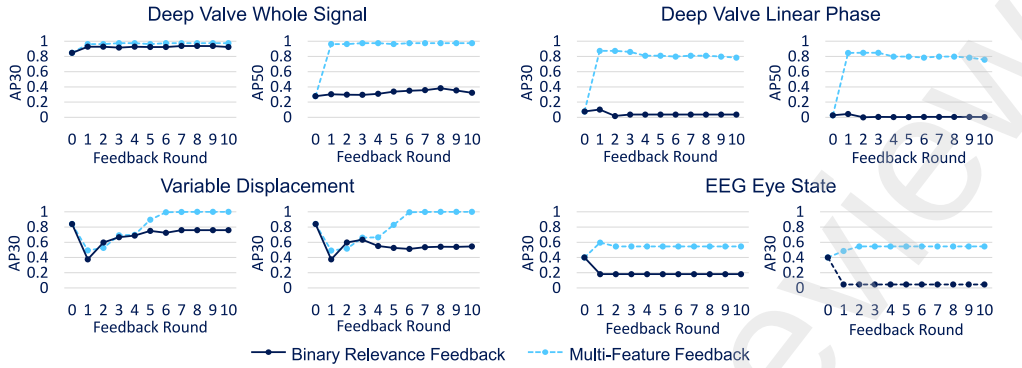

Figure 11: **Evolution of Retrieval Accuracy.** Both feedback types can raise retrieval accuracy, as indicated, e.g., by the rising AP30 in the `Deep Valve Whole Signal` use case. The improvement mainly happens after the first feedback round. Multi-feature feedback outperforms binary relevance feedback, both in terms of the correct hits according to AP30 and the quality of the found patterns according to AP50. In fact, binary relevance feedback fails to improve retrieval accuracy if measured by the stricter metric AP50.

stricter the metric, the larger the performance gain appears. Specifically, as metric strictness increases, scores for binary relevance feedback decrease rapidly, while those for multi-feature feedback decline only slightly, suggesting found patterns of higher quality with multi-feature feedback. In fact, binary relevance feedback failed in the more challenging `Deep Valve Linear Phase` and `EEG Eye State` use case.

As shown in Table H.7, multi-feature feedback raised retrieval accuracy on average by 35% and 59% according to AP30 and AP50, respectively.

In conclusion, multi-feature feedback outperforms binary relevance feedback and should be the preferred choice for feedback-driven pattern search, especially with nonmyopic search. Please refer to Appendix H.4 for a visual inspection of the found patterns after convergence of the retrieval accuracy.

Lastly, we conducted a case study to demonstrate the UI features and an expert study with two automotive engineers to validate its usability. They

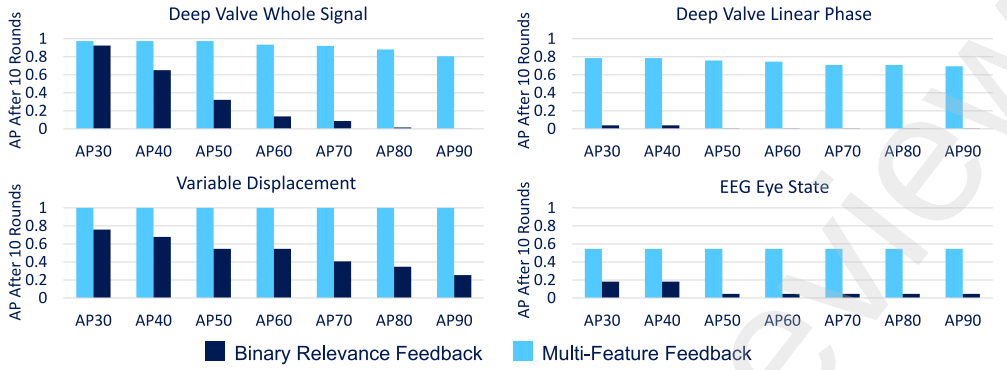

Figure 12: **Converged Retrieval Accuracy.** After ten feedback-driven search iterations, multi-feature feedback raised retrieval accuracy to a higher level than binary relevance feedback. The stricter the metric, the larger the performance gain appears.

are kept in Appendix I and Appendix J, respectively.

