

NOOPS: Nonmyopic Time Series Pattern Search

Yuncong Yu^{1,2}[0000-0002-2327-7737], Tim Becker¹[0000-0001-9615-0799], Peter Schichtel¹[0000-0002-7634-4964], Wolfgang Aigner³[0000-0001-5762-1869],
Alexandru C. Telea²[0000-0003-0750-0502], and Michael Behrisch²[0000-0002-1102-103X]

¹ IAV GmbH, Rockwellstr. 12, 76344 Gifhorn, Germany
{yuncong.yu,tim.becker,peter.schichtel}@iav.de

² Utrecht University, Princetonplein 5, 3584 CC Utrecht, Netherlands
{m.behrisch,a.c.telea}@uu.nl

³ University of Applied Sciences St. Pölten, Campus-Platz 1, A-3100 St. Pölten, Austria wolfgang.aigner@ustp.at

Abstract. We propose a pipeline for pattern search in multivariate time series. Existing machine-learning-based pattern search methods query time series with a sliding window, retrieving patterns of fixed sizes and assessing the pattern relevance based on the characteristics of the data within the window, oblivious to the pattern’s context. We propose **non-myopic search** - instead of direct pattern classification, it classifies whether each time point belongs to a target pattern and groups the in-target-pattern time points into pattern intervals using density-based clustering. This approach scans the data only once and retrieves variable-sized patterns, while being aware of the temporal context of the patterns. Experiments verify that nonmyopic search outperforms conventional search in retrieval accuracy and speed. Our method helps automotive engineers trace events of interest in ECU measurements, while the framework is open to other models and domains.

Keywords: Time series · Pattern search · Active learning.

1 Introduction

Searching for occurrences of an event in chronologically recorded real-valued observations, known as time series [3], is a ubiquitous task in various domains, such as tracking an incident in IoT measurements [20], detecting known pathogenic fragments in genomic sequences [10], identifying similar market behavior in historical stock prices [11]. These practical issues can be formulated as time series pattern search, which addresses the problem of retrieving intervals of patterns in a time series that are similar to the given examples.

This task is challenging due to the obscurity of “similarity”, which varies in use cases and can be subjective [13]. Machine learning models can capture the

notion of similarity exhibited in the labeled examples [9, 12]. Unfortunately, labels are often scarce, impeding model training for time series pattern search [19]. Accordingly, researchers propose inviting users to verify patterns suggested by a tentative, rule-based (i.e., not involving model training) initial search [10, 22]. We call such user-in-the-loop pattern retrieval “feedback-driven pattern search”.

Existing methods for feedback-driven pattern search suffer from two weaknesses. **1) Fixed pattern size:** The time series is queried by a sliding window, whose size dictates that of the found patterns. To retrieve patterns of variable durations, multiple differently sized sliding window kernels are required, each scouring the data once. **2) Context-obliviousness:** The pattern relevance is appraised based on the features in the window without context. Sometimes, the target event lacks characteristic features itself but may be preceded or succeeded by distinctive signals that the user may not even notice a priori. In that case, considering the temporal proximity may contribute.

We present N00PS, a model-agnostic pipeline for feedback-driven pattern search. It features nonmyopic search, classifying whether each time point in the time series is located in a target pattern or not, rather than classifying patterns at each position of the time series as relevant or not. The “in-target-pattern” time points are then merged to found pattern intervals. We call this technique nonmyopic search because, as we will show, it recognizes patterns of different sizes and even peeks into their context to overcome the aforementioned weaknesses.

Please refer to the unabridged preprint of this work including all appendices under <https://dx.doi.org/10.2139/ssrn.5180352>.

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 [1]. Basic options include cosine similarity [5], Mahalanobis distance [16], and variants of the L^p norm, such as Euclidean Distance (ED) [15], and Chebyshev distance [7]. To capture temporally distorted patterns, elastic similarity measures are proposed, where Dynamic Time Warping (DTW) and its variants [18] are the most widely accepted and proven hard to beat for time series classification [2]. Unfortunately, “no single algorithm accounts for human judgments of time series similarity” [13]. Hence, many works employ a model in place of the rule-based similarity measure to assess the pattern relevance [9, 12]. Our 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 due to its speed, the primary concern of the initial search, while the feedback-driven search aims for a high retrieval accuracy.

Traversal Approaches. The similarity measure or the model discussed in the previous paragraph requires a traversal approach to search time series. The classic approach, named “sequential scanning” [6], checks every position in the time

series against a pattern in a sliding window, which cannot retrieve patterns of variable sizes efficiently. One solution to this problem is to match pieces in the query with pieces in the time series with the same properties (e.g., monotonicity), and a matching pair of pieces may differ in size [14]. Another possibility is to formulate the query as a regular expression that can search a symbolic representation of the time series, where regex quantifiers describe the scaling along the time axis [17, 21]. In addition, other works propose computational techniques [8]. However, these approaches only apply to certain rule-based similarity measures. For model-based pattern search, only sequential scanning is viable [10, 22]. Our traversal approach breaks this convention and embraces nonmyopic search, overcoming the two weaknesses and achieving higher retrieval accuracy and speed.

3 Method

3.1 Pipeline: An Overview

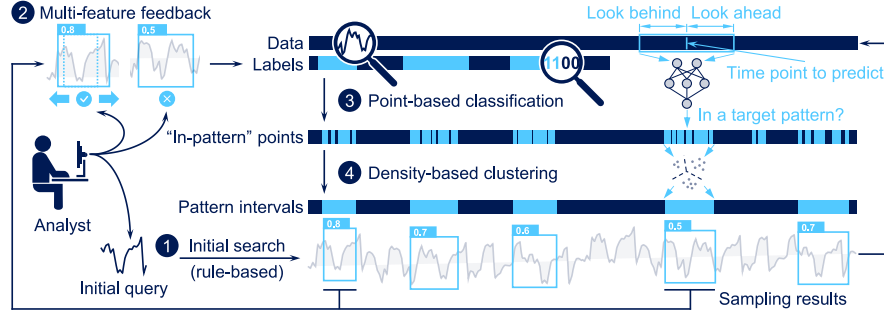


Fig. 1: **Processing Pipeline.** NOOPS combines time-point-based classification and density-based clustering to get variably sized patterns based on their internal dynamics and their context.

A time series $S = [s_i \in \mathbb{R}^d]_{0 \leq i < n}$ of length n and dimension d is an ordered sequence of n real-valued vectors in $\mathbb{R}^d$. We use the matrix notation with square brackets for indexed sequences and use curly brackets for sets of unordered, duplicate-free elements (e.g., all found patterns). We define: 1) a subsequence $S(a, b)$ of S , from index a inclusive to b exclusive, as $S(a, b) = [s_i]_{a \leq i < b}$; 2) a query of length w as $Q = [q_i \in \mathbb{R}^d]_{0 \leq i < w}$; and 3) a relevance function $f_{\text{rel}} : \mathbb{R}^{(b-a) \times d} \rightarrow [0, 1]$ that maps $S(a, b)$ to a relevance/confidence score.

We define the task time series pattern search as follows: Given a time series S and one or more target pattern examples, retrieve all subsequences $\{S(a, b) \mid f_{\text{rel}}(S(a, b)) \geq \tau \in (0, 1)\}$ in S , whose confidence is no less than a threshold τ .

We propose a four-stage pipeline for feedback-driven pattern search in time series, illustrated in Fig. 1. Stage ① detailed in Appendix C searches for the

query Q with a fast rule-based method, Mueen’s Algorithm for Similarity Search (MASS) [15] to obtain initial results. This stage is necessary if there is only one target pattern example provided. Stage ② elaborated in Appendix D collects user feedback, injecting domain knowledge and the user’s notion of relevance into the pipeline. Stage ③ and Stage ④ form the nonmyopic search, our major algorithmic contribution. Stage ③ trains a model with labels from Stage ②. The model input is $S(i - h, i + h + 1)$ around a time point s_i ; the model output is the probability of s_i located in a target pattern $\tilde{f}_{rel}(S(i - h, i + h + 1))$, where we use $\tilde{f}_{rel}: \mathbb{R}^{2h+1} \times d \rightarrow [0, 1]$ instead of f_{rel} to indicate the usage of the model for estimating the relevance of a time point, rather than the relevance of a pattern. Because users do not expect individual time point, but complete patterns, Stage ④ groups in-target-pattern time points into intervals $\{(a, b) \mid S(a, b) \in \{F\}\}$ using density-based clustering, where $\{F\}$ is the set of all found patterns.

3.2 Time-Point-Based Classification: The Adaptive Core

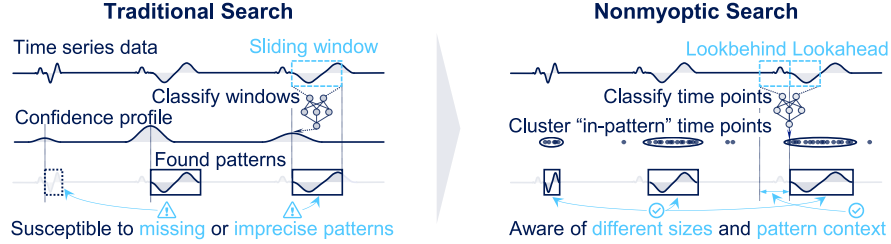


Fig. 2: **Traditional vs. Nonmyopic Search.** Traditional search applies a sliding window over the time series to get a confidence profile, where local maxima correspond to detected patterns, and the pattern length equals the window size. It is cumbersome for variable-length pattern retrieval and disregards pattern contexts. Nonmyopic search classifies each time point as inside or outside the target pattern and clusters the in-target-pattern points into pattern intervals. It recognizes variable-length patterns in a single pass and is context-aware.

In NOOPS’s pipeline, Stage ③ serves as the adaptive core. The traditional model-based pattern search classifies time series patterns directly. The input size of the model dictates the size of the found patterns. To obtain patterns of different sizes, the model has to go through multiple resampled versions of the time series.

We propose training the model to estimate if a time point is inside or outside a desired pattern (hereinafter called a positive or negative time point, respectively). The confidence of all time points is

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

where we pad the confidence with zeros (especially for $0 \leq i < h_1$, but also for $n - h_2 \leq i < n$) so that the time point indexes do not shift. 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. The indexes of the positive time point are

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

The shift from pattern-interval-based to time-point-based classification enjoys three advantages.

First, it retrieves variable-length patterns in a single scan. Though it still uses a sliding window of size $h_1 + h_2 + 1$, the window size no longer fixes the pattern length. Consequently, the data are only scanned once.

Second, it is context-aware. When classifying the starting time points of a pattern, the lookbehind covers part of the historical data, illustrated in Fig. 2 by the arrow from the symbol  above the text “context”. Likewise, the lookahead gradually probes into the future data beyond the pattern, as the window approaches the right end of the pattern. Thus, retrieving a pattern $F = S(a, b)$ considers an extended fragment $S(a - h_1, b + h_2)$, including the context $S(a - h_1, a) \cup S(b, b + h_2)$. This helps when there is characteristic information before or after the target event, as in the use case shown in Fig. S17.

Third, training becomes more robust. A positive user label $S(a, b)$ yields $b - a$ training instances $[S(i - h_1, i + h_2 + 1)]_{a \leq i < b}$. They view the pattern from different temporal perspectives, reducing variance in retrieval accuracy. In addition, the user label above implies negative labels $[S(i - h_1, i + h_2 + 1)]_{a - \tilde{h} \leq i < a} \cap [S(i - h_1, i + h_2 + 1)]_{b \leq i < b + \tilde{h}}$ nearby, preempting the provision of negative labels.

3.3 Density-Based Clustering: From Points to Intervals

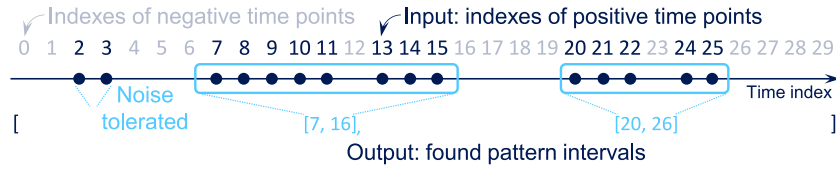


Fig. 3: **A Canonical Example of Time Point Clustering.** The (indexes of the) positive time points are converted to found pattern intervals, as the user expects to see found patterns in whole pieces instead of individual time points.

The definition of time series pattern search in Sec. 3.1 and the users expect patterns in whole pieces, not positive time points B from Stage 3. Therefore, Stage 4 converts B into found pattern intervals $\{(a, b) \mid S(a, b) \in \{F\}\}$.

Simply merging consecutive positive time points is unreliable due to inevitable misclassifications. False positive time points (e.g., Points 2 and 3 in

Fig. 3) would lead to spurious found patterns, while false negatives (Points 12 and 23) would fragment target patterns. A robust merging approach should tolerate such misclassifications.

We address this problem with one-dimensional density-based clustering. Because target patterns result in dense blobs of positive time points along the time axis, while gaps between target patterns contain sparse misclassified positive time points. This contrast in density makes density-based clustering suitable. Specifically, we use Density-Based Spatial Clustering of Applications with Noise (DBSCAN) (parameters in Appendix E.3) on the indices of positive time points B . The clusters correspond to the found patterns. The confidence of a found pattern $F = S(a, b)$ is the average confidence over its time points: $f_{rel}(F) = \frac{1}{b-a} \sum_{i=a}^{b-1} \tilde{c}_i$ where $\tilde{c}_i$ from Eq. 1 is the confidence of time point i .

4 Evaluation

We evaluated the proposed nonmyopic search in terms of retrieval accuracy and speed in a quantitative comparison with the traditional search.

Three labeled datasets of different characteristics are used. The **Deep Valve** dataset provides two use cases with different target patterns. Please refer to Appendix G for data plots, queries, ground truths, and the physical background of each dataset; Appendix E for Hardware, software, and parameter settings; and Appendix H for an introduction to the accuracy metric, Average Precision (AP).

Four models are included in the experiment: 1) **Random Forest** – top retrieval accuracy in traditional search [10]; 2) **XGBoost** – best accuracy–speed trade-off in our pilot study; 3) **LSTM** – widely used for time series analysis; and 4) **1D-CNN** – a common alternative to RNNs for time series [4]. Note that our goal is not to find the best model, but to benchmark the two pipelines.

Fig. 4 shows the results per use case. We omit **LSTM** because its much higher execution time distorts the scale, hindering comparison between other more efficient models. Full diagrams with **LSTM** are in Appendix I.2. In each diagram, the abscissa represents wall-clock time, and the ordinate retrieval accuracy (AP30), so better performance lies toward the top left. AP30 was chosen because it spreads scores well between 0 and 1. Light points denote model performance in nonmyopic search; dark linked ones that in traditional search. Since traditional search depends on the number of sliding window kernels for a size-invariant query, we report each model with multiple kernel counts, connected by lines. Each accuracy value is the mean of five runs, with vertical error bars showing standard deviation; each speed value is averaged over ten runs.

Nonmyopic search with the same model generally yields higher accuracy than traditional search. In fact, many traditional search scores are impractically low, indicating that traditional search fails with certain models in some use cases.

Nonmyopic search also enhances robustness: 1) Error bars around light points are barely visible, while those around dark points are prominent, indicating less consistency of traditional search outcomes across runs. 2) Light points for different models in the same plot lie at a comparable height, whereas dark lines

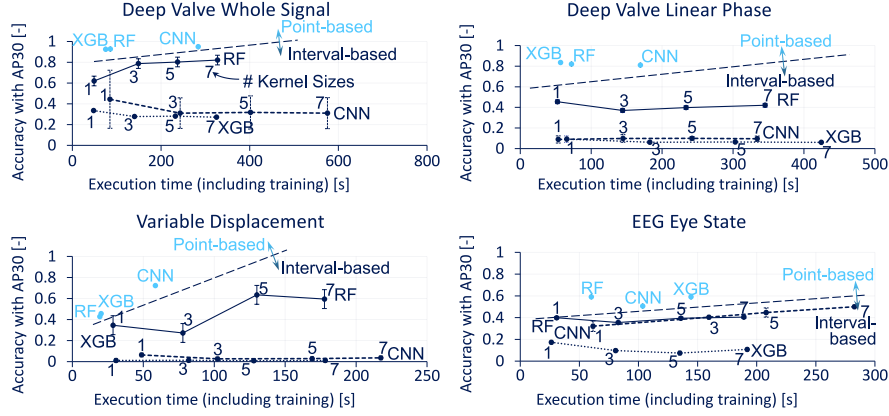


Fig. 4: **Nonmyopic vs. Traditional Search.** The light points (nonmyopic search) lie further up and left than the linked dark points (traditional search with varying numbers of sliding window kernels) in all plots, indicating better retrieval accuracy and speed. (RF: Random Forest, XGB: XGBoost)

are widely spread vertically, showing higher sensitivity of traditional search to model choice.

In the **Deep Valve Linear Phase** use case, all models in traditional search performed poorly because the target event (a primitive linear phase) lacks distinguishing features. Similar patterns appear in the background after normalization. However, this event is in a multi-phase process (Fig. S17 in Appendix G.2). Combined with the phases in proximity, the target becomes more distinctive. Therefore, the context-aware nonmyopic search maintains retrieval accuracy for all models in this use case as high as in the **Deep Valve Whole Signal** use case.

Not only is nonmyopic search more accurate and robust but also faster as it scans the data only once. In Fig. 4, the light points generally lie further left than the dark lines. In Appendix I.4, we decomposes the execution time into training and inference times, and benchmarked nonmyopic and traditional search, revealing more insights.

In summary, nonmyopic search improves retrieval accuracy and speed for feedback-driven pattern search. Further analysis is available in Appendix I.

5 Discussion and Conclusion

We present NOOPS, a model-agnostic pipeline for feedback-driven pattern search in multivariate time series, highlighting nonmyopic search. It combines time-point-based classification with density-based clustering, enabling variable-length pattern retrieval, context awareness, and more robust performance.

The following caveats and research directions are noteworthy: 1) Discriminate Overlapping Patterns. Users are warned that nonmyopic search would merge overlapping patterns into a single one, akin to semantic segmentation

in computer vision. 2) Diversify Feedback Candidates. We sample highly confident candidate patterns for feedback, which may lead to confirmation bias by converging on a narrow range of subclasses. Sampling strategy in active learning is a hot topic. It could potentially help to recommend moderately confident candidates that differ much from highly confident ones. 3) Add Channels in-between. The initial search may miss patterns without features that can be described by the rule-based similarity measure well. Therefore, it may help to focus on visually characteristic channels in the initial search and then add other relevant (according to domain knowledge) channels before the feedback-driven search.

Our tool is in production, helping automotive engineers trace events in diverse measurements. Next, we will primarily focus on improving the Visual Query System (VQS) design to better guide users through the analytical workflow.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Abanda, A., Mori, U., Lozano, J.A.: A Review on Distance Based Time Series Classification. *Data Mining and Knowledge Discovery* **33**(2), 378–412 (2019). <https://doi.org/10.1007/s10618-018-0596-4>
2. Bagnall, A., Lines, J., Bostrom, A., Large, J., Keogh, E.: 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). <https://doi.org/10.1007/s10618-016-0483-9>
3. Box, G.E.P., Jenkins, G.M., Reinsel, G.C., Ljung, G.M.: *Time Series Analysis: Forecasting and Control*. John Wiley & Sons (2015)
4. Chollet, F.: *Deep Learning mit Python und Keras: Das Praxis-Handbuch vom Entwickler der Keras-Bibliothek*. mitp, Frechen (2018)
5. Dong, Y., Sun, Z., Jia, H.: A Cosine Similarity-Based Negative Selection Algorithm for Time Series Novelty Detection. *Mechanical Systems and Signal Processing* **20**(6), 1461–1472 (2006). <https://doi.org/10.1016/j.ymssp.2004.12.006>
6. Faloutsos, C., Ranganathan, M., Manolopoulos, Y.: Fast Subsequence Matching in Time-Series Databases. In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 419–429. ACM, NY, USA (1994). <https://doi.org/10.1145/191839.191925>
7. Georgios, C., Dimitris, S., Kostas, P., Themis, P., Spiros, A., Spiros, S.: Efficient Range and kNN Twin Subsequence Search in Time Series. *IEEE Transactions on Knowledge and Data Engineering* (2022). <https://doi.org/10.1109/TKDE.2022.3167257>
8. Giao, B.C., Anh, D.T.: 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). <https://doi.org/10.1007/s40595-016-0062-4>
9. 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*, pp. 1–6 (2019). <https://doi.org/10.1109/CISP-BMEI48845.2019.8966048>

10. Lekschas, F., Peterson, B., Haehn, D., Ma, E., Gehlenborg, N., Pfister, H.: PEAX: Interactive Visual Pattern Search in Sequential Data Using Unsupervised Deep Representation Learning. *Computer Graphics Forum* **39**(3), 167–179 (2020). <https://doi.org/10.1111/cgf.13971>
11. Levchenko, O., Kolev, B., Yagoubi, D., Akbarinia, R., Masegla, F., Palpanas, T., Shasha, D., Valduriez, P.: BestNeighbor: Efficient Evaluation of kNN Queries on Large Time Series Databases. *Knowledge and Information Systems* **63**(2), 349–378 (2021). <https://doi.org/10.1007/s10115-020-01518-4>
12. Li, J., Xu, C., Zhang, T.: Similarity Measure of Time Series Based on Siamese and Sequential Neural Networks. In: *The 39th Chinese Control Conference*, pp. 6408–6413. IEEE (2020). <https://doi.org/10.23919/CCC50068.2020.9189261>
13. 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*, pp. 131–140 (2016). <https://doi.org/10.1109/VAST.2016.7883519>
14. Mannino, M., Abouzied, A.: Expressive Time Series Querying with Hand-Drawn Scale-Free Sketches. In: *Proceedings of the CHI Conference on Human Factors in Computing Systems*, pp. 1–13. ACM, NY, USA (2018). <https://doi.org/10.1145/3173574.3173962>
15. Michael, Y.C.-C., Yan, Z., Liudmila, U., Nurjahan, B., Yifei, D., Anh, D.H., Furtado, S.D., Abdullah, M., Eamonn, K.: 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*, pp. 1317–1322 (2016). <https://doi.org/10.1109/ICDM.2016.0179>
16. Prekopcsák, Z., Lemire, D.: Time Series Classification by Class-Specific Mahalanobis Distance Measures. *Advances in Data Analysis and Classification* **6**(3), 185–200 (2012). <https://doi.org/10.1007/s11634-012-0110-6>
17. Rodrigues, J., Folgado, D., Belo, D., Gamboa, H.: SSTs: A Syntactic Tool for Pattern Search on Time Series. *Information Processing & Management* **56**(1), 61–76 (2019). <https://doi.org/10.1016/j.ipm.2018.09.001>
18. Shokoohi-Yekta, M., Wang, J., Keogh, E.: On the Non-Trivial Generalization of Dynamic Time Warping to the Multi-Dimensional Case. In: *Proceedings of the SIAM International Conference on Data Mining*, pp. 289–297 (2015). <https://doi.org/10.1137/1.9781611974010.33>
19. Xi, L., Yun, Z., Liu, H., Wang, R., Huang, X., Fan, H.: Semi-supervised Time Series Classification Model with Self-supervised Learning. *Engineering Applications of Artificial Intelligence* **116**, 105331 (2022). <https://doi.org/10.1016/j.engappai.2022.105331>
20. Younan, M., Houssein, E.H., Elhoseny, M., Ali, A.E.-M.: Performance Analysis for Similarity Data Fusion Model for Enabling Time Series Indexing in Internet of Things Applications. *PeerJ. Computer science* **7**, e500 (2021). <https://doi.org/10.7717/peerj-cs.500>
21. Yu, Y., Becker, T., Trinh, L.M., Behrisch, M.: SAXRegex: Multivariate Time Series Pattern Search with Symbolic Representation, Regular Expression, and Query Expansion. *Computers & Graphics* **112**, 13–21 (2023). <https://doi.org/10.1016/j.cag.2023.03.002>
22. Yu, Y., Kruffy, D., Jiao, J., Becker, T., Behrisch, M.: PSEUDO: Interactive Pattern Search in Multivariate Time Series with Locality-Sensitive Hashing and Relevance Feedback. *IEEE Transactions on Visualization and Computer Graphics* **29**(1), 33–42 (2022). <https://doi.org/10.1109/TVCG.2022.3209431>

A List of Symbols

We use uppercase letters to denote vectors, matrices, or sets, and lowercase letters to denote scalars or functions. This rule applies to both Latin and Greek symbols. The only exception is s_i that denotes the value(s) at i -th time point in the time series S . If the time series S is multivariate and has d channels, s_i is a vector, namely, $s_i \in \mathbb{R}^d$. However, we cannot use the uppercase version S_i because it would collide with S_j which denotes the j -th channel in the time series S . Unlisted symbols may vary in meaning according to the context. Nonetheless, unlisted symbols are rare cases. Note that the listed symbols also include the ones used in the appendices, especially App. C, App. D, and App. E.

Table S1: Latin Symbols

Symbol	Meaning
$A_{filtered}$	Starting time point indexes of potential found patterns after filtering A_{nms} with a confidence threshold τ .
A_{nms}	Starting time point indexes of potential found patterns after non-maximum suppression in A_{raw} .
A_{raw}	Local maximum points in merged confidence profile C_{merged} , which are starting time point indexes of potential found patterns.
B	Indexes of the time points classified as in a target pattern, also called positive time points in this work.
$C_{interim}$	Confidence of patterns starting at every time point with every resampled version of the query Q after averaging C_{raw} over channels during the rule-based initial search.
C_{merged}	Confidence of patterns starting at every time point after selecting the best resampled version of the query Q with the highest confidence in $C_{interim}$ during the rule-based initial search.
C_{raw}	Confidence of patterns starting at every time point in every channel with every resampled version of the query Q directly after similarity profile calculation during the rule-based initial search.
$\tilde{C}$	Confidence of every time point in a time series, whether it is located in a target pattern.
F	A found pattern.
$\{F\}$	All found patterns in a time series.
FN	The number of false negatives.
FP	The number of false positives.
G	A ground truth.
$\{G\}$	All ground truths in the time series. Please note that a time series may have target patterns $\{T\}$, regardless of whether ground truths $\{G\}$ are provided or not.
$\mathbb{N}_0$	The set of non-negative integers.
⋮	

Table S1: Latin Symbols – Continued

Symbol	Meaning
$\mathbb{N}_{>0}$	The set of positive integers.
P	A pattern.
$\mathbb{R}$	The set of real numbers.
S	A time series.
S_j	The j -th channel in the time series S .
$S(a, b)$	Subsequence in S that starts with time point index a inclusive and ends with time point index b exclusive.
T	A target pattern.
$\{T\}$	All target patterns in a time series.
TP	The number of true positives.
Q	Query to search for.
Q_j	The j -th channel in the query Q .
Q_{jk}	The j -th channel in the k -th resampled version of Q .
W	Lengths of every resampled versions of Q .
$\widetilde{W}$	Lengths of the resampled version of Q chosen for every time point.
Y	An array of the same length as the time series containing boolean values of whether each time point in the time series is in a found pattern (1 if time point in a target pattern, else 0).
Z	An array of the same length as the time series containing boolean values of whether each time point in the time series is in a ground truth (1 if time point in a ground truth, else 0).
a	Index of the starting time point of a subsequence.
b	Index of the end time point of a subsequence.
c_i	Confidence of the pattern starting at the time point index i .
c_{ijk}	Confidence of the univariate pattern starting at time point index i in j -th channel, calculated with j -th channel in k -th resampled version of the query Q .
c_{ik}	Confidence of the pattern starting at time point index i , calculated with k -th resampled version of the query Q after averaging over j for c_{ijk} .
$\widetilde{c}_i$	Confidence that the time point with index i is located in a target pattern.
d	Dimension (number of channels) of the time series.
f_{rel}	A function that takes a subsequence $S(a, b)$ of S as the input and returns the relevance/confidence that $S(a, b)$ is a target pattern.
$\widetilde{f}_{rel}$	A function that takes a subsequence $S(i - h_1, i + h_2 + 1)$ around the time point s_i and returns the confidence that the time point s_i is in a target pattern.
$\vdots$	

Table S1: Latin Symbols – Continued

Symbol	Meaning
f_{sim}	Similarity measure that takes two time series patterns as input and returns their similarity normalized to $[0, 1]$.
h	Lookaround. To classify a time point s_i , its proximity $S(i - h, i + h + 1)$ is used as input data.
h_1	Lookbehind, number of historical time points to consider when classifying whether a time point as in a target pattern or not. In practice, we set $h_1 = h$.
h_2	Lookahead, number of future time points to consider when classifying whether a time point as in a target pattern or not. In practice, we set $h_2 = h$.
$\tilde{h}$	Length of the neighborhood around a positively labeled pattern $S(a, b)$, within which the time points are considered negative, i.e., all time points in $S(a - \tilde{h}, a) \cap S(b, b + \tilde{h})$ are considered negative.
i	Index for time points.
j	Index for channels.
k	Index for resampled versions of Q .
m	Number of resampled versions of Q , we set $m = 8$.
n	Length (number of time points) of the time series.
s_i	The time point at index i in time series S .
w	Window length (number of time points in the window).
w_k	Length of the k -th resampled version of Q .
$\tilde{w}_i$	Length of the chosen resampled version of Q for time point with index i .
y_i	Whether the time point with index i is in a found pattern. If yes, $y_i = 1$, otherwise $y_i = 0$.
z_i	Whether the time point with index i is in a ground truth. If yes, $z_i = 1$, otherwise $z_i = 0$.

Table S2: Greek Symbols

Symbol	Meaning
δ	Radius used in non-maximum suppression in the rule-based initial search.
τ	Confidence threshold, which the confidence of a found pattern should reach.
v	Intersection over Union (IoU) threshold for the metric AP.
$\vdots$	

Table S2: Greek Symbols – continued

Symbol	Meaning
v^*	IoU threshold for the BR-Agent.

B Recent Visual Query Systems

Our proposed algorithm and the user interface constitute a VQS. According to Lee et al [8], the term VQS is defined by Ryall et al [14] as “systems that allow users to specify and search for desired line chart patterns via visual interfaces”. Literature categorizes VQSs primarily according to the approach to specifying the query. There are two prevailing approaches, viz., query-by-example and query-by-sketch. The former defines the query by providing an example, e.g., by marking an interval on a line chart [9, 19, 18], while the latter by sketching a prototype, e.g., on a canvas [10, 8, 2]. Other query definition approaches include formulas, regular expressions [16], and natural language [16, 6]. Tab. S3 compares between recent VQSs including NOOPS. We settled on query-by-example because it is overly demanding to ask our users to draw or describe in other ways complex temporal dynamics, let alone in multiple channels. Conversely, our engineers can usually provide at least one example of the target event. On the debit side, query-by-example can be brittle [6]. The user interface is not our focus. It mainly serves to assist the user in applying the proposed algorithm, e.g., for acquiring multi-feature user feedback.

Table S3: Recent VQSS

[illegible]

Table S3: Recent VQSs – Continued

VQS ¹	Year ²	Query Definition	Representation Indexing	Similarity Measure / Classifier	MV ³	FB ⁴	VD ⁵	CA ⁶	Focus	Contributions
PSEUDO [19]	2022	Example	LSH	ED / DTW	Yes	Yes	No	No	Speed for high-dimensional time series retrieval, feedback-driven pattern search	
SAXRegEx [18]	2023	Example	SAX	Regex	Yes	No	Yes	No	(Multivariate) pattern distortions, speed	
NOOPS (this work)	2024 ¹⁰	Example	None	Unlimited (best model in experiments: random forest)	Yes	Yes	Yes	Yes	Feedback-driven search	

Table Footnotes

¹ VQSs sorted in ascending order of the first publication year. VQSs published in the same year are sorted alphabetically in ascending order.

² If there are multiple publications that present different versions of the VQS, only the first publication is recorded in this table. If the paper was first published in a conference and then indexed in a special issue of a journal, the journal version is preferred.

³ **MV**: supporting multivariate time series retrieval.

⁴ **FB**: supporting feedback-driven search.

⁵ **VD**: supporting retrieving patterns of variable durations.

⁶ **CA**: supporting context-awareness that allows taking into consideration the temporal proximity of patterns during retrieval.

⁷ TimeSearcher is obviously not recent. However, it is the pioneer of VQS design on visualizations and interactions dedicated to time series pattern search. Thus, it is mentioned in this table.

⁸ The user interface visualizes multivariate time series up to eight channels. Pattern search is conducted in each channel independently. Found patterns are also independently highlighted in each channel.

⁹ MASS is not a VQS because it does not have a visual interface. It is also not proposed recently. However, it is an important benchmark algorithm. Therefore, it is listed in the table.

¹⁰ **NOOPS** has not been published yet. The publication year is estimated.

C Initial Search: Proposing Label Candidates

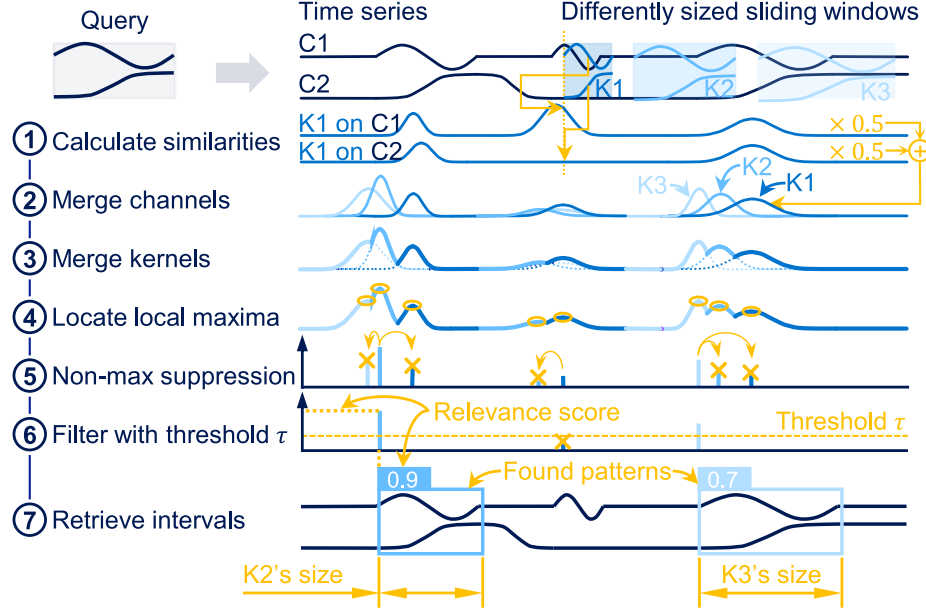


Fig. S1: **Rule-Based Initial Search.** It scans the data to locate prospective temporal positions of relevant patterns. At each position, it tries out different pattern sizes.

NOOPS’s pipeline needs to begin with a rule-based initial search (Stage ① in Fig. 1) because the user often can only provide one target pattern example at the beginning. That said, if multiple target pattern examples are available, the user can go ahead with the nonmyopic search (Stage ③ and Stage ④). Conversely, if the results from the rule-based initial search (Stage ①) have already satisfied the user’s expectation, the user may skip the feedback (Stage ②) and the feedback-driven search (Stage ③ and Stage ④).

Our work aims to contribute to the feedback-driven pattern search. Stage ① is rule-based and does not contain academic contributions. Nonetheless, we explain it in detail for two reasons. First, this technique is not self-explanatory, and nowhere else have we found a complete and detailed description of how it works. Second, the rule-based initial search pipeline is similar to the traditional model-based feedback-driven pattern search pipeline after model training, which is our baseline to benchmark the proposed nonmyopic search.

It is worth noting that f_{rel} works differently in rule-based and model-based pattern search. In the former, it is common to begin with one target pattern example, i.e., the query Q . f_{rel} is a partial function of a similarity measure f_{sim}

that takes two patterns as input and returns their similarity normalized to $[0, 1]$, where one of the patterns is Q , i.e., $f_{rel}(P) = f_{sim}(P, Q)$. The user usually does not know the optimal τ (equivalently, the desired number of top hits / nearest neighbors) a priori and needs to adjust τ after inspecting the search results. In contrast, model-based pattern search is driven by multiple target pattern examples (ground truths or labels from user feedback), and f_{rel} takes the form of a model, Manual optimization of τ is unnecessary.

Fig. S1 illustrates the pipeline of the rule-based initial search. Given a d -dimensional time series S of length n and a query Q of length w , it undergoes seven steps to arrive at the found pattern intervals and their confidence $\{((a, b), f_{rel}(S(a, b))) \mid S(a, b) \in \{F\}\}$.

Step ① calculates the similarity between Q and every pattern $[S(i, i + w)]_{0 \leq i \leq n-w}$ in S . We adopt MASS to carry out the task. It computes the similarity profile highly efficiently, the primary concern for the initial search. Like many other methods, MASS does not support multivariate time series off-the-shelf. So, we apply it to each channel and merge the results over all channels in Step ②. In practice, we use m resampled (temporally scaled) versions of Q of lengths $W = [w_k]_{0 \leq k < m}$ to find patterns of variable sizes. Please refer to App. E.3 Appendix E.3. for details of the parameter settings, e.g., for m and W . MASS calculates distance rather than similarity. We inverse the distance and normalize it to range $[0, 1]$ to get raw confidence for each time point i in each channel j with each resampled version k of the query

$$C_{raw} = \left[c_{ijk} = \begin{cases} f_{sim}(S_j(i, i + w_k), Q_{jk}) & \text{if } 0 \leq i \leq n - w_k \\ 0, & \text{otherwise} \end{cases} \right]_{\substack{0 \leq i < n \\ 0 \leq j < d \\ 0 \leq k < m}} \quad (S3)$$

where S_j is the j -th channel in S , and Q_{jk} is the j -th channel in k -th resampled version of Q . Note that C_{raw} is padded with 0s, so that $\forall i \in (n - w_k, n)$, $c_{ijk} = 0$. The padding ensures that all m resampled versions of Q have the same size n along the time axis indexed by i , simplifying subsequent calculations.

Step ② merges C_{raw} by averaging over channels to get the interim confidence for each time point i with each resampled version of Q

$$C_{interim} = \left[c_{ik} = \frac{1}{d} \sum_{j=0}^{d-1} c_{ijk} \right]_{\substack{0 \leq i < n \\ 0 \leq k < m}} \quad (S4)$$

Step ③ merges $C_{interim}$ by choosing the highest confidence over k resampled version of Q for each time point i to get the merged confidence

$$C_{merged} = \left[c_i = \max_{0 \leq k < m} (c_{ik}) \right]_{0 \leq i < n} \quad (S5)$$

During Step ③, we retain the length of the chosen resampled version of Q for each time point index i

$$\widetilde{W} = \left[\widetilde{w}_i = w_{\arg\max_{0 \leq k < m} (c_{ik})} \right]_{0 \leq i < n} \quad (S6)$$

in order to retrieve the found pattern lengths later in Step ⑦.

Step ④ retrieves all strict local maximum points in C_{merged} to get potential start positions of the found patterns

$$A_{raw} = \{i \mid c_{i-1} < c_i, c_i > c_{i+1}\} \quad (S7)$$

To avoid discovering the same pattern as several separate overlapping ones, Step ⑤ conducts non-maximum suppression

$$A_{nms} = \left\{ i \in A_{raw} \mid c_i = \max_{i-\delta \leq t \leq i+\delta} c_t \right\} \quad (S8)$$

where δ is the radius within which the highest local maximum suppresses the lower local maxima in proximity.

Step ⑥ filters A_{nms} with the confidence threshold τ

$$A_{filtered} = \{i \in A_{nms} \mid c_i \geq \tau\} \quad (S9)$$

The user usually cannot set an optimal τ a priori, but needs to adjust it after inspecting the results. Then, Step ⑥ and Step ⑦ are repeated.

Finally, Step ⑦ retrieves the found pattern intervals and their confidence

$$\{((i, i + \tilde{w}_i), c_i) \mid i \in A_{filtered}\} \quad (S10)$$

D Multi-Feature Feedback: Procuring Training Labels

In practice, model-based pattern search is impeded by the label scarcity. Researchers proposed relevance feedback / active learning [12], whereby users are invited to label the found patterns, possibly in multiple iterations. Since its introduction in time series analysis [7], this technique has also gained attention in time series pattern search [9, 15, 19]. However, the state-of-the-art methods incorporating relevance feedback allow the user to accept, reject, or eschew a found pattern, from which a positive, a negative, or no label for model (re)training is extracted. None of the options is satisfactory when a found pattern overlaps a target one partially, as illustrated in Fig. S2.



Fig. S2: **Problem with Traditional Relevance Feedback.** User feedback is confined to acceptance, rejection, or abstention. all of which are controversial given a partially correct found pattern.

We propose to solicit feedback on the pattern positions and sizes besides correctness/relevance. We name this approach **multi-feature feedback** and will show that we cannot afford to omit this minor additional labeling, as it substantially raises retrieval accuracy even further.

Technically, multi-feature feedback happens in Stage ② in Fig. 1, closing the proposed pipeline. It supplies labels for model training while infusing the user’s domain knowledge into the pipeline.

Conventional active learning only collects feedback on the correctness of the found patterns. For each sampled pattern, the user can accept, reject, or skip it. Accepted found patterns become positive labels, and rejected ones negative labels. Skipped ones are ignored. This poses the dilemma in Fig. S2: if accepted, the pattern imperfection may mislead the model; if rejected, its partial correctness may confuse the model; if skipped, the partial correctness is not exploited.

In nonmyopic setting, label imperfection propagates to multiple incorrectly labeled training instances. Recall that an accepted found pattern $F = S(a, b)$ yields $(b - a)$ positive training instances $[S(i - h_1, i + h_2 + 1)]_{a \leq i < b}$. Suppose the user actually wants $T = S(c, d)$ with $a < c < b < d$ that overlaps F , i.e., F starts and ends too early. The feedback would correctly label time points in $S(c, b)$ (instances $[S(i - h_1, i + h_2 + 1)]_{c \leq i < b}$) as positive, incorrectly labels all time points in $S(a, c)$ (instances $[S(i - h_1, i + h_2 + 1)]_{a \leq i < c}$) as positive.

We address this problem by introducing multi-feature feedback on the position and size of found patterns in addition to their relevance. In the previous example, when the system proposes F , the user should have noticed T . Otherwise, the user would not have accepted F . The user can then adjust F so that

$a_{\text{new}} := c$, $b_{\text{new}} := d$, i.e., $F_{\text{new}} := T$. This resolves the issues from imprecise binary relevance feedback and “salvages” found patterns that slightly overlap a target pattern and would otherwise be rejected.

E Implementation Details

E.1 Hardware and Operating System

We conducted all experiments locally on the same laptop HP EliteBook 850 G5 with

- Processor: Intel® Core™ i7-8650U CPU @ 1.90GHz 2.11GHz
- Installed RAM: 16.0 GB (15.8 GB usable)
- Operating System: 64-Bit Windows 10 Enterprise, Version 22H2
- Storage: 1TB SAMSUNG MZVLB1T0HBLR-000H1

We disabled all parallel computing and GPU acceleration, especially for neural networks and XGBoost.

E.2 Essential Libraries

Please refer to Tab. S4.

E.3 Parameter Settings

- Rule-based initial search
 - Number of kernels m : $m = 8$. The lengths of these kernels $W = [w_k]_{0 \leq k < m}$ scale logarithmically between a lower bound w_0 and an upper bound w_{m-1} (see the next point). The kernel sizes are

$$W = \left[w_i = \left\lfloor w_0 \times \left(\frac{w_{m-1}}{w_0} \right)^{\frac{k}{m-1}} \cdot w \right\rfloor \right]_{0 \leq k < m}$$

where the operator $\lfloor \cdot \rfloor$ denotes the floor function

- Lower and upper bound of the lengths of the resampled query w_0 and w_{m-1} : they are determined according to how much the durations of the target patterns in the time series scale. For instance, $w_0 = 0.5$ means that the lengths of the found patterns are half as long as the query; and w_{m-1} means that the lengths of the found patterns are at most twice as long as the query. Specifically for each use case,
 - * **Deep Valve Whole Signal**: $w_0 = 0.5$, $w_7 = 3.0$;
 - * **Deep Valve Linear Phase**: $w_0 = 0.5$, $w_7 = 3.0$;
 - * **Variable Displacement**: $w_0 = 0.5$, $w_7 = 2.0$; and
 - * **EEG Eye State**: $w_0 = 0.5$, $w_7 = 12.0$ (the durations of the target patterns vary substantially in this use case).
- Non-maximum suppression radius δ : the same as the query length w .
- Time-point-based classification
 - The concrete implementation of $\tilde{f}_{rel}$ differs for each model. For instance, some models, like CNN and LSTM, support multivariate time series out of the box and can take $S(i - h_1, i + h_2 + 1)$ as their input directly. Other models, like Random Forest and XGBoost, require flattening $S(i - h_1, i + h_2 + 1)$ beforehand.