5. Discussion and Conclusion

In this work, we propose a data processing pipeline and tool called `NOOPS` for feedback-driven pattern search in multivariate time series. It features two contributions: 1) Nonmyopic search combining time-step-based classification and density-based clustering, which enables retrieving patterns of variable sizes, considers the pattern context, and makes model training more robust; 2) Multi-feature feedback, which comments not only on the correctness of the found patterns, as suggested by existing active learning methods, but also on their positions and sizes, leading to even higher retrieval accuracy. Surprisingly, the traditional pattern-interval-based feedback-driven search pipeline and the binary relevance feedback performed poorly in our experiments, and are thus in our view immature for general real-world applications. On the other hand, we did not encounter any significant limitations with the proposed nonmyopic search and multi-feature feedback. As such, we

urge researchers and practitioners conducting time series pattern searches, especially feedback-driven pattern searches, to try out our method.

In the future, we plan to improve `NOOPS`, especially in the following aspects.

Discriminate Overlapping Patterns. Nonmyopic search struggles with overlapping patterns, often merging them into a single long pattern, akin to semantic segmentation in computer vision. We are interested in transferring methods from object detection and instance segmentation into time series pattern search. In practice, however, overlapping occurrences are often irrelevant for many physically induced target events. For instance, it is physically meaningless for an opening-enlarging period of a valve to overlap another opening-enlarging period of the same valve.

Diversify Feedback Candidates. We present highly confident candidate patterns for feedback, which may lead to confirmation bias by focusing on a narrow range of subclasses. Therefore, we plan to explore recommending moderately confident candidates that differ from highly confident ones.

Externalize Knowledge. While we strive to seamlessly imbue the data processing pipeline with domain knowledge, knowledge-assisted visual analytics inspires us to externalize tacit expert knowledge, e.g., through semantic interaction analysis. We can imagine that the necessity of certain data processing operations, such as normalization, can be inferred from user feedback. The VQS can extract and manage such knowledge as explicit rules.

Our tool is in production, assisting automotive engineers in tracking events of interest in various measurements. In the near future, we would like to enhance `NOOPS` by optimizing its components and improving the VQS design to guide the user through the analytical workflow.

References

[1] G. E. P. Box, G. M. Jenkins, G. C. Reinsel, G. M. Ljung, Time Series Analysis: Forecasting and Control, John Wiley & Sons, 2015.

[2] M. F. Mbouopda, E. Mephu Nguifo, Scalable and accurate subsequence transform for time series classification, Pattern Recognition 147 (2024) 110121. doi:10.1016/j.patcog.2023.110121.

[3] M. Younan, E. H. Houssein, M. Elhoseny, A. E.-M. Ali, Performance analysis for similarity data fusion model for enabling time series indexing in internet of things applications, PeerJ. Computer science 7 (2021) e500. doi:10.7717/peerj-cs.500.

[4] H. Zhang, Y. Dong, J. Li, D. Xu, Dynamic time warping under product quantization, with applications to time-series data similarity search, IEEE Internet of Things Journal 9 (14) (2022) 11814–11826. doi:10.1109/JIOT.2021.3132017.

[5] T. Siddiqui, P. Luh, Z. Wang, K. Karahalios, A. Parameswaran, Shapesearch: A flexible and efficient system for shape-based exploration of trendlines, in: Proceedings of the ACM SIGMOD International Conference on Management of Data, ACM, New York, NY, USA, 2020, pp. 51–65. doi:10.1145/3318464.3389722.

[6] F. Lekschas, B. Peterson, D. Haehn, E. Ma, N. Gehlenborg, H. Pfister, Peax: Interactive visual pattern search in sequential data using unsupervised deep representation learning, Computer Graphics Forum 39 (3) (2020) 167–179. doi:10.1111/cgf.13971.