Library	Usage	Link
Libraries for Algorithm		
MASS	Calculating similarity profiles in the rule-based initial search	https://github.com/matrix-profile-foundation/mass-ts
scikit-learn	Machine learning model random forest and clustering algorithm DBSCAN in the feedback-driven search, PCA and t-SNE in the found pattern candidate distribution scatter plot for finding false negatives in the UI	https://scikit-learn.org/stable/
TensorFlow	Machine learning models LSTM and CNN in the feedback-driven search	https://www.tensorflow.org/
Keras		
UMAP	Dimensionality reduction technique UMAP in the found pattern candidate distribution scatter plot for finding false negatives in the UI	https://umap-learn.readthedocs.io/en/latest/
XGBoost	Machine learning model XGBoost in the feedback-driven search	https://xgboost.readthedocs.io/en/stable/
Libraries for Web Development		
BokehJS	General charts	https://docs.bokeh.org/en/latest/docs/dev_guide/bokehjs.html
D3	Customize forces for the library React Force Graph	https://d3js.org/
Django	Backend API	https://www.djangoproject.com/
MUI	UI design	https://mui.com/
React	Frontend library	https://react.dev/
React Force Graph	Force-directed graph for identifying the false positives in the <i>Analysis View</i>	https://github.com/vasturiano/react-force-graph

Table S4: Essential Libraries Sorted in Alphabetical Order in Each Category

- lookaround h : average length of all positive labels.
 - lookbehind h_1 : $h_1 = h$.
 - lookahead h_2 : $h_2 = h$.
 - negative context around a positive label $\tilde{h}$: $\tilde{h} = h$.
 - Models
 - * Random forest: default parameters from `sklearn.ensemble.RandomForestClassifier`.
 - * XGBoost: `tree_method = "hist"`, otherwise default parameters from `xgboost.XGBClassifier`.
 - * LSTM: first layer `tf.keras.layers.LSTM` with 20 units, return sequence; second layer `tf.keras.layers.LSTM` with 20 units; third layer `tf.keras.layers.Dense` with one unit and activation `tf.keras.activations.sigmoid`; other parameters use defaults.
 - * CNN: first two layers `tf.keras.layers.Conv1D` with 20 filters of size 5 and “same” padding, activation `tf.keras.activations.relu`, kernel initializer `tf.keras.initializers.LecunNormal`, followed by a flatten layer `tf.keras.layers.Flatten` and a dense layer `tf.keras.layers.Dense` with one unit and activation `tf.keras.activations.sigmoid`; other parameters use defaults.
 - Early stopping: 10 epochs, except for random forest, which does not support early stopping.
- Density-based clustering (parameters of DBSCAN)
- *eps*: As the `scikit-learn` documentation explains, it is “the maximum distance between two samples for one to be considered as in the neighborhood of the other.” This is a standard DBSCAN parameter. Conceptually, the higher the value, the more likely for the algorithm to retrieve long found patterns. We estimate that this parameter should be comparable and smaller to the average length of the positive labels in feedback-driven search. Empirically, we set it $\frac{2}{3} \cdot h$. On a side note, density-based clustering distinguishes itself from partition-based and hierarchical clustering by not necessitating the provision of the often missing prior knowledge of the number of clusters or the number of levels. Nevertheless, this property is not necessarily an advantage of density-based clustering due to other parameters to set, which indirectly relate to the number of clusters / found patterns, like *eps* and the subsequent *min_samples*.
 - *min_samples*: As the `scikit-learn` documentation explains, it is “the number of samples (or total weight) in a neighborhood for a point to be considered as a core point. This includes the point itself.” This is also a DBSCAN standard parameter. Conceptually, the higher the value, the denser the positive time points in a found patterns need to be. Theoretically, this parameter is conditioned by the similarity between the target patterns in the time series data. Empirically, we set it $\lfloor 2 \times eps \times min_proportion \rfloor$, and *min_proportion* = 0.1.
 - Other parameters use defaults from `sklearn.cluster.DBSCAN`.

F User Interface

We designed a user interface to support query definition, result inspection & analysis, and feedback provision. While indispensable for interactive, feedback-driven search, the UI itself is not our focus in this work. Consequently, we keep the UI design and its evaluation (a case study and an expert study with two experts) in appendices.

This appendix shows an annotated single screenshot, which is vertically split into multiple figures to fit the page size of this paper. It shows the final status with three measurement data files from the **Variable Displacement** use case shown in App. G.3. It undergoes one feedback loop and achieves a nearly perfect result in the three given measurement data files. This appendix serves to introduce the features of the user interface. Please check the video <https://youtu.be/7-MPiC7asBg> for a 20-minute introduction to the UI. Please refer to App. K for a cohesive workflow and step-by-step screenshots of the same use case.

We try to concentrate the annotations in as few figures as possible. However, due to space limit for annotations, we may have to place the annotations in multiple figures and in results for different measurement files. On the other hand, because the visualizations in different files are similar in terms of their functions, we may for example explain the functions of visualizations for result inspection in the first file in the figure caption and skip the explanation in the figure caption for the same visualization for result inspection in the second file.

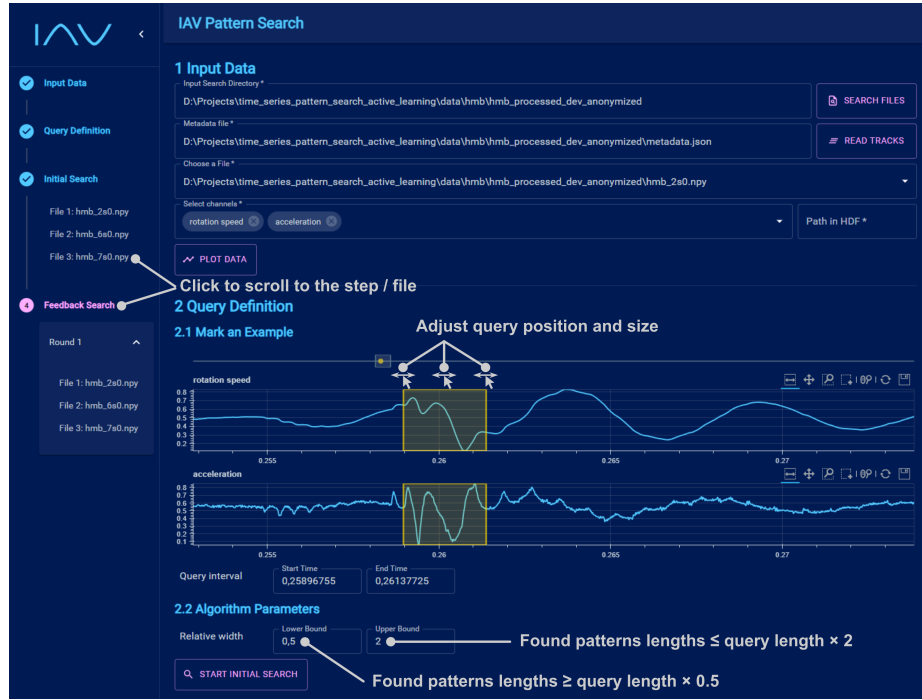


Fig.S3: Sidebar, Inputs, and Query Definition. NOOPS is designed with a streamlined workflow shown in the left sidebar. The main area shows all information on the same (potentially very long) page to provide a streamlined experience. The content in the main area is organized in sections and subsections like an essay for easy reference. Let us call a section (e.g. “1 Input Data”) a step and a subsection (e.g. “2.1 Mark an Example”) a sub-step. Step 1 Input Data contains input fields for the measurement data, metadata (mainly a list of channel names in the data), the measurement data file in which to define the query, the relevant channels, and the data path in HDF (only for HDF files). Step 2 Query Definition plots the channels in the selected data file in Step 1, where the query is defined by drawing a box in any channel. A box with the same time interval will appear in other channels. The user can adjust the position and size of the box denoting the query after marking it in any channel. The adjustment will be broadcast in other channels. The algorithm parameters for the rule-based initial search are also configured in this step. There are only a pair of parameters, namely, the lower and upper bounds for the duration scaling factor compared with the query. A short explanation of the bounds can be found in App. E.3. A canonical example: if the lower bound is 0.5 and the upper bound is 2.0, the found patterns are at least half and at most twice as long as the query. There is a small inconsistency. The *Overview* is called *Data Overview* in subsequently the feedback-driven search. Please forgive us for this mistake. We will only use the term *Overview* in this work.

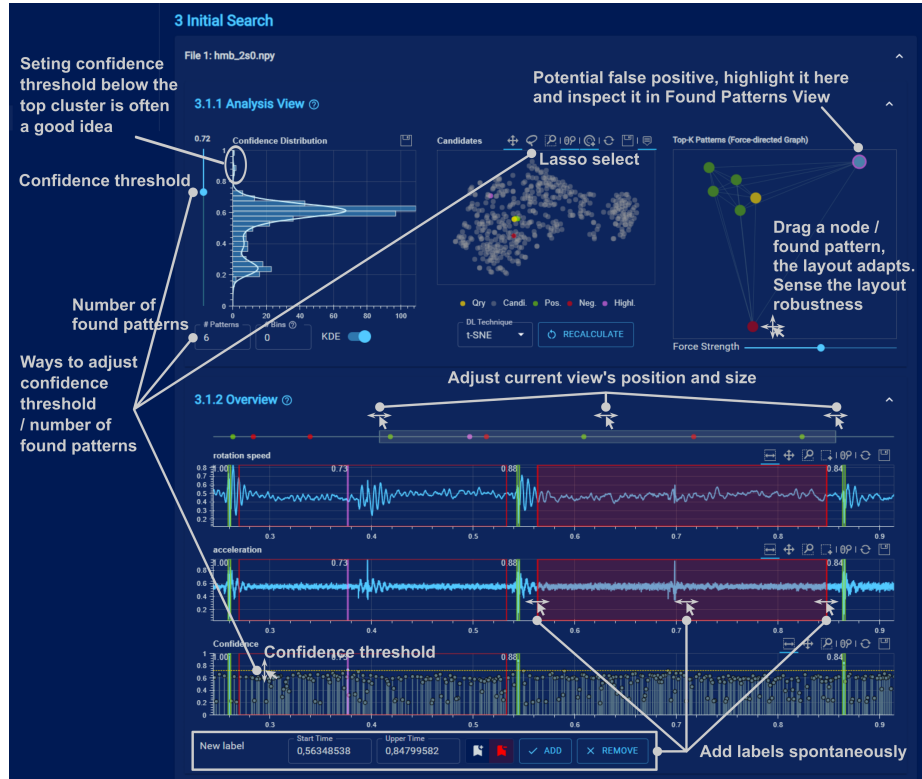


Fig. S4: **Initial Search - File 1 - Analysis View and Overview.** Step 3 shows the results of the rule-based initial search. The result of each data file is shown separately and collapsed by default. For each file, there are four views, showing the result from different perspectives. This figure shows two views. Namely, the *Analysis View* and the *Overview*. The *Analysis View* is collapsed by default because some engineers would like to jump directly to the found patterns and would refrain from much analysis. This view supports tuning the confidence threshold / number of found patterns and discovering false negatives / positives. The *Overview* plots the whole measurement data and all events for exploration. It is also the place to add labels spontaneously. It is recommended to add one or two long negative labels to help the model recognize the background, rather than picking random negative labels that are not necessarily representative. This is only possible with and thanks to time-point-based classification in nonmyopic search.

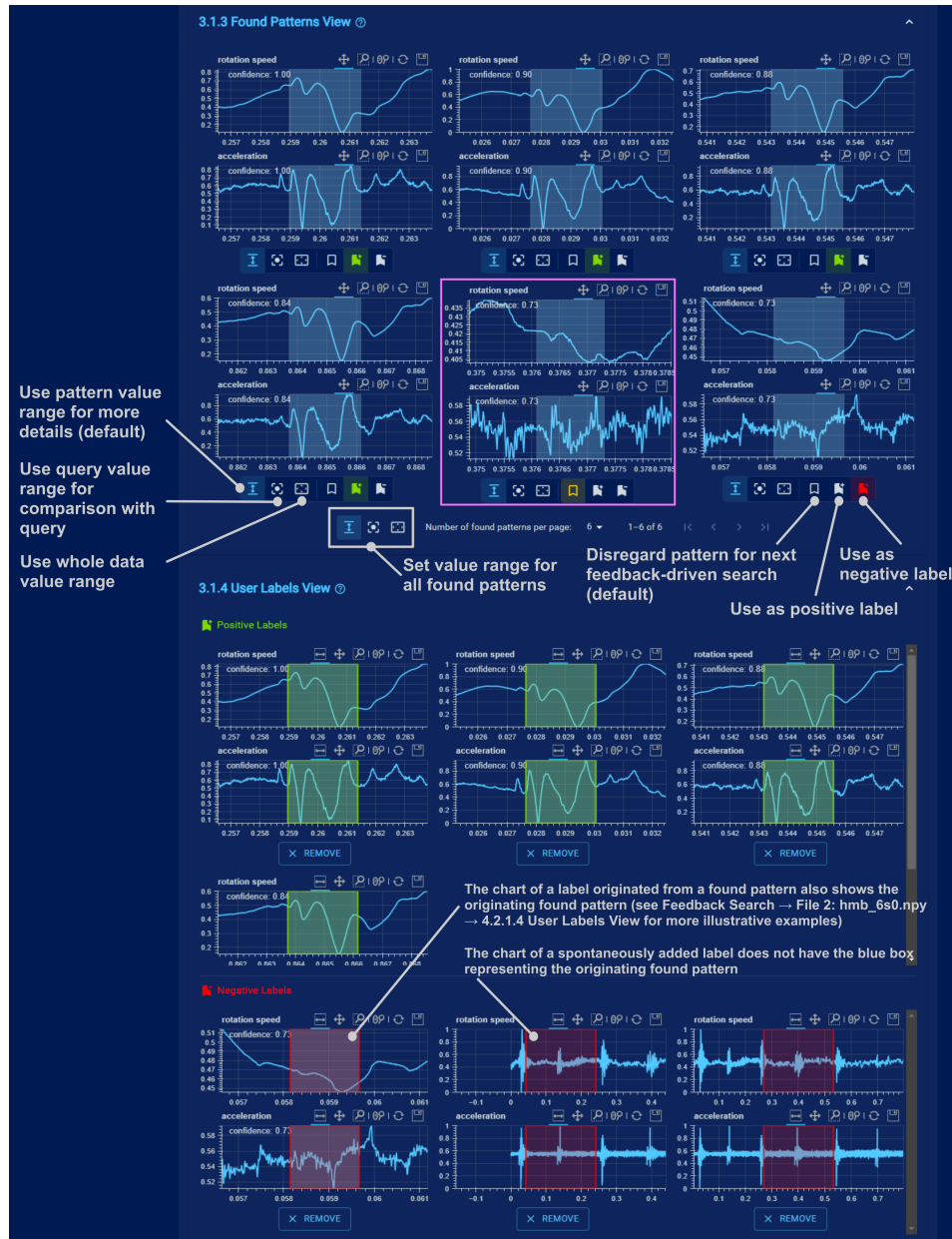


Fig. S5: Initial Search - File 1 - *Found Patterns View* and *User Labels View*. The *Found Patterns View* lists the found patterns in descending order of their confidence for inspection and comparison. The *User Labels View* lists the positive and negative labels from the user. They will be used to train the classification model when the user starts a new feedback-driven search.

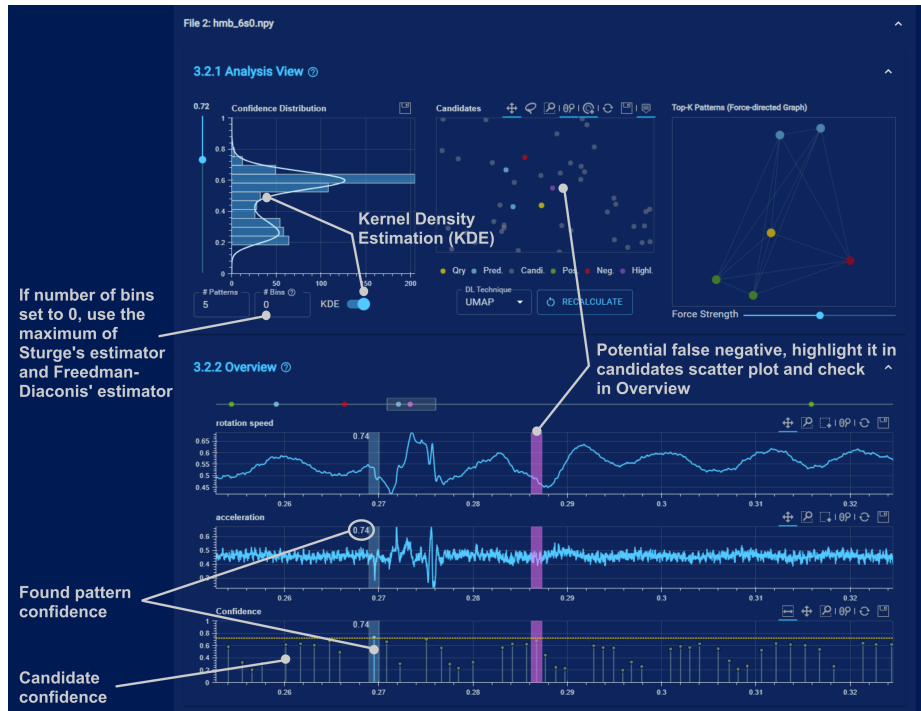


Fig. S6: Initial Search - File 2 - *Analysis View* and *Overview*.

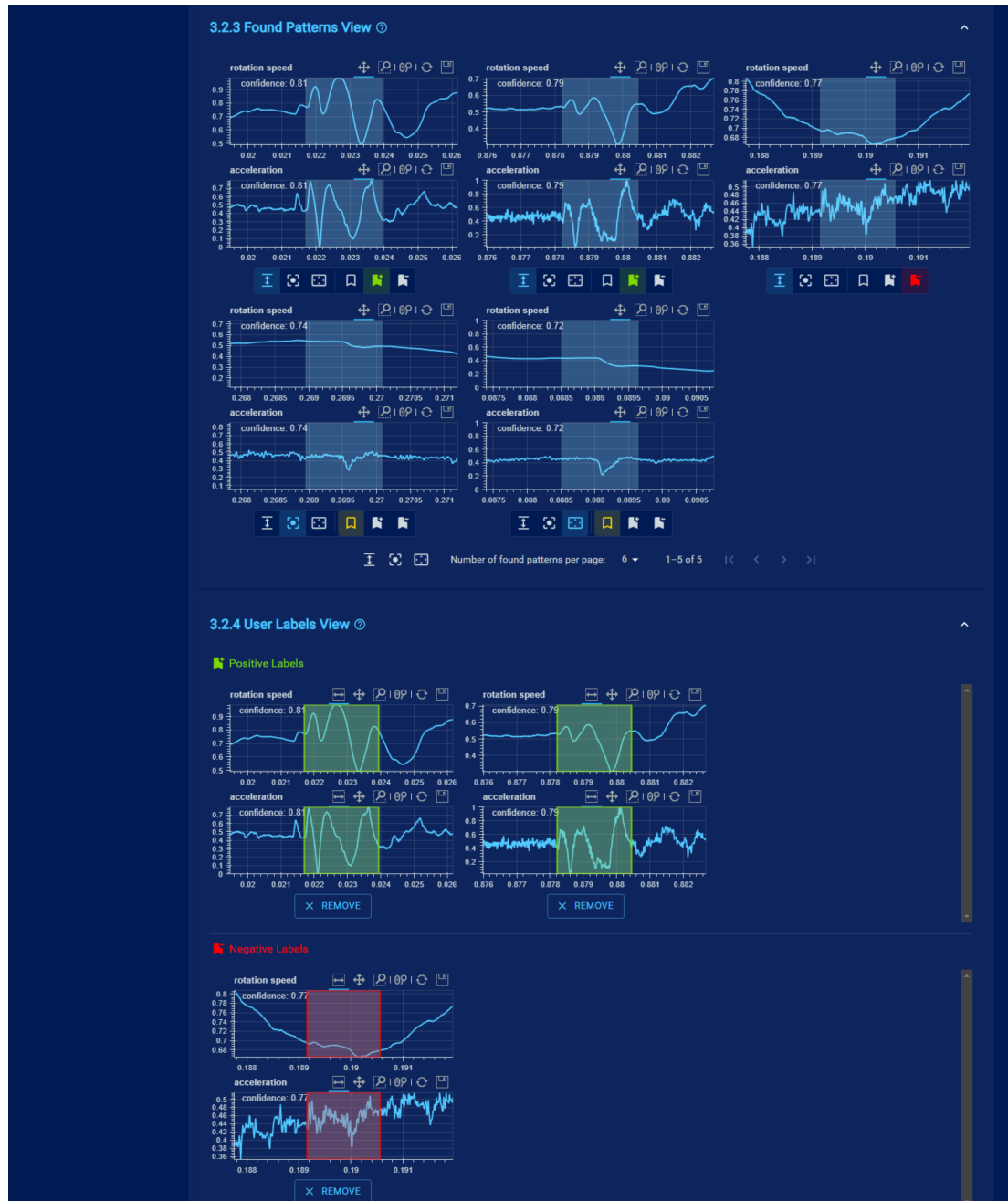


Fig. S7: Initial Search - File 2 - *Found Patterns View* and *User Labels View*.

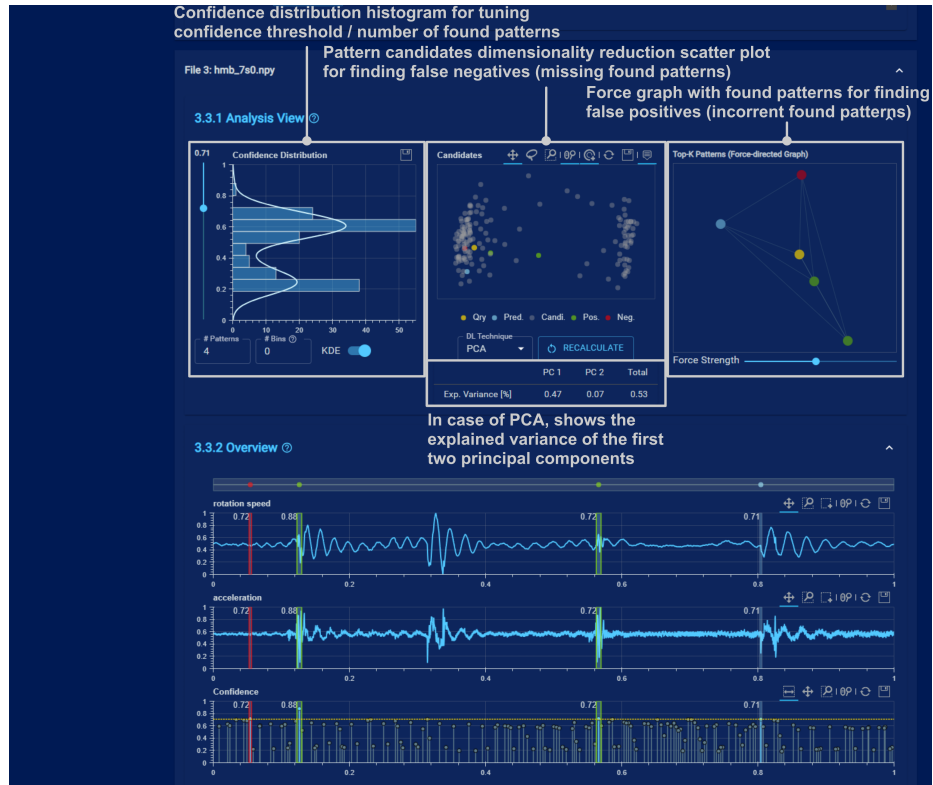


Fig. S8: Initial Search - File 3 - *Analysis View* and *Overview*.