[7] O. Levchenko, B. Kolev, D. Yagoubi, R. Akbarinia, F. Masseglia, T. Palpanas, D. Shasha, P. Valduriez, Bestneighbor: Efficient evaluation of knn queries on large time series databases, Knowledge and Information Systems 63 (2) (2021) 349–378. doi:10.1007/s10115-020-01518-4.

[8] J. Liu, J. Li, L. Liu, Fastopm—a practical method for partial match of time series, Pattern Recognition 130 (2022) 108808. doi:10.1016/j.patcog.2022.108808.

[9] M. Correll, M. Gleicher, The semantics of sketch: Flexibility in visual query systems for time series data, in: IEEE Conference on Visual Analytics Science and Technology, 2016, pp. 131–140. doi:10.1109/VAST.2016.7883519.

[10] L. Hou, X. Jin, Z. Zhao, Time series similarity measure via siamese convolutional neural network, in: 12th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics, 2019, pp. 1–6. doi:10.1109/CISP-BMEI48845.2019.8966048.

[11] J. Li, C. Xu, T. Zhang, Similarity measure of time series based on siamese and sequential neural networks, in: 2020 39th Chinese Control Conference, IEEE, 2020, pp. 6408–6413. doi:10.23919/CCC50068.2020.9189261.

[12] L. Xi, Z. Yun, H. Liu, R. Wang, X. Huang, H. Fan, Semi-supervised time series classification model with self-supervised learning, Engineering Applications of Artificial Intelligence 116 (2022) 105331. doi:10.1016/j.engappai.2022.105331.

[13] T. Schlegl, S. Schlegl, D. Tomaselli, N. West, J. Deuse, Adaptive similarity search for the retrieval of rare events from large time series databases, Advanced Engineering Informatics 52 (2022) 101629. doi:10.1016/j.aei.2022.101629.

[14] Y. Yu, D. Kruyff, J. Jiao, T. Becker, M. Behrisch, Pseudo: Interactive pattern search in multivariate time series with locality-sensitive hashing and relevance feedback, IEEE Transactions on Visualization and Computer Graphics 29 (1) (2022) 33–42. doi:10.1109/TVCG.2022.3209431.

[15] A. Abanda, U. Mori, J. A. Lozano, A review on distance based time series classification, Data Mining and Knowledge Discovery 33 (2) (2019) 378–412. doi:10.1007/s10618-018-0596-4.

[16] M. R. Da Silva, O. A. de Carvalho, R. F. Guimarães, R. A. Trancoso Gomes, C. Rosa Silva, Wheat planted area detection from the modis ndvi time series classification using the nearest neighbour method calculated by the euclidean distance and cosine similarity measures, Geocarto International 35 (13) (2020) 1400–1414. doi:10.1080/10106049.2019.1581266.

[17] Z. Prekopcsák, D. Lemire, Time series classification by class-specific mahalanobis distance measures, Advances in Data Analysis and Classification 6 (3) (2012) 185–200. doi:10.1007/s11634-012-0110-6.

[18] Y. C.-C. Michael, Z. Yan, U. Liudmila, B. Nurjahan, D. Yifei, D. H. Anh, S. D. Furtado, M. Abdullah, K. Eamonn, Matrix profile i: All pairs similarity joins for time series: A unifying view that includes motifs,

discords and shapelets, in: IEEE 16th International Conference on Data Mining, 2016, pp. 1317–1322. doi:10.1109/ICDM.2016.0179.

[19] C. Georgios, S. Dimitris, P. Kostas, P. Themis, A. Spiros, S. Spiros, Efficient range and knn twin subsequence search in time series, IEEE Transactions on Knowledge and Data Engineering (2022). doi:10.1109/TKDE.2022.3167257.

[20] M. Shokoohi-Yekta, J. Wang, E. Keogh, On the non-trivial generalization of dynamic time warping to the multi-dimensional case, in: Proceedings of the SIAM International Conference on Data Mining, 2015, pp. 289–297. doi:10.1137/1.9781611974010.33.