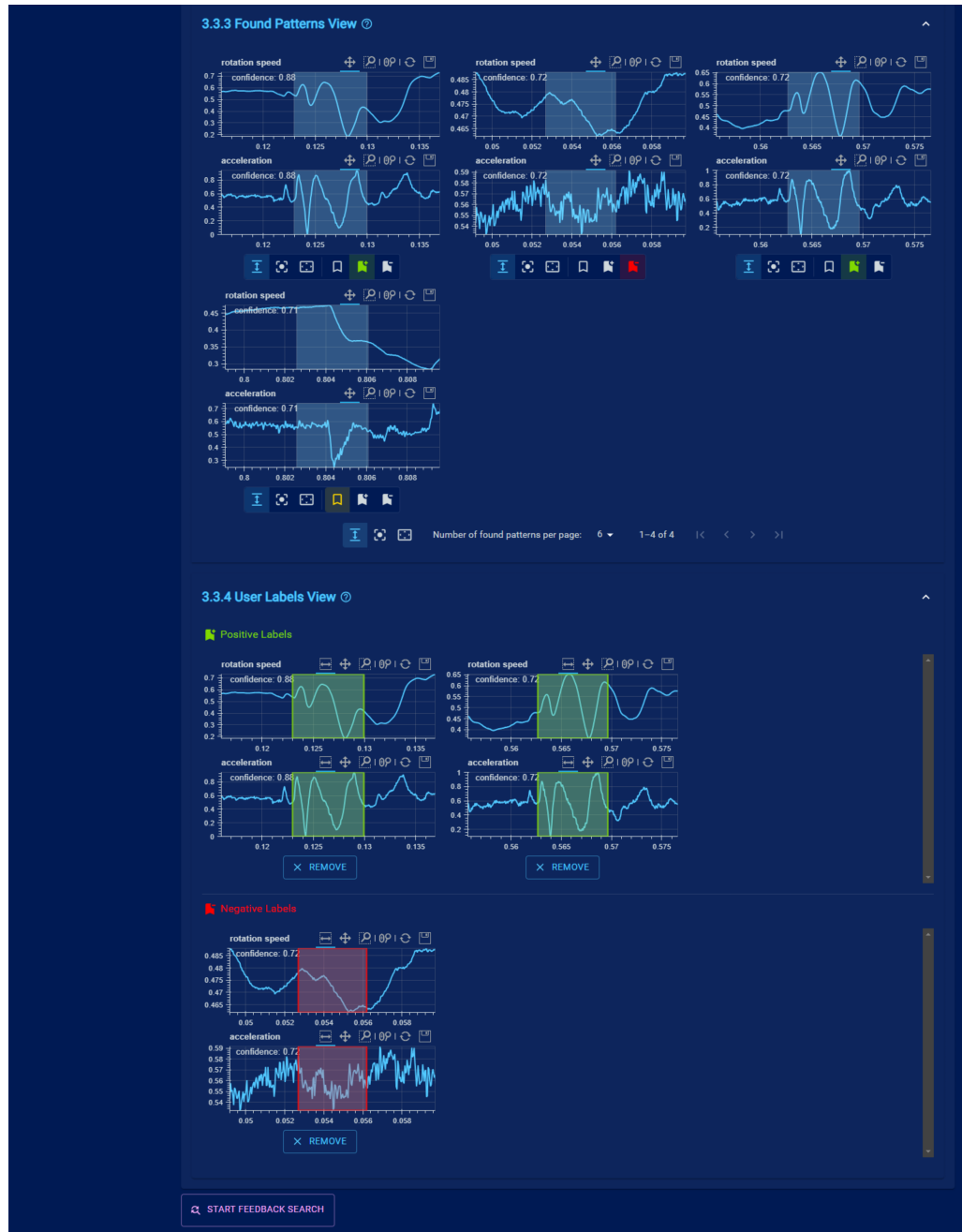


Fig. S9: Initial Search - File 3 - *Found Patterns View* and *User Labels View*.

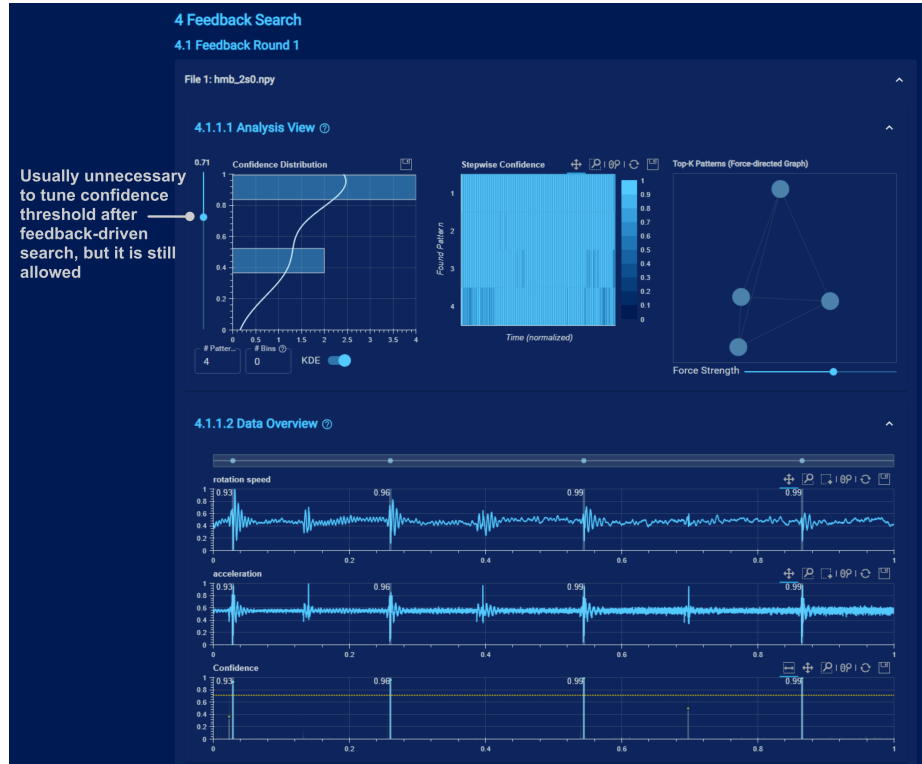


Fig.S10: **Feedback-Driven Search - File 1 - Analysis View and Overview.** The views in feedback-driven search are similar to those in rule-based initial search. In feedback-driven search, it is usually unnecessary to filter the candidates with a confidence threshold or the desired number of found patterns because the machine learning model makes such decision on its own. That said, the found patterns still have confidence and can be filtered. The confidence of a found pattern is simply the average confidence over all time points in the found pattern. Therefore, filtering is still possible and sometimes sensible. In this case/screenshot, we removed two found patterns of relatively low confidence and kept four. In such cases, the model learns to separate the relevant patterns from the irrelevant ones well. However, the position of the boundary between the two is unclear. Because there are not many candidates in feedback-driven search, the candidate dimensionality reduction scatter plot in the *Analysis View* is removed. Because dimensionality reduction does not work well with just a few found patterns. Even if we create it anyway, the plot will not help to find false negatives since there are few candidates (detected patterns with confidence lower than the confidence threshold). In place of the scatter plot, we introduced a heatmap that helps to diagnose the erroneous stage (classification or clustering). Fig. S12 shows an example of how to use this visualization.



Fig.S11: Feedback-Driven Search - File 1 - *Found Patterns View* and *User Labels View*. They are the same as in the rule-based initial search.

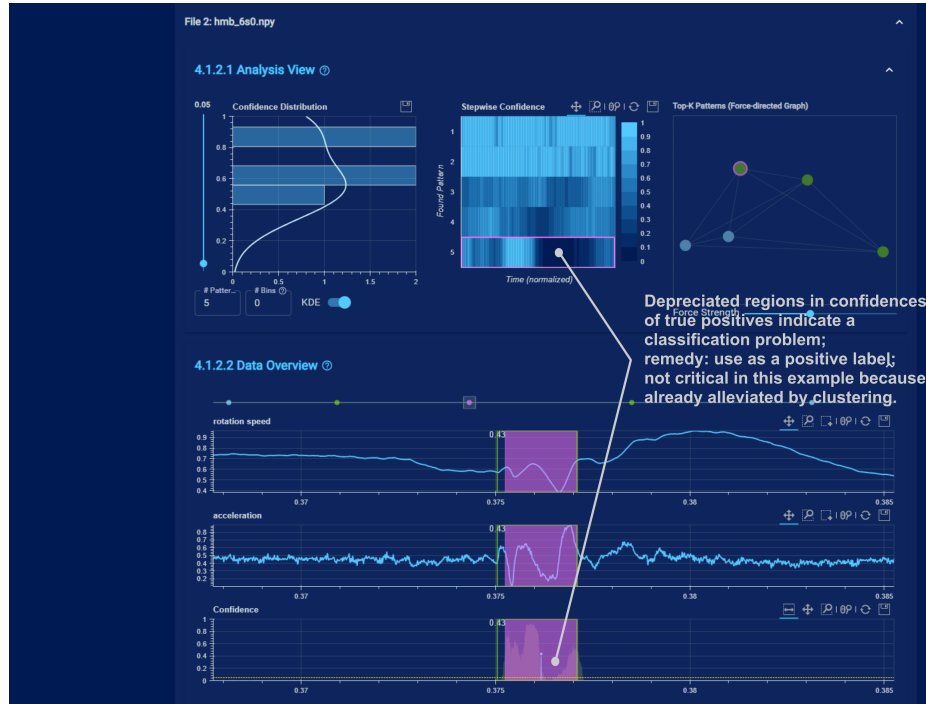


Fig.S12: **Feedback-Driven Search - File 2 - Analysis View and Overview.** If the confidence of the time points is incorrect to a large extent, the classification model is responsible. If the confidence of the time points is reasonable, but the found patterns are incorrect, especially too short or too long, it is likely that the parameters for clustering is inappropriate. Notwithstanding, the clustering parameters are relatively robust. We have set them once with the values detailed in App. E.3 and have kept them throughout all experiments. Therefore, we have not provided input fields for adjusting the parameters for clustering. Otherwise, domain users may feel too overwhelmed as the analysis requires much knowledge of the internal mechanism of the data processing pipeline. In the heatmap and the confidence overlay in the *Overview*, the “dominant” regions with high confidence in the found patterns imply the distinctive features leading to the detection of the patterns, while the “recessive” regions with low confidence may indicate regions with high variance among the target patterns.



Fig.S13: Feedback-Driven Search - File 2 - *Found Patterns View* and *User Labels View*.



Fig. S14: Feedback-Driven Search - File 3.

G Datasets

In this appendix, we describe all datasets as well as the use cases used in the evaluation. There are four datasets. The first three datasets are labeled and used in the quantitative algorithmic evaluation presented in the main text of the paper. The **Deep Valve** dataset has two use cases with distinctive target events. the fourth dataset, namely the **PHEV** dataset, is unlabeled. It was freshly provided by our calibration engineers and used in the expert study described in App. L.

For both nonmyopic and traditional search, we used the earliest (by starting time) ground-truth patterns as positive labels. Negative labels differ by method due to different mechanisms. For nonmyopic search, we provided only two or three relatively long negative labels to show the background. On the other hand, negative labels are optional for nonmyopic search, as explained in App. 3.2. For traditional search, negative labels are mandatory and it makes sense to choose them to have similar lengths as positive labels. We therefore added at least five and up to twenty randomly chosen negative labels, in accordance with the number of positive labels. All labels can be found in this Appendix.

G.1 Metadata of All Datasets

Dataset	Files	Channels	Data Length [time points]	Sampling rate [Hz]	Total Volume [MB]	Query Length [time points]
Deep Valve Whole Signal	1	1	100000	100 Hz	1.52	150
Deep Valve Linear Phase	1	1	100000	100 Hz	1.52	65
Variable Dis- placement	8	2	44595	anonymized	1.05	167
EEG Eye State	1	14	14980	128 Hz	1.72	319
PHEV	3	3	131417	anonymized	5.01	97

Table S5: **Metadata of All Datasets.** Volumes are measured in HDF format. The **Deep Valve** dataset has two use cases / queries, therefore, it takes two rows in the table. The **PHEV** dataset contains many measurement files. However, we used only three in the expert study. Therefore, the number of files for the dataset is three in the table.

G.2 Deep Valve Dataset

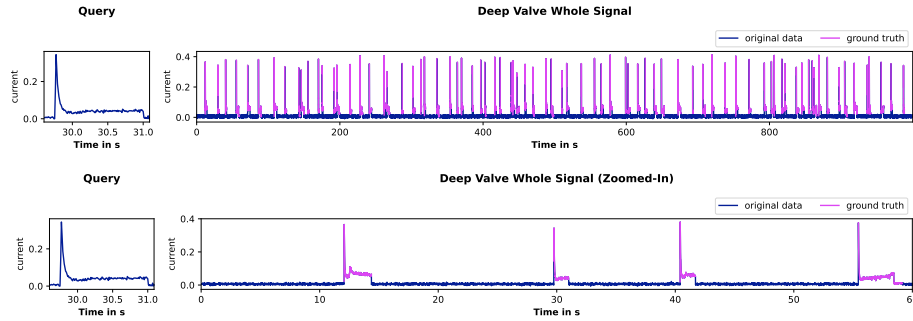


Fig.S15: **Deep Valve Whole Signal Use Case.** The **Deep Valve** dataset records the electric current through a solenoid valve used in the cooling system test bench. The pattern captures a complete valve operating cycle, which consists of maximal five phases (sometimes, an operating cycle may skip some phases), illustrated in Fig. S17. In the first phase, the pattern starts with a peak when the valve is turned on. At this time, the movable part in the valve is still stationary, the electrical impedance is low, and thus there is a current pulse. In the second phase, the electromagnetic force overcomes the mechanical force, and the movable part starts to move. The current plummets and bounces to form a “J”-like shape. In the penultimate linear phase, the movable part moves from one end to another. The valve changes from completely closed to completely open (or a shorter period within these two states, to be rigorous). In this dataset collected from the test bench, the lengths of the linear phases are determined by a random variable and set to be between 0.5 and 4.5 seconds. The length of the linear phases is also influenced by the current state of the valve, i.e., how far the valve is between the two end states (completed open and completed closed). After reaching the stop, the current may deviate a bit from the linear phase and entering the fourth phase. In the last phase, the current is set to drop back to zero, and the valve closes. The duration of the operating cycle varies substantially; the duration of the same phase in different operating cycles may also vary greatly or even disappear. The patterns in the **Deep Valve Whole Signal** use case include a complete operating cycle. The **Deep Valve** dataset is proprietary and therefore cannot be published.

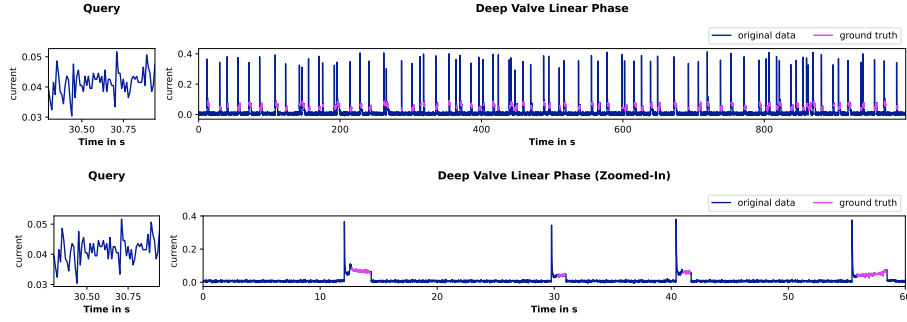


Fig.S16: Deep Valve Linear Phase Use Case. The dataset is the same as the Deep Valve Whole Signal use case. The patterns correspond to the linear phase in the operating cycle of the solenoid valve, during which the movable part physically moves from one end to another, and the valve state changes from completely closed to completely open. This use case reflects the actual needs of our engineers. For precise control and automatic error detection, they want to find a way to detect the linear phases when the movable part in the valve physically moves. There are no conventional domain-specific ways to know the internal state / current working phase of the valve. Luckily, the electric current is indicative. Recognition of this target phase by merely looking at the patterns themselves or the interval characteristics of the patterns is difficult because the linear phase per se does not contain rich features and is not distinctive enough. Moreover, the linear phase may have an increasing or decreasing trend. However, if the context is taken into account, namely the preceding and succeeding phases in the whole signal, the detection becomes more feasible because there are more distinctive features.

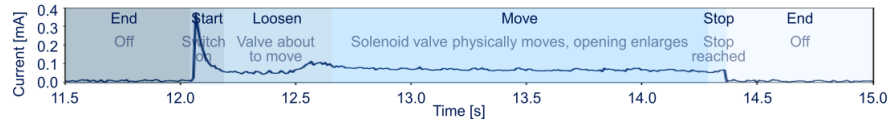


Fig.S17: Target Event in Deep Valve Linear Phase. The target is the “Move” phase in a five-phase process. Though “Move” itself is inconspicuous, its more distinguishable surrounding phases (“Start”, “Loosen”, “Stop”) can aid retrieval.

G.3 Variable Displacement Dataset

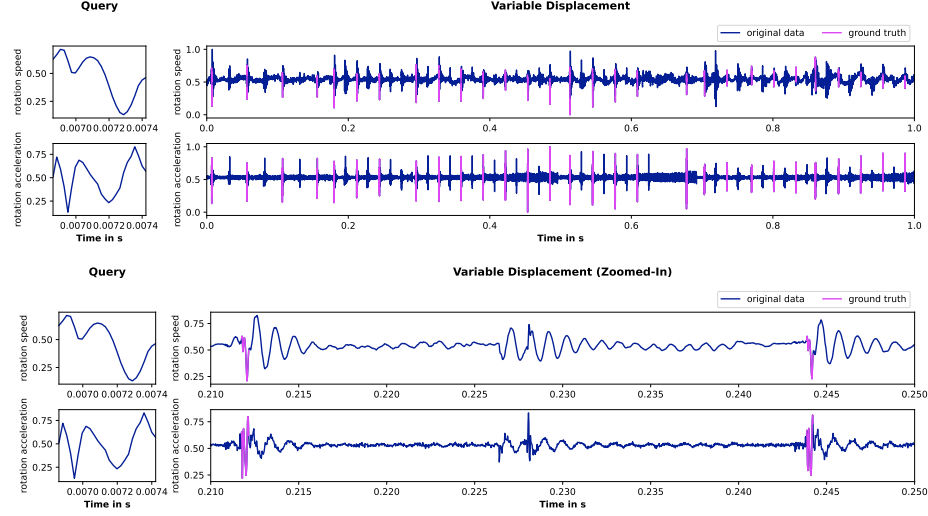


Fig.S18: Variable Displacement Use Case. The Variable Displacement dataset is derived from the rotational speed of a gasoline engine recorded by the engine control unit. It captures the harshness during the engine’s operation in a test automobile. The two channels are extracted features using domain-specific signal processing techniques by our engineers. The first channel corresponds roughly to the rotational speed, and the second to the acceleration. During displacement variation, the rotational speed and acceleration vibrate with large amplitudes, causing Noise Vibration Harshness (NVH) problems. Our engineers try to address this problem. The query corresponds to the transition from full-engine operation to half-engine operation. The data contain eight measurement files. Although our app works for multiple files, we have concatenated the files into one file for convenience during algorithmic evaluation. This dataset is proprietary and therefore cannot be published. In fact, the dataset is even more sensitive than the Deep Valve dataset, and we must normalize the data before showing the plots. Hence, the time spans from 0 to 1, and the values in each channel also range from 0 to 1.

G.4 EEG Eye State Dataset

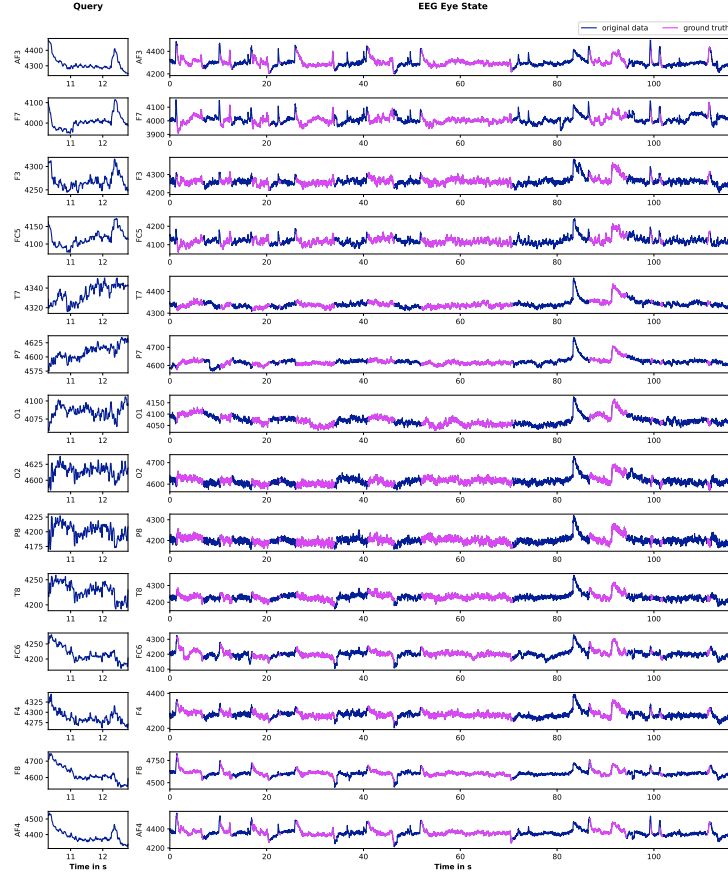


Fig. S19: **EEG Eye State Use Case**. EEG stands for electroencephalogram and measures electrical activity with sensors attached to the subject’s scalp. The **EEG Eye State** dataset comes from one continuous EEG measurement with the Emotiv EEG Neuroheadset. During the measurement, the participant’s eye state (closed or open) was recorded via a camera. The eye state in each video frame was labeled. The typical task for this dataset is to classify each time point (eye closed or open at this time point). The original data have 14 channels plus a channel for ground truths. The patterns correspond to the periods when the subject’s eyes are closed. Visually, the patterns are not well distinguishable and are extensively scaled along the time axis. This dataset is publicly available. Please refer to <https://archive.ics.uci.edu/ml/datasets/EEG+Eye+State> for more information.

G.5 PHEV Dataset

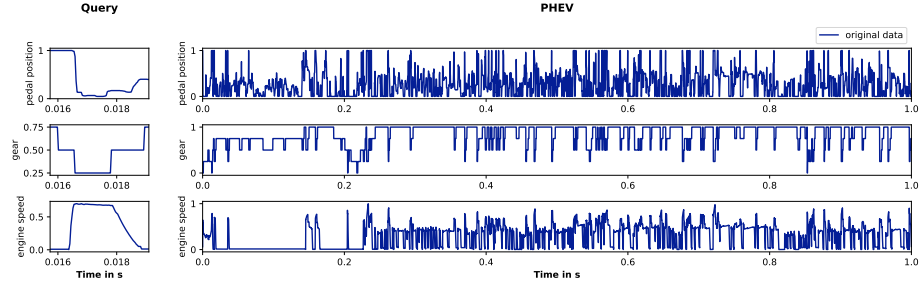


Fig.S20: **PHEV Use Case**. PHEV stands for Plug-in Hybrid Electric Vehicle. It means a car with both an internal combustion engine powered by conventional fuel and an electrical motor powered by a battery pack. The battery pack can be charged not only by a generator driven by the engine but also by an external electric power source. The data came from our calibration engineers, who strive to find the best parameterization for the power chain. There are many channels in the dataset, but according to the engineers, three of them are physically relevant for the target patterns. The first channel is the position of the accelerator pedal, where 0 full throttle and 1 means foot off the gas pedal. The second channel is the current gear. The third channel is the rotational speed of the engine. The target patterns represent an unexpected behavior of the engine that the engineers want to reproduce and investigate. During a target event, the car starts nearly exclusively electrically. Specifically, engine rotational speed is very low at the beginning (0 in the third channel). Then, the driver depresses the gas pedal fully (decrease in the first channel) and demands more power. In reaction, the combustion engine starts (increase in the third channel), albeit with a delay. The transmission shifts to a lower gear (decrease in the second channel) to prepare for an acceleration, and the engine revolution rises undesirably high during start-up (too high a rise in the third channel). During the second half of the pattern, the gear and the engine speed resume. The data are unlabeled (except the query marked by one of the engineers), so we cannot plot the ground truths along with the data. Nor could we use this dataset in the algorithmic evaluation. There are many measurement files in the dataset. The characteristics do not differ significantly among them. Therefore, we have plotted only one file here. Because this dataset was recently provided by our engineers, and they had expressed a great interest in the data, we used it in the expert study. We have chosen the first three files in the dataset for the expert study. This dataset is proprietary and therefore cannot be published. Like the **Variable Displacement** dataset, it is anonymized by normalizing the time and values to $[0, 1]$.

H Evaluation Metric

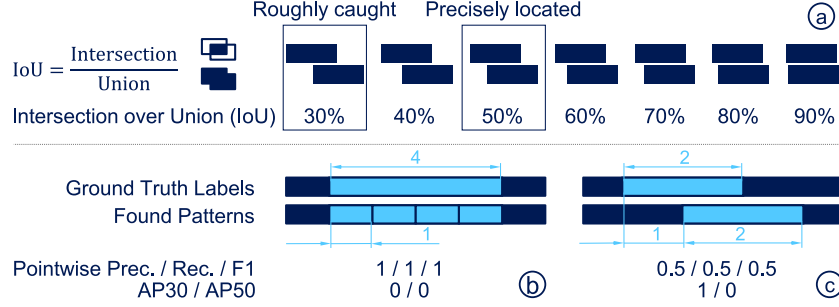


Fig. S21: **Time-Point-Based vs. Pattern-Interval-Based Metrics.** There are cases where the found patterns do not match the target patterns well, yet the time points are all correctly classified. For instance, the algorithm fails to recognize a target pattern as a whole piece in (b). The time-point-based metrics will arrive at high scores despite this problem, while the pattern-interval-based metric AP avoids such pitfalls.

Numerous algorithms for time series pattern search have been proposed. However, a standardized evaluation approach or metric to assess the quality of the found patterns is missing. 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 [9, 3]. Another possibility is to measure differences in results with a more accurate baseline method [4, 16], which is suitable when the proposed algorithm is an accelerated version of the baseline. Luckily, our engineers have provided ground truths. Nonetheless, we still need an evaluation metric that can take information in the ground truths into account.

Suppose a time series $S = [s_i]_{0 \leq i < n}$ 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 devise the metric in two directions: time-point-based and pattern-interval-based.

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

$$Y = \left[y_i = \begin{cases} 1, & \text{if } a \leq i < b, S(a, b) \in \{F\} \\ 0, & \text{otherwise} \end{cases} \right]_{0 \leq i < n} \quad (\text{S11})$$

Likewise, we can derive whether each time point s_i is in a ground truth

$$Z = \left[z_i = \begin{cases} 1, & \text{if } a \leq i < b, S(a, b) \in \{G\} \\ 0, & \text{otherwise} \end{cases} \right]_{0 \leq i < n} \quad (\text{S12})$$

Subsequently, we can count 1) the number of time points 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 points in a found pattern but not in a ground truth (false positives) $FP = |\{i \mid y_i = 1, z_i = 0\}|$, and 3) the number of time points 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-point-based metrics may lead to pitfalls, e.g., in Fig. S21 ⑥, where the algorithm fails to recognize the target pattern as a single piece, despite the correct classification of every time point, and hence, perfect time-point-based scores.

We propose, therefore, to transfer the metric Average Precision (AP) [20] from object detection in computer vision to pattern search in time series. It operates on pattern intervals instead of time points.

AP requires an IoU threshold v to judge whether a found pattern, in accordance with its matching ground truth, is relevant. 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 v ; and 2) G does not overlap another found pattern F' of higher confidence by at least v (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. $IoU(F^*, G) \geq v$ and $\forall F' \in \{F' \in \{F\} \mid IoU(F', G) \geq v\}$, $f_{rel}(F^*) \geq f_{rel}(F')$.

The IoU threshold v regulates the strictness of the metric. Fig. S21 ① shows a series of IoUs. We used 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 (especially results from the baseline methods) would be too low and not well separated, making result comparison difficult. Fig. S21 ② 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-point-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 points.

I Complete Benchmark between Nonmyopic Search and Traditional Search

I.1 Raw Results

The raw results contain information that will reveal the identity of the authors. We will publicize all raw results in an archive upon acceptance.

I.2 Complete Diagrams and Analysis on Retrieval Accuracy and Speed

The evaluation of the nonmyopic search and the traditional search is very extensive. Its inputs have four dimensions: 1) four use cases, 2) nonmyopic search or traditional search, 3) four machine learning model to choose from, 4) four different numbers of sliding window kernels in the case of traditional search. We examined the performance in two aspects: retrieval accuracy (including robustness) and speed.

The retrieval accuracy and speed of traditional search depend on the number of sliding window kernels. We tried 1, 3, 5, and 7 sliding window kernels. Consequently, the performance of traditional search is represented by lines rather than points in the diagram. In our opinion, 7 sliding window kernels is enough in production, provided that the classification model works decently. We use 8 sliding window kernels in rule-based initial search in production. More about the number of sliding window kernels and their actual sizes can be found in App. E.3.

In our experiments, LSTM is much slower than the other models without an advantage in retrieval accuracy. It compresses the visual elements (points and lines) of other models to the left, causing skewed diagrams. Therefore, we also show a version without LSTM because we are more interested in the efficient models.

Fig. S22, Fig. S23, Fig. S24, and Fig. S25 show retrieval accuracy measured in AP30 versus execution time in wall-clock time in seconds in all use cases, without and with LSTM. Based on these diagrams, we can draw the following conclusions.

- **(Key Conclusion 1) Retrieval Accuracy.** Nonmyopic search is much more accurate than traditional search because the points denoting nonmyopic search are much higher than the lines denoting traditional search. In fact, some models in traditional search do not work at all in some use cases, indicated by the low lines. An extreme case is the **Deep Valve Linear Phase** use case in Fig. S23, where the target patterns (linear phases) themselves do not have distinctive characteristics. It is the contexts of these target patterns that are telling. Consequently, nonmyopic search’s consideration of the pattern contexts contributes much to the detection. But even in other cases, where the contexts of the target patterns themselves are not as helpful, nonmyopic search is still substantially more accurate than traditional search.

One possible explanation is the efficient usage of the labels. In nonmyopic search, a label is used as many training instances from slightly different temporal perspectives. A labeled found pattern comprises of many time points, and each time point corresponds to a training instance during model training. In traditional search, a labeled pattern can only be used as a single training instance. Translational data augmentation is also unimaginable. In this sense, nonmyopic search better respects the nature of the data type, i.e., time series, during model training. Because it lets the model see an event from different temporal perspectives. Whereas, traditional search only shows one perspective of a target pattern.

- **(Key Conclusion 2) Speed.** Nonmyopic search is generally faster than traditional search because the points denoting nonmyopic search are generally located left to the lines denoting traditional search. An exception is LSTM, which is by far the slowest model and is not recommended anyway. Although nonmyopic search spends considerable time in training due to the many more training samples, it has an edge during inference because it only scans the data once (please refer to the raw data for training time and inference time in App. I.1). Moreover, training time will be dominated by inference time, because training time is determined by the number of labels, which is generally small, while inference time by the size of the data, which is generally large. Therefore, we expect a greater advantage of nonmyopic search when the data are bigger. On a side note, this reminds us of **You Only Look Once (YOLO)** in object detection in computer vision, which passes images through the model only once, making it faster than its competitors like **Faster R-CNN**. While YOLO suffers from a minor accuracy sacrifice in exchange for speed, our pipeline actually boosts the accuracy enormously, as explained by Key Conclusion 1.
- **Robustness.** Nonmyopic search is more robust according to the plots, because 1) the retrieval accuracy of various models used in nonmyopic search are comparable (of similar height), while the retrieval accuracy of various models used in traditional search vary significantly; and 2) the error indicator bars showing the standard deviation of retrieval accuracy in five runs with the same model in nonmyopic search are so small that they are nearly imperceptible, while the error indicator bars for traditional search are sometimes relatively long, e.g., in Fig. S22.
- **Model Accuracy.** In nonmyopic search, except for LSTM’s less satisfactory performance, all models achieve comparable retrieval accuracy. On the contrary, in traditional search, **random forest** clearly performs the best, showing the model’s general capability (rather than the pipeline’s capability) for the task time series pattern search.
- **Model Speed.** In both pipelines, **random forest** and **XGBoost** run the fastest. Note that GPUs are not used during training. If GPU is enabled, we expect **XGBoost** to run fastest. Early stopping is enabled when possible, namely for **XGBoost**, **LSTM**, and **CNN**, not for **random forest**, which does not support early stopping.

- **Model Memory Consumption.** Random forest and XGBoost require all data and labels in memory, which may cause out-of-memory error when the memory resource is limited. Whereas, neural networks like CNN and LSTM work in batches and enjoy better memory consumption.
- **Number of Kernels.** This point is only relevant for traditional search. Increasing the number of sliding window kernels does not automatically lead to better accuracy. Whereas, the execution time is guaranteed to increase approximately proportionally to the number of kernels. It is likely that the model in traditional search fails to steer the model during training anyway, and increasing the resolution (number of differently sized sliding window kernels) during inference does not work for poorly trained models.

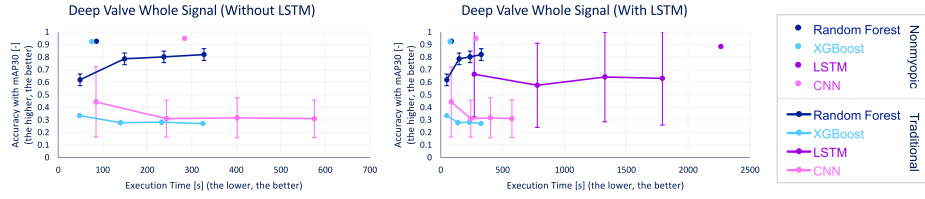


Fig. S22: Deep Valve Whole Signal Use Case.

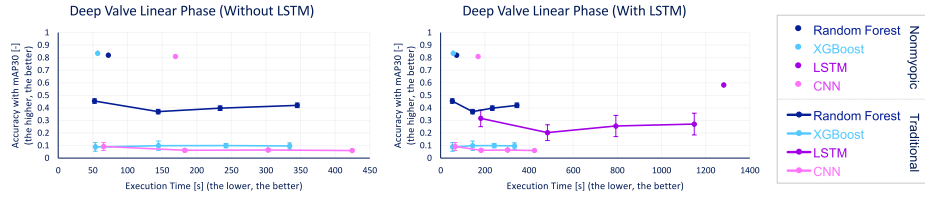


Fig. S23: Deep Valve Linear Phase Use Case.

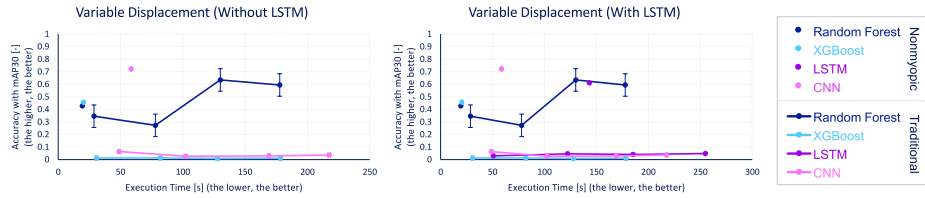


Fig. S24: Variable Displacement Use Case.

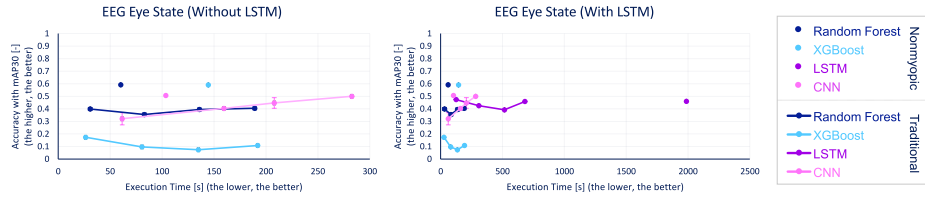


Fig. S25: **EEG Eye State Use Case.** CNN in traditional search with three kernels seems to take too long to run. However, we repeated the experiment and got the same result. Due to the time limit, we haven't checked the root of this issue but report it here.

I.3 Visual Inspection

Please find in the OSF project under <https://osf.io/xudk7> the plots with the data, the ground truth, the positive labels, the negative labels, and the found patterns (called “predictions” in the plots).

We cannot place the plots in this appendix because there are too many plots. Specifically, there are four use cases, two search pipelines (nonmyopic search and traditional search), and four models; for traditional search, there are four choices for the number of sliding window kernels; except for the **EEG Eye State** use case, whose time series data are short, we kept not only the plots containing the whole data, but also a zoomed-in plot to better show the quality of the found patterns. Therefore, there are 60 plots in total.

For demonstration, Fig. S26 shows an example plot for visual inspection in the **EEG Eye State** use case with **random forest** in nonmyopic search.

As mentioned in App. 4, each experiment for retrieval accuracy is repeated five times, and each experiment for speed is repeated ten times. The results of the five runs are visually similar. Therefore, we have kept only the plots in the first runs. Please also note that the benchmark between nonmyopic search and traditional search only involves running the feedback-driven search once. Thus, the result is likely not converged, especially for the **Variable Displacement** use case. Benchmarking of the converged retrieval accuracy is a topic in App. J.1 as well as in App. J.4.

The positive labels and negative labels may cover the ground truths in the plots and make some of the ground truths completely or partially invisible. Please refer to App. G for complete ground truth labels.

Generally, we can observe that the found patterns after a feedback-driven search iteration with nonmyopic search are fairly decent, while the found patterns after a feedback-driven search iteration with traditional search are often quite problematic.

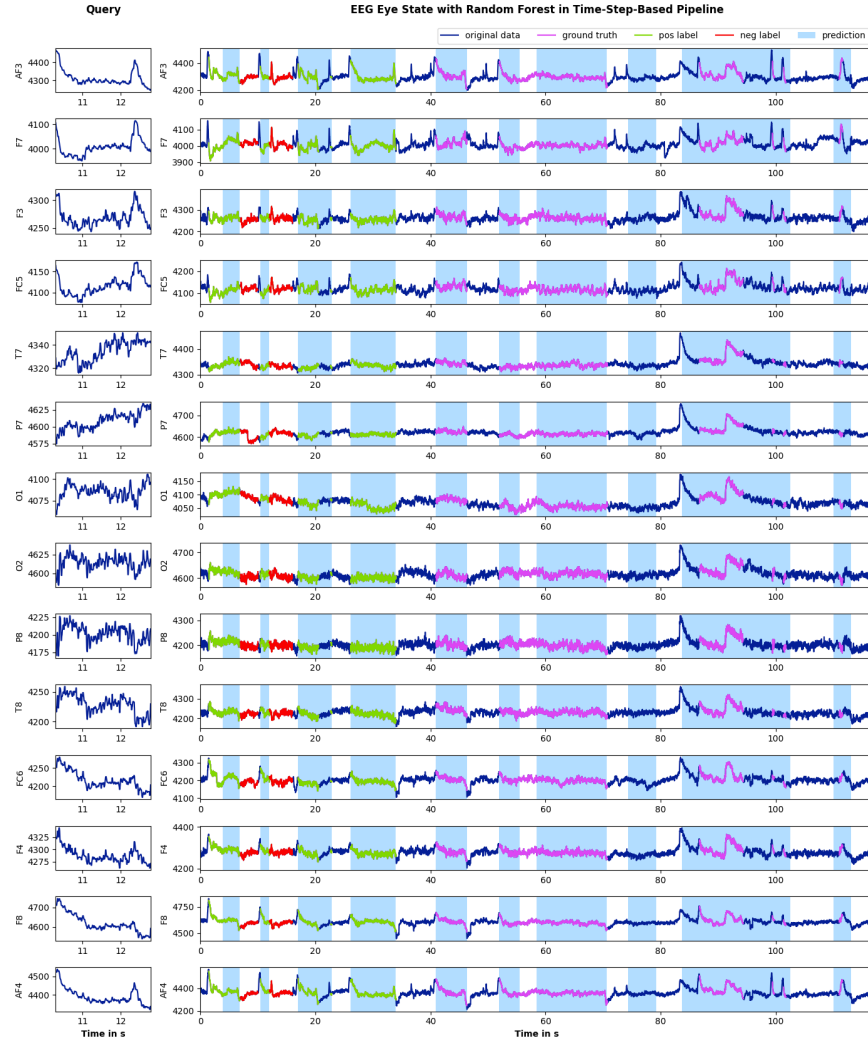


Fig. S26: An Example Plot for Visual Inspection with the EEG Eye State Use Case. It shows the time series data, ground truths, positive labels, negative labels, and found patterns (called “predictions” in the legend of the plot) in the EEG Eye State use case with random forest in nonmyopic search (called “time-point-based pipeline” in the plot title). The positive labels are also ground truths (for all ground truths uncovered by labels, please refer to App. G; specifically for the EEG Eye State use case, please refer to App. G.4). The query is used in the rule-based initial search, not in the feedback-driven search. It is plotted only for reference as a typical zoomed-in target pattern. In this use case, we used the first four ground truths as positive labels. They are all at least partially found. We provided two negative labels. As expected, they are not in the found patterns. The other ground truths (not as positive labels) are mostly covered by the found patterns. However, the long ground truth around 60 second is recognized as two found patterns. The found pattern before 80 second is a false positive. And the long found pattern that begins after 80 seconds and ends shortly after 100 seconds subsumes three ground truths, among them two are extremely short. As mentioned, there are 60 such plots, and this figure only serves as an example.

I.4 Diagrams of Training and Inference Time for All Use Cases

Fig. S27 shows the training and inference time with **random forest** in nonmyopic search and traditional search with different numbers of sliding window kernels. We only show plots for the **random forest**. The qualitative relationships between different execution time periods are typical. The training and inference time for other models are similar. We choose **random forest** because it is the best model in terms of retrieval accuracy and speed for traditional search, and it has also shown strong performance in nonmyopic search. Therefore, it is suitable for the task time series pattern search.

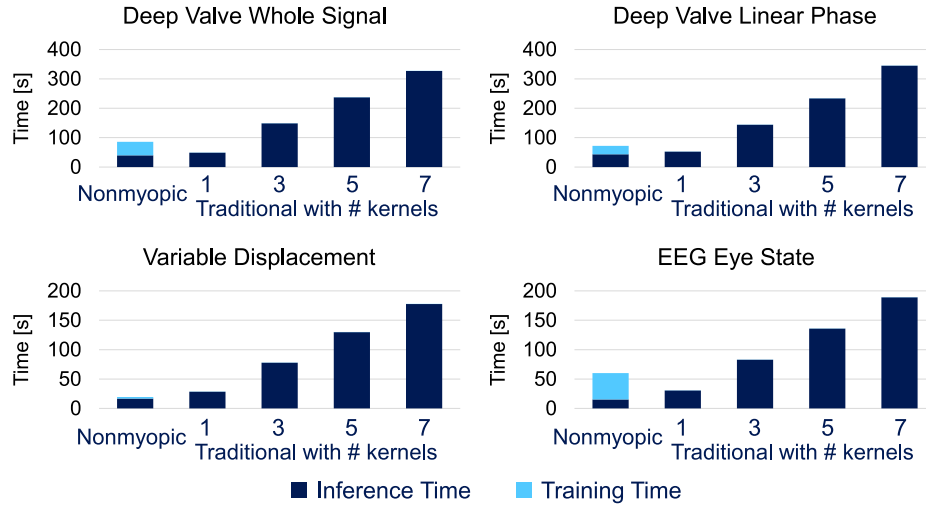


Fig. S27: **Training and Inference Time for All Use Cases with Random Forest.** This series of diagrams shows the relationships between training and inference time in nonmyopic and traditional search. For one thing, the inference time of nonmyopic search is comparable with traditional search with one sliding window kernel, both scanning the data only once. The execution time in traditional search scales roughly proportional to the number of sliding window kernels. When comparing nonmyopic search with traditional search with five sliding window kernels, which is coarse enough (we use eight sliding window kernels in production in rule-based initial search), nonmyopic search clearly has an edge. For another, while the training time in traditional search is negligible, the training time in nonmyopic search is considerable due to the massive increase in training instances. However, training time is not a concern for large data because training time is determined by the number of training instances, which in turn is determined by the number of user labels, which is usually small. Whereas, inference time scales with the size of the time series data, which is generally large. For traditional search, the low training time actually hints at its low retrieval accuracy.

I.5 Performance Summarization

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 Tab. S6 derived from the raw results under <https://osf.io/36kf2>.

Use Case	Retrieval Accuracy with AP30			Execution Time in Seconds		
	Nonmyopic	Traditional	Gain	Nonmyopic	Traditional	Reduction
DV-Whole	0.93	0.76	0.17	86	237	64%
DV-Linear	0.82	0.41	0.41	72	234	69%
VD	0.43	0.46	-0.03	19	130	85%
EEG	0.59	0.39	0.20	60	136	56%
Average	0.69	0.50	0.19			68%

Table S6: **Simplified Performance Gain with Nonmyopic Search.** If limited to **Random Forest** (the most accurate and fastest model in traditional search) and to five sliding window kernels, nonmyopic search increased retrieval accuracy by 0.19 and reduced execution time by 68% on average. (DV-Whole: Deep Valve Whole Signal, DV-Linear: Deep Valve Linear Phase, VD: Variable Displacement, EEG: EEG Eye State)

J Complete Benchmark between Multi-Feature Feedback and Binary Relevance Feedback

J.1 Multi-Feature vs. Binary Relevance Feedback - An Overview

This section benchmarks multi-feature feedback with traditional relevance feedback quantitatively on retrieval accuracy. Since multi-feature feedback creates higher-quality labels, better accuracy is expected. Our aim, however, is to verify that the improvement is substantial and cannot be omitted. Given its strong performance in App. 4, nonmyopic search with **Random Forest** is used.