[21] H. El Amouri, T. Lampert, P. Gançarski, C. Mallet, Constrained dtw preserving shapelets for explainable time-series clustering, Pattern Recognition 143 (2023) 109804. doi:10.1016/j.patcog.2023.109804.

[22] A. Bagnall, J. Lines, A. Bostrom, J. Large, 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) (2017) 606–660. doi:10.1007/s10618-016-0483-9.

[23] C. Faloutsos, M. Ranganathan, Y. Manolopoulos, Fast subsequence matching in time-series databases, in: Proceedings of the ACM SIGMOD International Conference on Management of Data, ACM, New York, NY, USA, 1994, pp. 419–429. doi:10.1145/191839.191925.

[24] M. Mannino, A. Abouzied, Expressive time series querying with hand-drawn scale-free sketches, in: Proceedings of the CHI Conference on

Human Factors in Computing Systems, ACM, New York, NY, USA, 2018, pp. 1–13. doi:10.1145/3173574.3173962.

[25] J. Rodrigues, D. Folgado, D. Belo, H. Gamboa, Ssts: A syntactic tool for pattern search on time series, Information Processing & Management 56 (1) (2019) 61–76. doi:10.1016/j.ipm.2018.09.001.

[26] Y. Yu, T. Becker, L. M. Trinh, M. Behrisch, Saxregex: Multivariate time series pattern search with symbolic representation, regular expression, and query expansion, Computers & Graphics 112 (2023) 13–21. doi:10.1016/j.cag.2023.03.002.

[27] B. C. Giao, 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) (2016) 181–196. doi:10.1007/s40595-016-0062-4.

[28] P. Ren, Y. Xiao, X. Chang, P.-Y. Huang, Z. Li, B. B. Gupta, X. Chen, X. Wang, A survey of deep active learning, ACM Comput. Surv. 54 (9) (2021). doi:10.1145/3472291.

[29] E. J. Keogh, M. J. Pazzani, An enhanced representation of rime series which allows fast and accurate classification, clustering and relevance feedback, in: The 4th International Conference on Knowledge Discovery and Data Mining, Vol. 98, 1998, pp. 239–243.

[30] O. Rösler, EEG eye state, accessed on 2024-02-01. (Jun. 2013). doi:10.24432/C57G7J.

[31] M. Nilashi, R. A. Abumalloh, H. Ahmadi, S. Samad, A. Alghamdi, M. Alrizq, S. Alyami, F. K. Nayer, Electroencephalography (eeg) eye state classification using learning vector quantization and bagged trees, Heliyon 9 (4) (2023). doi:10.1016/j.heliyon.2023.e15258.

[32] C. Zhang, C. Czado, Vine copula-based bayesian classification for multivariate time series of electroencephalography eye states, Journal of the Royal Statistical Society Series C: Applied Statistics (2023). doi:10.1093/jrsssc/qlad038.

[33] D. Folgado, M. Barandas, M. Antunes, M. L. Nunes, H. Liu, Y. Hartmann, T. Schultz, H. Gamboa, Tssearch: Time series subsequence search library, SoftwareX 18 (2022). doi:10.1016/j.softx.2022.101049.

[34] E. Zhang, Y. Zhang, Average precision, in: L. LIU, M. T. Özsu (Eds.), Encyclopedia of Database Systems, Springer US, Boston, MA, 2009, pp. 192–193. doi:10.1007/978-0-387-39940-9_482.

[35] F. Chollet, Deep Learning mit Python und Keras: Das Praxis-Handbuch vom Entwickler der Keras-Bibliothek, 1st Edition, mitp, Frechen, 2018.

[36] R. van de Schoot, J. de Bruin, R. Schram, P. Zahedi, J. de Boer, F. Weijdema, B. Kramer, M. Huijts, M. Hoogerwerf, G. Ferdinands, et al., An open source machine learning framework for efficient and transparent systematic reviews, Nature Machine Intelligence 3 (2) (2021) 125–133. doi:10.1038/s42256-020-00287-7.