Designing experiments involving users is challenging due to the entailed subjectivity. Following [17, 19], we simulate user behavior with two agents: the BR-Agent (Binary Relevance) and MF-Agent (Multi-Feature), shown in Fig. S28.

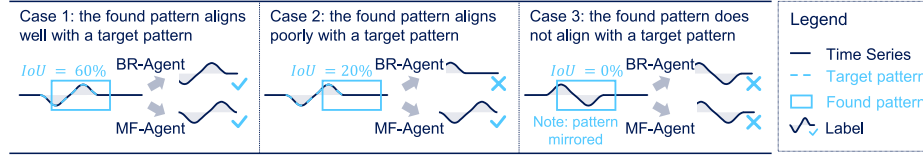


Fig. S28: **BR-Agent vs. MF-Agent.** The BR-agent labels a found pattern positive if it overlaps a ground truth sufficiently, otherwise negative. The MF-agent labels a found pattern positive, as long as it overlaps a ground truth, and the target pattern is used as the label instead of the found pattern. The MF-agent only labels a found pattern negative when it does not overlap any ground truth.

The BR-Agent accepts a found pattern F if: 1) it overlaps a ground truth G by at least the IoU threshold v^* (set to 30%), 2) it does not overlap any already added positive label F' by v^* . If F fails 1), the BR-Agent labels it negative. If it passes 1) but fails 2), the BR-Agent ignores it.

The MF-Agent adds a found pattern F as a positive label if it intersects any ground truth G , regardless of their IoU. This setting assumes the user can notice G upon seeing F . Instead of labeling F , the MF-Agent adds G as a label (if not already labeled), simulating the user's correction to the pattern position and size. F is labeled negative only when it does not intersect any ground truth.

The experiment began with imperfect patterns from the rule-based initial search. Each agent evaluated these patterns against the ground truths and returned five labels. Because **Random Forest** and **XGBoost** do not support incremental training, we retrained models at each iteration on the new and all the old labels. We ran 10 feedback iterations, which is sufficient for retrieval accuracy to converge in all use cases.

Fig. S29 shows AP30 and AP50 after the initial rule-based search and each of ten feedback iterations. By AP30, both feedback types can improve upon the rule-based baseline (Iteration 0), e.g., with the rising AP30 in **Deep Valve Whole**

Signal. In the **Deep Valve Whole Signal** use case, multi-feature feedback improves the final AP30 by about 0.05 over binary relevance feedback. Other use cases show a larger gain. By AP50, the difference is even clearer: accuracy with binary relevance feedback barely increases, whereas it rises to a much higher level with multi-feature feedback.

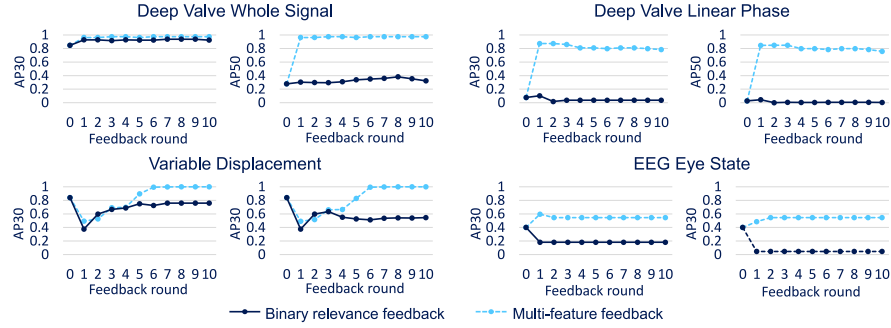


Fig. S29: Evolution of Retrieval Accuracy. Both feedback types can increase retrieval accuracy, mainly after the first iteration. Multi-feature feedback consistently outperforms binary relevance feedback under both AP30 and the stricter AP50. In fact, binary relevance feedback fails to improve accuracy in some cases.

In conclusion, multi-feature feedback outperforms binary relevance feedback and should be established for feedback-driven pattern search, especially when combined with nonmyopic search. More experiments and deeper analyses are documented in the following.

J.2 Raw Results

The raw evaluation results are kept in the second sheet titled “multi-feature vs. binary rel.” in the file “evaluation.xlsx” under <https://osf.io/36kf2>. Instead of viewing the file directly in OSF, please download the file and open it locally to see the plots and formatted content.

J.3 Accuracy Evolution

Please refer to Fig. S30.

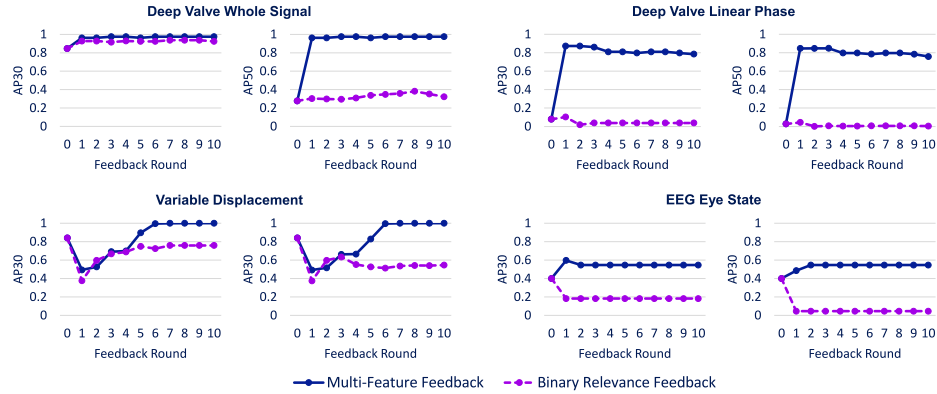


Fig. S30: Evolution of Retrieval Accuracy in All Use Cases. While both feedback types may improve accuracy, as indicated by the rising AP30 in the **Deep Valve Whole Signal** use case, multi-feature feedback significantly outperforms binary relevance feedback, as verified by the blue curves denoting multi-feature feedback above the purple curves denoting binary relevance feedback. In fact, binary relevance feedback does not always work, as the low purple lines in the **Deep Valve Linear Phase** use case show. In this use case, the temporal contexts of the target patterns need to be taken into consideration during pattern search. Hence, correction of positions and sizes is especially helpful in this case. Multi-feature feedback also significantly outperforms the binary relevance feedback in the **EEG Eye State** use case, where boundaries between the target patterns and the background are relatively difficult to delineate, requiring the user's help. Not only does multi-feature feedback catch more target patterns correctly, as indicated by the higher AP30s, but the true positives are also of higher quality (more precise locations and sizes), as indicated by the higher AP50s. In fact, binary relevance feedback with IoU 30% as the boundary between true positives and false positives never really improves AP50, suggesting that training with low-quality labels may not result in a high-quality outcome. Another finding is that the retrieval accuracy is not guaranteed to increase monotonically; it may deteriorate, not only for binary relevance feedback, but also for multi-feature feedback, as in the **Deep Valve Linear Phase** use case. A worse result is observed after the first feedback round in the **Variable Displacement** use case. This is probably due to the limit we set for the feedback. Specifically, we allow only five new labels in each feedback round. The rule-based initial search already works relatively well in this case. However, the first feedback round only has five labels to begin with, which can be challenging for model training, leading to worse retrieval accuracy. Finally, even multi-feature feedback may stagnate, as in the **EEG Eye State** use case, where the search result converges to a suboptimal state. That said, the found patterns in visual inspection, e.g., in Fig. S26, count as a reasonable outcome and indicate that the model training goes in the right direction.

J.4 Converged Accuracy

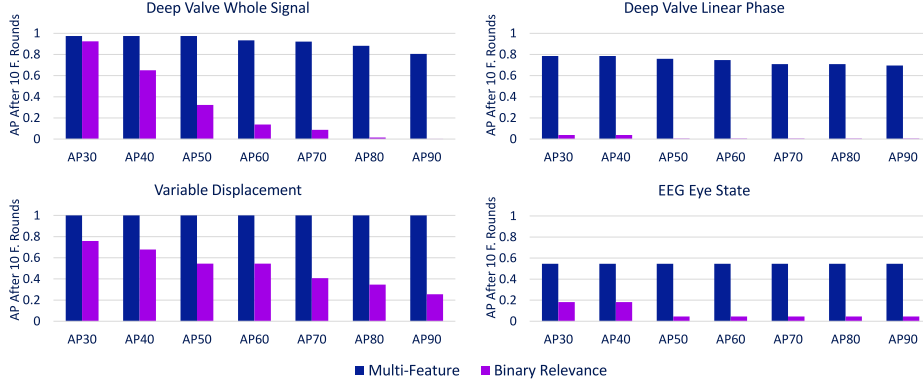


Fig. S31: Converged Accuracy of All Use Cases. It can be seen that the results from multi-feature feedback stand the test of increasing strictness of the metric or the increasing IoU threshold, indicating the higher quality of the found patterns in terms of their positions and sizes. Even just looking at the leftmost pair of bars in the four bar charts (with the lowest metric strictness), we can see that the multi-feature feedback leads to higher retrieval accuracy, implying that the multi-feature feedback catches the target patterns better. Binary relevance feedback does not work at all in the **Deep Valve Linear Phase** use case and the **EEG Eye State** use case. The target patterns in the **Deep Valve Linear Phase**, as illustrated in Fig. S17 and Fig. S16, represent a linear phase in a process. The internal characteristics of this phase do not contain rich features for detection. Luckily, the surrounding phases, e.g., the “J”-shaped phase, are distinctive and can contribute to the search. However, if the labeled found patterns are poorly bounded, as in binary relevance feedback, model training may not benefit much from this advantage because the characteristic context is sometimes in the training instances and sometimes not, making this regularity that could have been relied on for detection irregular. In the **EEG Eye State** use case, the start and end time points of the target patterns are very blunt. It is challenging for the model if the labeled found patterns do not have sharp bounds because there is basically no clear information for the model to learn how to locate the start and end time points. Moreover, the special data augmentation that labels the time points surrounding a positive label, introduced as the third benefit in App. 3.2, requires that the labeled found patterns have correct start and end time points. Otherwise, it may cause incorrect negative training instances. All in all, the adjustment of the pattern positions and sizes contributes greatly to the detection of the target patterns, especially for nonmyopic search.

J.5 Visual Inspection

We show the found patterns after ten feedback-driven search iterations with multi-feature feedback in this section. As can be seen in Fig. S30, ten feedback rounds are sufficient for retrieval accuracy to converge in all our four use cases. Please find all plots under <https://osf.io/ntemw>, including converged found patterns with binary relevance feedback.

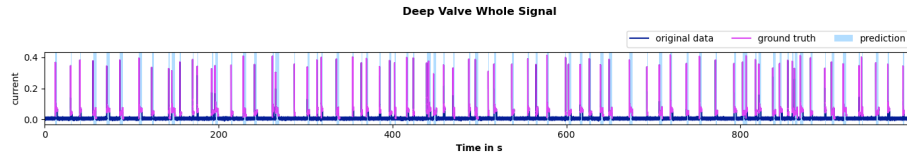


Fig. S32: Deep Valve Whole Signal.

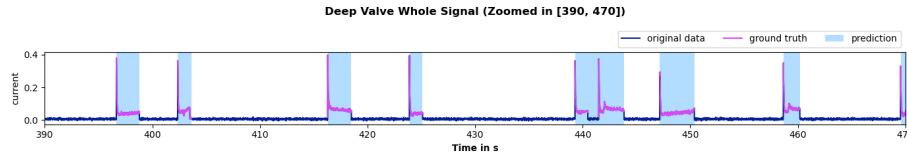


Fig. S33: Deep Valve Whole Signal (Zoomed in).

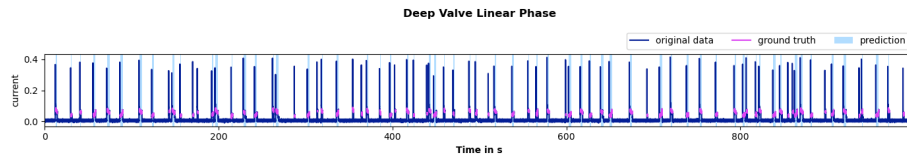


Fig. S34: Deep Valve Linear Phase.

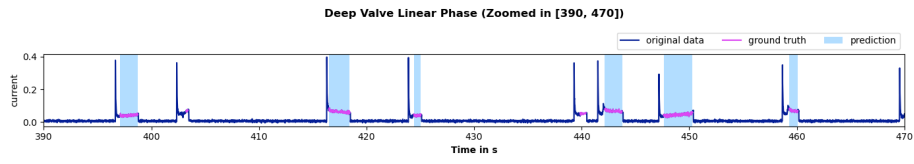


Fig.S35: Deep Valve Linear Phase (Zoomed in).

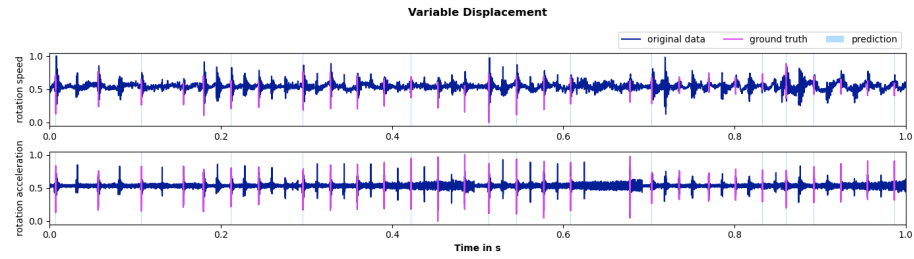


Fig.S36: Variable Displacement.

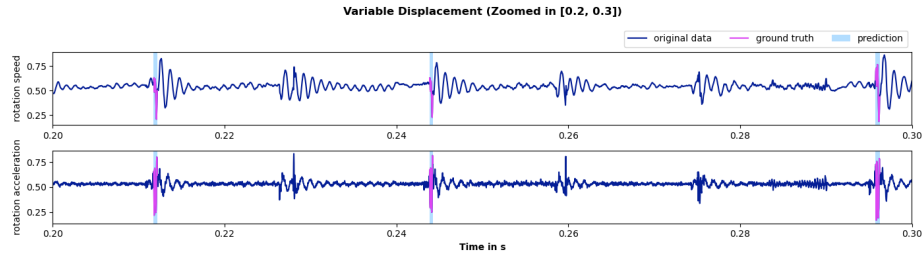


Fig.S37: Variable Displacement (Zoomed in).

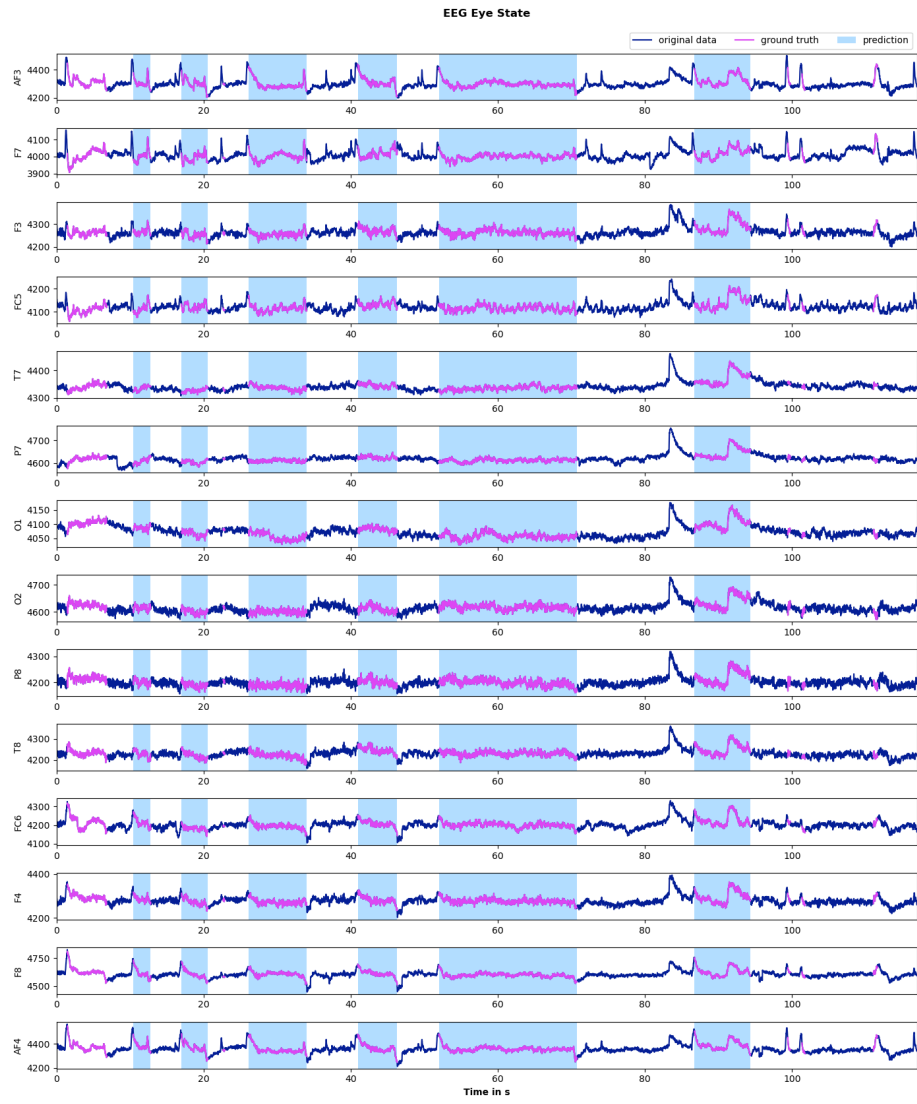


Fig.S38: EEG Eye State

J.6 Performance Summarization

As shown in Tab. S7 (raw results under <https://osf.io/36kf2>), multi-feature feedback raised retrieval accuracy on average by 35% and 59% according to AP30 and AP50, respectively.

Use Case	AP30				AP50		
	MF	BR	Gain		MF	BR	Gain
DV-Whole	0.97	0.92	0.05		0.97	0.32	0.65
DV-Linear	0.78	0.04	0.75		0.76	0.00	0.75
VD	1.00	0.76	0.24		1.00	0.54	0.46
EEG	0.55	0.18	0.36		0.55	0.05	0.50
Average	0.83	0.48	0.35		0.82	0.23	0.59

Table S7: **Simplified Performance Gain with Multi-Feature Feedback.** Multi-feature feedback raised retrieval accuracy on average by 0.35 and 0.59 according to AP30 and AP50, respectively. (DV-Whole: Deep Valve Whole Signal, DV-Linear: Deep Valve Linear Phase, VD: Variable Displacement, EEG: EEG Eye State)

K Case Study

In this appendix, we show step-by-step screenshots during the case study. The video <https://www.youtube.com/watch?v=7-MPiC7asBg&t=827s> demonstrates the same use case in action. To be clear, no actual user was involved in the case study. Instead, we demonstrate the functions of the App in a real-world use case ourselves. We have chosen the **Variable Displacement** use case because it has more than one channel and contains multiple files, while the other use cases have only one file or contain only one channel. Due to compliance reasons, we have normalized the data, so the time spans from 0 to 1, and the value range also expands from 0 to 1 in each data file. The use case has eight files. We used all of them in the algorithmic evaluation. For the case study, we chose the first three. They should be sufficient to showcase the features and verify the usability of the UI. In this case study, we can see how the imperfect result from the rule-based initial search was improved by feedback-driven search. Specifically, there were some false negatives and some false positives after the rule-based initial search. After one feedback-driven search round, the missing ones were found, the incorrect hits were removed, and the result was nearly perfect. In the case study, we tried to test and demonstrate as many interactive features of the UI as possible. However, the only mandatory user interaction is the provision of at least two true positives / correctly found patterns in each feedback-driven search iteration. While other user interactions help the user explore target events in the data, as well as understand and diagnose the algorithm, they are optional.

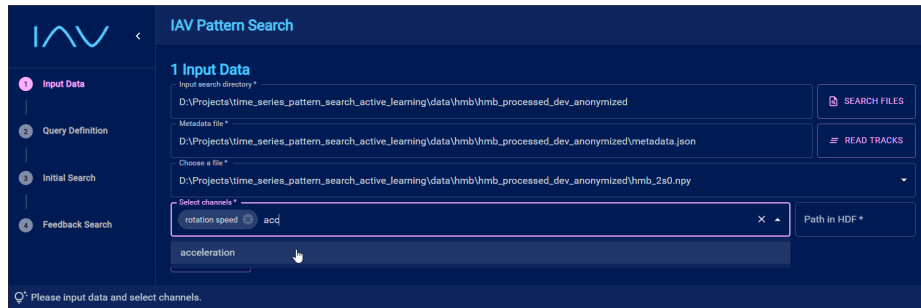


Fig. S39: **Input Data.** The user input the directories of the time series measurement data and the metadata (mainly a list of channels in the data). The user selected a file and the channels relevant to the target event for inspection. In case of HDF-files, the user also needed to specify the path to the data within the HDF-file.



Fig.S40: **Inspect Data.** After clicking the button “Plot Data”, the selected channels in the chosen data file were plotted for inspection. The selected two channels roughly correspond to the engine rotational speed and acceleration. They were extracted by our engineers based on domain-specific signal processing techniques. We can see the harshness, or periods where the signals vibrate with high amplitudes, in the data. They occurred during transitions between half-engine operation and full-engine operation in a variable displacement engine. Because there are two types of transitions, namely, from half-engine to full-engine and vice versa, there are also two types of events. The transition from full-engine to half-engine looks more interesting because the harshness is more severe.

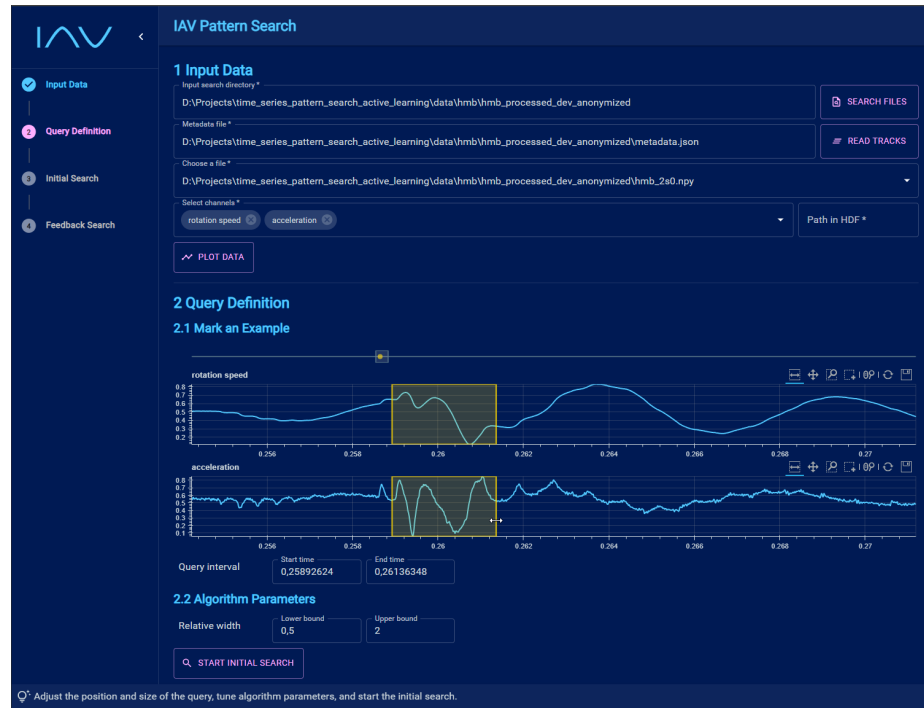


Fig. S41: **Mark a Query.** The user zoomed in on a target pattern and marked it as the query to search for. The query describes the transition phase from full-engine operation to half-engine operation for a variable displacement engine. This period is notorious for NVH problems, and our engineers strive to alleviate the problems.



Fig.S42: **Start Initial Search.** The user clicked the button “Start Initial Search”, to run the quick-and-dirty rule-based initial search. The status bar reported on the processing status of the backend (computing similarity profiles, conducting dimensionality reduction for the scatter plot, calculating pairwise similarity between found patterns for the force graph, etc.)

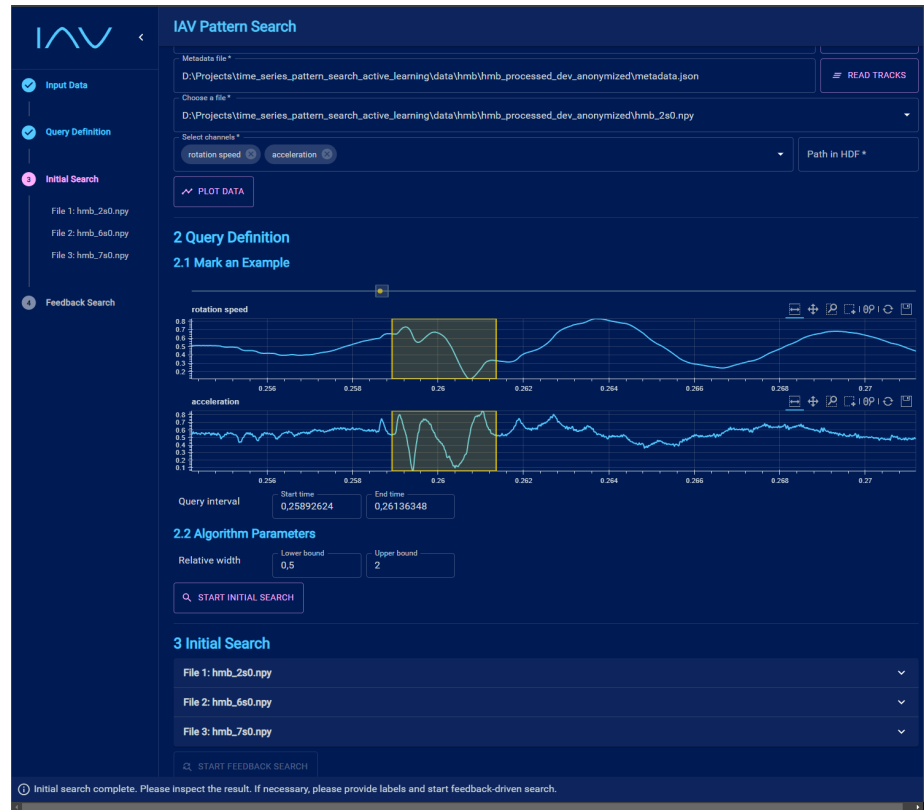


Fig.S43: **Initial Search Complete.** Upon completion, the user noticed a new section titled “3 Initial Search” with three collapsed areas corresponding to the three data files in the given directory.

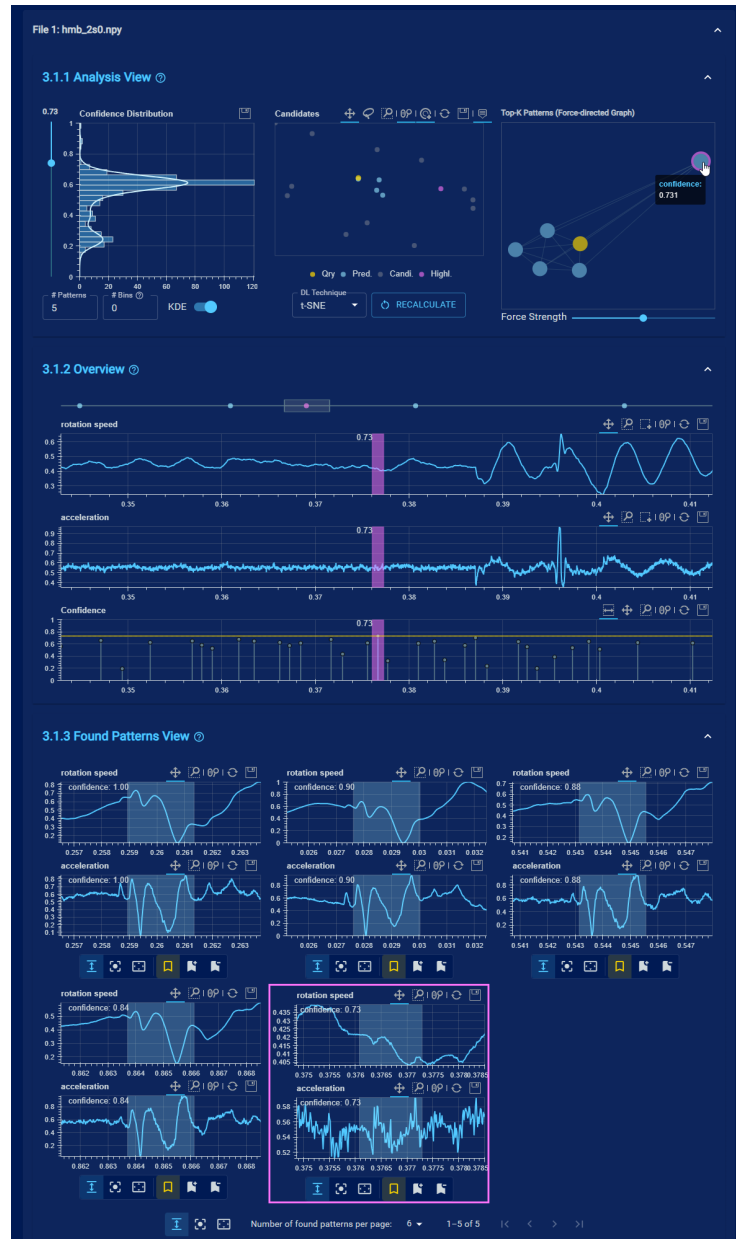


Fig.S44: Initial Search Result (File 1). (Caption on the next page)

Fig. S44: **Initial Search Result (File 1)**. The user expanded the first file. As a sanity check, the found patterns in the *Found Patterns View* looked reasonable. The dimensionality reduction scatter plot titled “candidates” in the *Analysis View* did not suggest any potential false negatives (missing target patterns) for examination, as there were no candidates (dark blue points) that were more similar (closer) to the query (the yellow point) than the found patterns (bright blue points). However, the user spotted a node / found pattern far from the cluster around the query in the force graph. It hinted at a false positive (incorrectly found pattern). The user hovered the cursor over the lonely node and noticed in the tooltip that its confidence was not particularly high. The user clicked on the node in the force graph to highlight it. The pink highlight appeared not only in the force graph, but also the dimensionality reduction scatter plot, in the *Overview*, and in the *Found Patterns View*. In the *Found Patterns View*, the user found that the highlighted pattern indeed did not look like a target pattern. Above all, its value range was much smaller. The user clicked the second button in the button group  to display the found pattern in the same value range as the query. After that, this suspicious found pattern became almost a horizontal line in both channels. Accordingly, this found pattern was not relevant and belonged to background noise. The user checked this pattern in the *Overview*, where the context of the pattern confirmed the suspicion. Luckily, this found pattern ranked the lowest among the found patterns, because the found patterns were displayed in descending order of their confidence, and this found pattern was the last one. This meant that the user could easily remove this false positive by raising the confidence threshold, or even simpler in this case, by reducing the number of found patterns by one. This can be achieved in the first input field beneath the confidence distribution histogram in the *Analysis View*. In the meantime, the user noticed a tiny cluster with high confidence in the histogram. It is often a good idea to set the confidence threshold just below this cluster, should it exist. The dimensionality reduction scatter plot and the force graph both transform high-dimensional patterns into a 2D space. The former performs better with many instances and is used to visualize the pattern candidates that are usually massive after the initial search, if not filtered with a confidence threshold. Whereas, the force graph does not scale well and is used to visualize the usually scarce found patterns. Furthermore, the user can move the nodes in the force graph to sense the robustness of the layout. Sometimes, the two even hold opposite views, as we will show in Fig. S45.

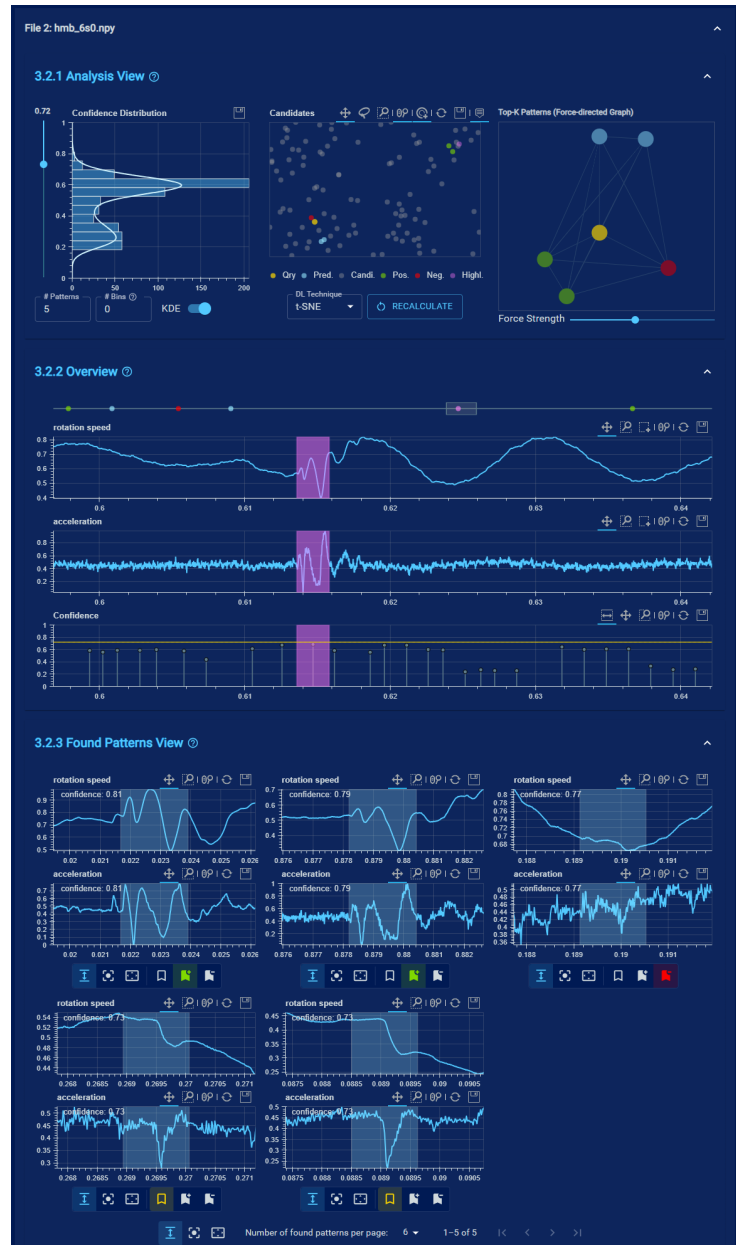


Fig.S45: Initial Search Result (File 2). (Caption on the next page)

Fig. S45: **Initial Search Result (File 2)**. The user expanded the second data file. First, the user scanned the found patterns in the *Found Patterns View*. The top two found patterns looked potentially relevant, while the others irrelevant. The user then skimmed the *Overview* and realized that there were definitely more than two target patterns in the data file (please note the long empty area in the navigation slot above the line chart. For example, we highlighted a pink point, which was obviously a false negative). Consequently, the user turned to the dimensionality reduction scatter plot, whose function is to suggest potential false negatives. There, the user did find some candidates that were even closer to the query than some found patterns. The user clicked on these candidates in the scatter plot to highlight them in all views. After that, the user checked them in the *Overview*. Unfortunately, the user did not find more target patterns. This indicated that the rule-based initial search did not work well in separating the relevant patterns from the irrelevant ones, and the user should resort to feedback-driven search in order to improve the result. On the other hand, there were two found patterns faraway from the query in the scatter plot. The user thought that they were probably false positives. Surprisingly, that was quite the opposite case. After highlighting them, the user found that they were actually the two top true positives in the *Found Patterns View* (the first two that are selected as positive labels, as indicated by the green icon )! The user clicked on a candidate close to the two true positives / green points in the dimensionality reduction scatter plot (near the top right corner). As expected, the user found false negatives around them, such as the one highlighted in the figure. This unexpected finding implies that the variance of the patterns may be greater between measurements than within the same measurement. Namely, the query originated from File 1, the target patterns in File 1 were similar to it, and the found patterns were relatively decent in File 1; the target patterns in File 2 deviated comparably more from the query, and the retrieval quality suffered. Interestingly, though the two true positives were far from the query in the dimensionality reduction scatter plot, they were close to the query in the force graph, showing the superiority of the force graph over the dimensionality reduction scatter plot in assessing the relevance of the found patterns and in finding false positives, at least in this use case. We labeled the true positives as green positive labels and the top false positive as a negative label, so that the reader can better identify the issue in the scatter plot and the benefit of the force graph. Namely, in the dimensionality reduction scatter plot, the red point (false positive) is very close to the yellow point (query), and the two green points (true positives) are very far away from the yellow point (in the top right corner); conversely, in the force graph, the green points are close to the yellow point, and the red point is farther from the yellow point than from the two green points, which is desirable.

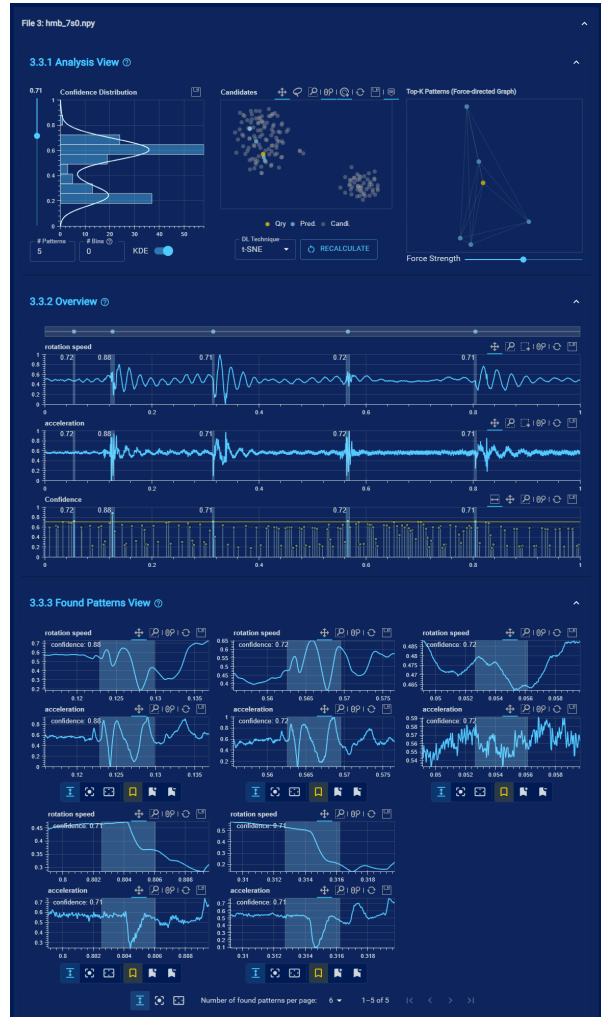


Fig.S46: **Initial Search Result (File 3).** The user proceeded with the third measurement file. There was nothing special in this file. The first two found patterns in the *Found Patterns View* were relevant, which the user also verified in the *Overview* by checking their context. The other found patterns were irrelevant. There were no other false negatives discovered with the help of the dimensionality reduction scatter plot. Again, the confidence histogram helped in this case. Namely, the user adjusted the confidence threshold right below the first tiny cluster with top confidence (exactly the position in the vertical slider), which left only the true positives.



Fig.S47: Add Labels. (Caption on the next page)

Fig.S47: **Add Labels.** After result analysis, the user decided to conduct the feedback-driven search, mainly in order to improve the result in the second data file with many false negatives. The user clicked on the second button or the third button in the button group  below a found pattern in the *Found Patterns View* to mark it as a positive label or a negative label. The user would also like to mark a couple of spontaneous negative labels directly in the *Overview*. This would help the model better recognize the background. The user clicked on the Bokeh box select tool  and dragged a box between the first two positive labels in the *Overview* to add a long negative label. Thanks to nonmyopic search, such long labels become possible during feedback-driven pattern search in time series. The uncommitted label looked like the long right negative label, which was filled with transparent red. The user could adjust its position and size. After clicking the “ADD” button, the label was committed and looked like the long negative label on the left with only a frame and no fill color. The labels from found patterns are also visible in the dimensionality reduction scatter plot and the force graph.

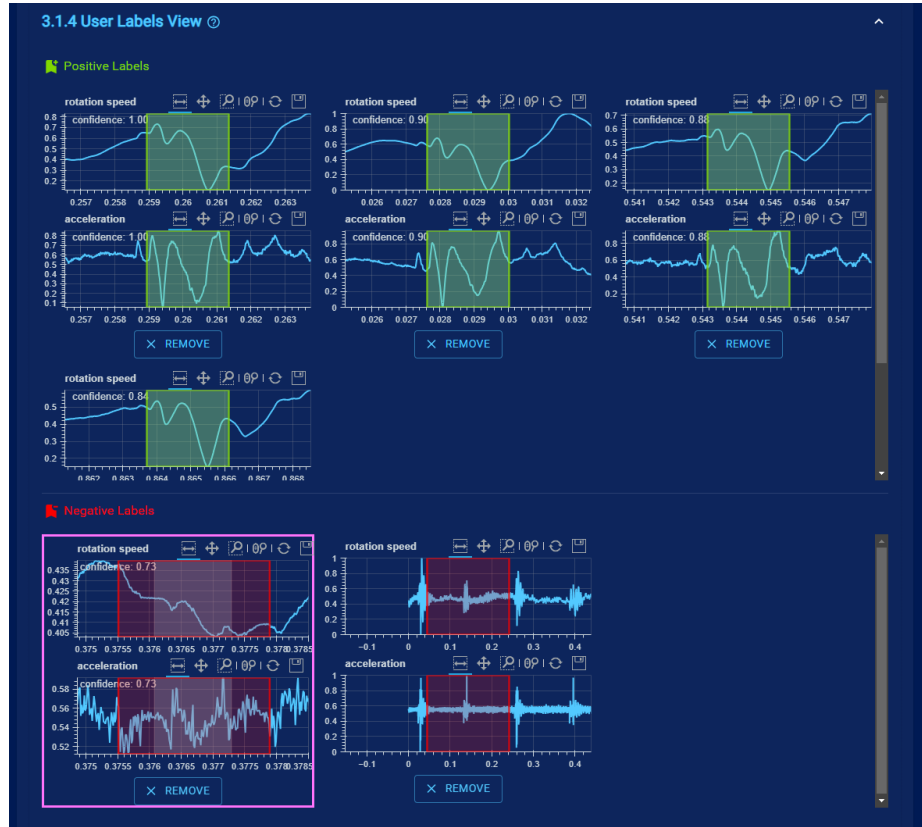


Fig. S48: **Review Labels (File 1)**. The user provided feedback in each file. After that, the user reviewed the labels before starting the feedback-driven search. In the first file, the user marked the labels shown in this figure. Note that highlighting (the pink frame) also works in this view. The second negative label was spontaneously added in the *Overview* in Fig. S47. Other labels were originated from found patterns in the *Found Patterns View*. The line charts show the found patterns as blue boxes from which the labels originate. This is best seen in the line chart of the first negative label, where the negative label is larger than the originating found pattern. The user can adjust the positions and sizes of the labels, as we have unnecessarily done for the first negative label, just for illustration. The user can also remove labels by clicking the “REMOVE” button.



Fig. S49: **Review Labels (File 2)**. The user provided the labels shown in the figure for the second data file. To show the originating found patterns, we have included the *Found Patterns View* in the screenshot. Basically, the user marked the two true positives as positive labels and the top false positive as a negative label.

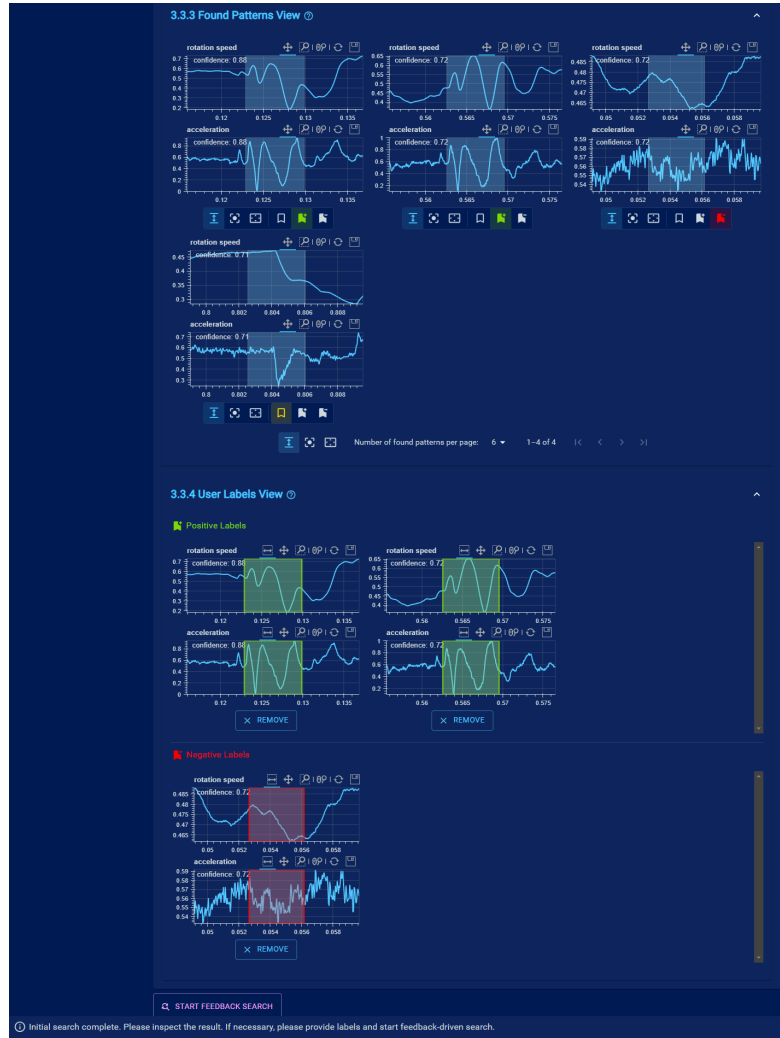


Fig. S50: **Review Labels (File 3)**. The user provided the labels shown in the figure for the third data file. As in Fig. S49, the user marked the two true positives as two positive labels and the top false positive as a negative label. At this point, the user had inspected the results in all data files and was ready to start the feedback-driven search. To proceed, the user clicked the button “START FEEDBACK SEARCH”.

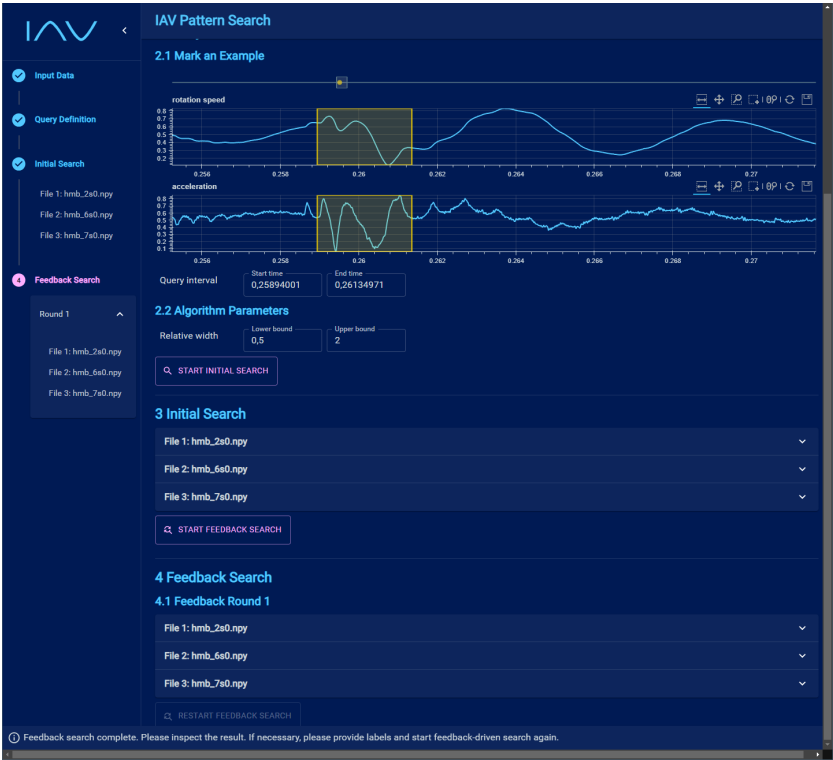


Fig.S51: **Feedback-Driven Search Result Ready.** Upon completion of the feedback-driven search, the spinner in the button “START FEEDBACK SEARCH” stopped. The user saw three collapsed areas for the results in the three data files after feedback-driven search.

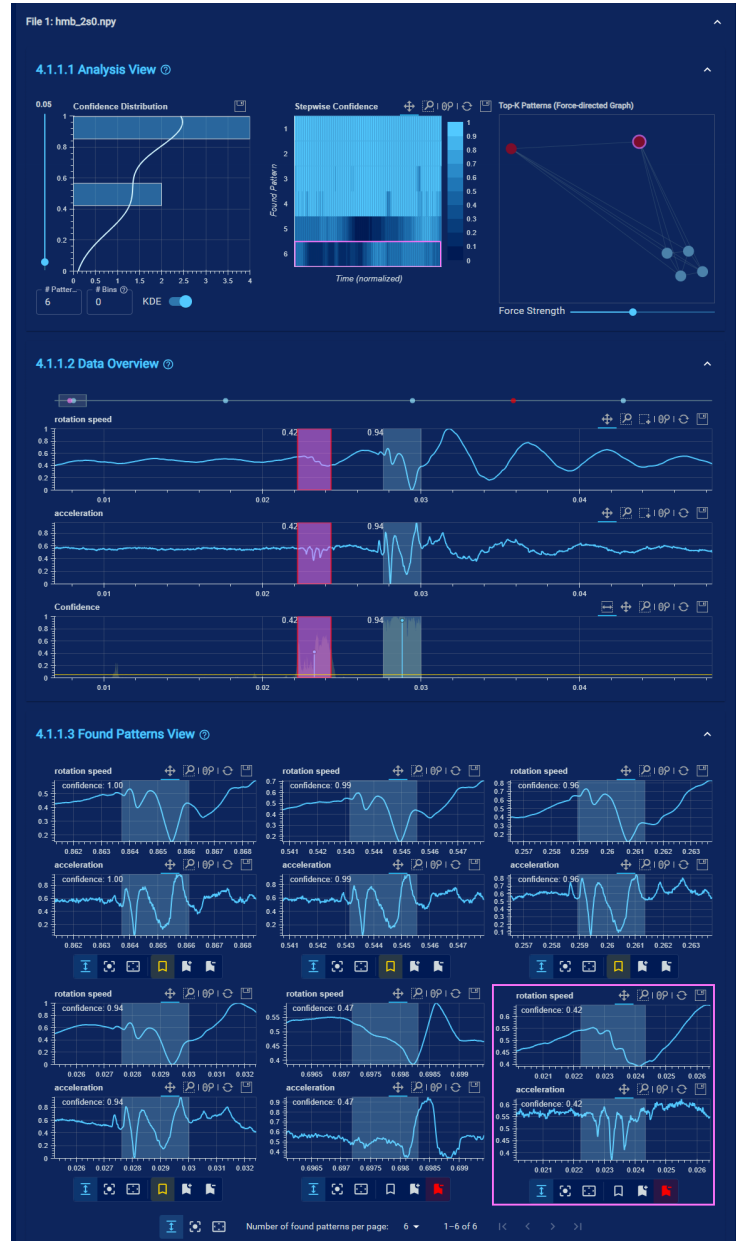


Fig. S52: Feedback-Driven Search Result (File 1). (Caption on the next page)

Fig.S52: Feedback-Driven Search Result (File 1). The user opened the result in the first data file. Without surprise, the user again met the four positive labels in the *Found Patterns View*. The fifth found pattern was clearly a false positive. The sixth resembled the target pattern. However, its value range did not match that of the query. The user verified it by clicking  to scale the pattern to match the value range of the query. Furthermore, the user highlighted the pattern and checked its context in the *Overview* (erroneously titled “Data Overview” in the plot). There was exactly one true positive next to the pattern to inspect. It was apparent that the highlighted one belonged to the background. Otherwise, the force graph suggested the same conclusion. The user did not see the dimensionality reduction scatter plot in the feedback-driven search because there are usually not many pattern candidates after the feedback-driven search, and dimensionality reduction does not work well when the number of instances is small. Instead, the user saw a heatmap showing the confidence of each time point in each found pattern. The rows / found patterns are sorted in the same order as in the *Found Patterns View* (descending order of confidence). The time points in the last two rows / found patterns in the heatmap were relatively dark, in contrast to the first four, indicating that the model was uncertain. Indeed, they are the two false positives (last two found patterns) clearly shown in the *Found Patterns View*. In the confidence histogram, the user saw two bars. The upper bar had very high confidence and was well separated from the lower. Therefore, the user could raise the confidence threshold between the two bars or simply reduce the number of found patterns by two. It seemed that the four true positives found by the rule-based initial search were likely the only four target patterns in this data file.

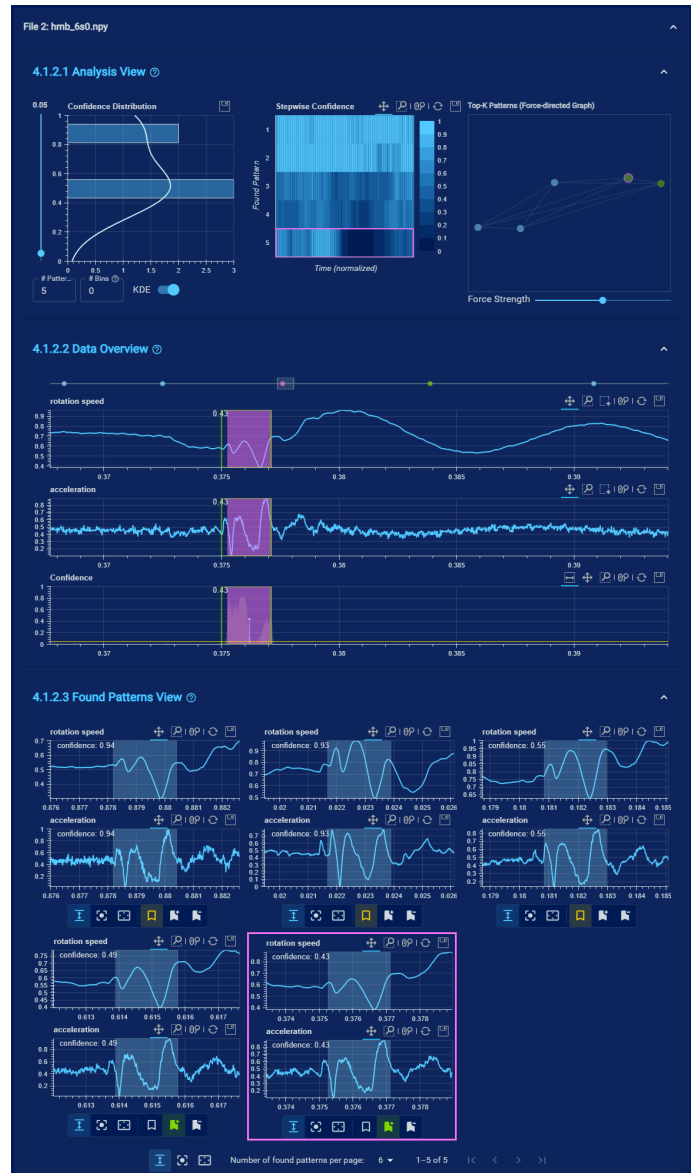


Fig. S53: Feedback-Driven Search Result (File 2). (Caption on the next page)

Fig. S53: **Feedback-Driven Search Result (File 2)**. With high expectation, the user expanded the result for the second file because there were many false negatives in this file in the rule-based initial search. To the user's satisfaction, the feedback-driven search returned three new patterns besides the old two, as shown in the *Found Patterns View*. The new found patterns were indeed relevant patterns after the user's examination in the *Overview*. The user noticed that the confidence of the last found pattern had a trough on the right side of the pattern in the heatmap, showing that the model was uncertain in this region of the found pattern. The user double-clicked on the row / pattern in the heatmap to highlight this found pattern in all views. The third added channel in the *Overview* showed the time point confidence. This low confidence region in this found pattern indicated a classification problem (rather than a clustering problem), which could be solved with a second feedback-driven search with this found pattern as a positive label. The user also noticed that the last two patterns in the *Overview* shrank slightly on the left side. If another feedback-driven search was intended, this imprecision could be corrected with multi-feature feedback in the *User Labels View*. However, depending on the domain requirements, it was unnecessary to proceed with another feedback-driven search round just for this minor imperfection.

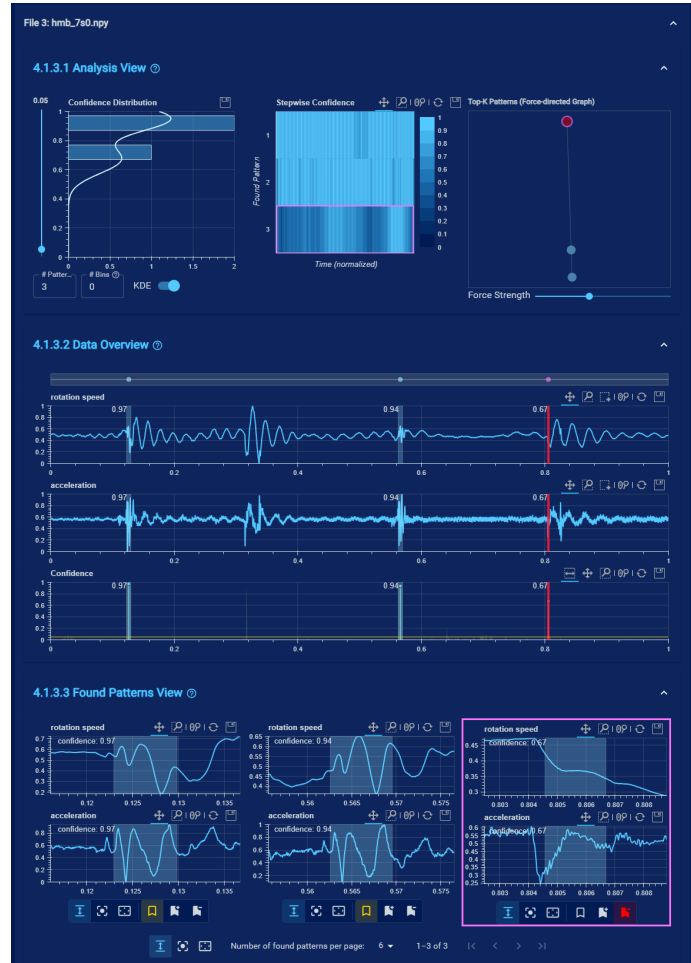


Fig. S54: **Feedback-Driven Search Result (File 3)**. The user went on to the third data file. Its situation was similar to the first. The first two found patterns were the positive labels after the rule-based initial search. The third found pattern was obviously a false positive, which was also implied in the force graph. Considering that the time points in the found pattern have high confidence, as shown in the heatmap, the user could mark it as a negative label to correct the misclassification in the next feedback round. For the result in this file per se, the user could simply raise the confidence threshold to between the two bars or reduce the number of found patterns by one. There were actually no other target patterns in the data file. The user was satisfied with the results and proceeded with subsequent domain-specific analysis.

L Expert Study

L.1 Procedure

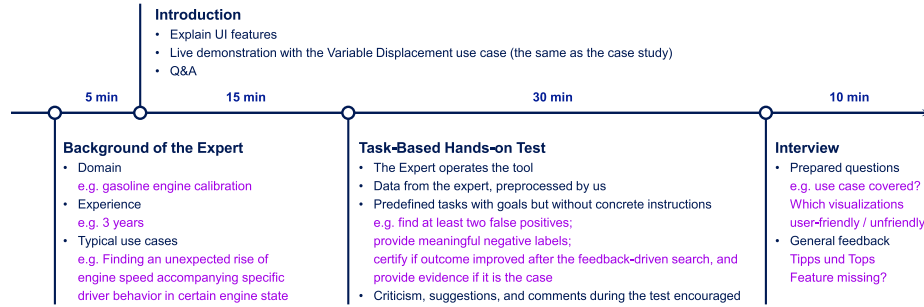


Fig. S55: Procedure for the Expert Study.

In this appendix, we elaborate on the expert study.

Fig. S55 shows the procedure for the expert study. We invited two experts, each to a one-hour session. It began with an inquiry about the expert's domain, experience, and typical use cases. Then we explained the major features of NOOPS, followed by a live demonstration with the **Variable Displacement** use case used in the case study in App. K. After we clarified all the questions from the expert, the expert took full control of the laptop and received tasks one after another from us. Both experts used the PHEV dataset described in App. G.5. Both experts are very familiar with the dataset and the use case. For the expert study, we only used the first three data files that the expert provided us. Before the experts took over, we had already started the web application and typed the paths to the data and metadata into the corresponding input fields. The tasks that the experts needed to work through one by one were

Tasks Guiding the Hands-on Test

- T1: Inspect the data and find at least one example of target patterns in the data.
- T2: Specify a query (the pattern to search for) and start the initial search.
- T3: Inspect the found patterns of the first file and point out three true positives (relevant found patterns), or all true positives if there are fewer than three.
- T4: Search for false negatives (missing target patterns) with the help of the dimensionality reduction scatter plot. Notify us when you believe that all potential false negatives suggested by the tool have been examined.
- T5: Find at least one false positive (irrelevant found pattern) or assert that there are no false positives in the file.
- T6: Answer the question: is it possible to filter out only the true positives? If yes, please demonstrate; if no, please argue.
- T7: Inspect the results in the other two files.
- T8: Provide feedback on the results. At least three positive labels and one negative label are desired. Please do not forget the possibility of marking spontaneous labels.
- T9: Conduct the feedback-driven search and verify whether the results are improved.
- T10: Point out an imperfect found pattern (the found pattern is relevant, but its position or size is imprecise), if any, and suggest the remedy.

All communication was in German because our experts were not completely comfortable with English. We thereby translate the protocol into English. During the design of the tasks, we try to convey the goals and avoid concrete instructions or means to test the usability of the UI. During the 30-minute task-based hands-on test, the experts were encouraged to provide feedback whenever they deemed necessary.

Finally, an interview after the test rounded off the expert study. The following questions were posed during the interview.

Questions During Interview after the Hands-on Test

- Q1: Does the tool cover your use case?
- Q2: Could you imagine other scenarios beyond this use case that you have provided where the tool may be of help?
- Q3: Which visualizations and features are the most useful based on your experience?
- Q4: Which visualizations and features are user-unfriendly based on your experience?
- Q5: Do you miss a feature? If so, please elaborate.
- Q6: Other criticisms, suggestions, and comments.

L.2 Protocol

We conducted the expert study with two experts individually. Expert A is a senior technical consultant with 15 years of experience in the development and calibration of gasoline engines. Expert B is an on-board diagnostics engineer with one year of experience in air, fuel, and thermal-related features of gasoline engines. We recorded thought-provoking remarks and behavior of the experts. The protocols of the two experts are merged for a compact presentation. After reporting on the observations, we provide our analysis of the issues.

- T1: Inspect the data and find at least one example of the target patterns in the data.
No interesting behavior was observed.
- T2: Specify a query (the pattern to search for) and start the initial search.
Expert B pointed out that in some cases, the target pattern could be as simple as binary bit sequences. For instance, he experienced such signal fragments of interest in CAN bus data. In such cases, it would be simpler to specify the query by sketching. Although he would opt for query-by-example if he must choose one as a permanent solution, he argued that the two query definition approaches were not mutually exclusive.
- T3: Inspect the found patterns of the first file and point out three true positives (relevant found patterns), or all true positives if there are fewer than three. The experts browsed all found patterns in the *Found Patterns View* and labeled the true positives and false positives simultaneously in one go. Expert A suggested that the query could be placed next to the found patterns for an easier comparison. Expert B suggested that it would be handy to list the found patterns in all data files in a summary view, rather than listing them in each data file separately. Because the number of files could easily reach hundreds or even thousands in his use cases. He added that the order of the views could be optimized. He would prefer the *Found Patterns View* at the top, followed by the *Overview*, then the *Analysis View*, and finally the *Feedback View*, which better respected his workflow. Moreover, he would not compare the found patterns with each other, so it was unnecessary to

show multiple found patterns simultaneously in the *Found Patterns View*. It did not matter to list all found patterns in a table, where each row represented a found pattern with some meta-information. After clicking a row, the corresponding found pattern would be shown in a line chart. Whereas, he could not certify that other engineers also do not have the need to see multiple found patterns simultaneously.

Our Reflection: we will offer different ways to present the results according to the engineers' different needs.

- T4: Search for false negatives (missing target patterns) with the help of the dimensionality reduction scatter plot. Notify us when you believe that all potential false negatives suggested by the tool are examined.

No interesting behavior was observed. Both experts had never heard of dimensionality reduction. We explained the meaning, purpose, and desired interactions in the Q&A session after the live demonstration. Expert A was very amazed that such a technique existed. During the hands-on test, both experts had no problem exploiting the dimensionality reduction scatter plot in search of the false negatives, though they did express bitter experience regarding the steep learning curve. Expert A mentioned that it would be critical to embed our explanation during and after the live demo, somehow in the app itself. Otherwise, he doubted future users without seeing our live demo and hearing our explanation could understand the visualization and know what to do with it. He appreciated the tooltips and hints in the bottom status bar. Nonetheless, he wished for something like another sidebar or bottom pop-up with video tutorials or multiple pictures explaining the visualization. Expert B expressed a similar opinion. He admitted that there were currently no ways to find false negatives, and that the need was valid. In that sense, the dimensionality reduction scatter plot did address his need. However, it took quite some effort for him to fully comprehend dimensionality reduction. He expressed frustration and stressed that this could pose a usability issue. Both of them also expressed similar opinions about the force graph and the heatmap. Moreover, Expert B suggested adding contours or color gradients in the background of the dimensionality reduction plot, so that it would be clearer which points / pattern candidates were closer to the query and other true positives. Additionally, Expert B pointed out that it would be a good practice to have axis titles all the time. Even in such cases, where the axes did not have concrete physical meanings, they could still be labeled somewhat like "Primary Embedding Dimension" and "Secondary Embedding Dimension".

Our Reflection: we realized that even for such standard techniques like the dimensionality reduction scatter plot in visual analytics, domain users without a data science background may have problems applying it. According to our previous experience, domain experts are rather reluctant to learn data science algorithms and wish for a magic tool, which, fed with data, would spew their desired outcome. In case of the scatter plot, they need to know at least that a point in the scatter plot represents a pattern candidate, and similar patterns / points are closer. We should not assume that the user

will bring this knowledge with them. Rather, we need to explicitly convey the knowledge to the user in the UI. However, they may still be unwilling to assimilate this information. Thus, we could go one step further and try to algorithmically and automatically find the potential false negatives based on the dimensionality reduction and find the potential false positives based on the force graph. Namely, we can investigate algorithms that analyze scatter plots and force graphs automatically. In the future, we can show the potential false positives and negatives directly as line charts to the user for interactions and feedback, so that the domain user does not need to learn the techniques and the visualizations.

- T5: Find at least one false positive (irrelevant found pattern) or assert that there are no false positives in the file.

Both experts accomplished this task during T3.

Our Reflection: we expected the experts to find the false positives with the help of the force graph. However, it seems that it is more straightforward for them to differentiate true positives and false positives by looking at the line charts in the *Found Patterns View* directly. In that sense, we failed to validate the usefulness of the force graph with the expert study. Nonetheless, the case study verified its benefit to some extent.

- T6: Answer the question: is it possible to filter out only the true positives. If yes, please demonstrate; if no, please argue.

Both experts were able to deduce that if all true positives have higher confidence than the false positives, it is possible to separate them by tuning the confidence threshold or tuning the number of found patterns. Otherwise, if there were false positives with higher confidence than at least one true positive, a perfect separation would be impossible. When asked for a remedy, both came to the right answer, namely feedback-driven search. Irrelevant to the question, Expert B praised the added channel in the *Overview* with confidence of all pattern candidates and a draggable horizontal line for adjusting the confidence threshold. He preferred this added channel to the confidence distribution histogram. Even though we explained that the histogram offered a unique way to find the appropriate confidence threshold, which would be just below the tiny cluster with top confidence, he argued that ideally, the prominent found patterns stood out in the added channel, while the irrelevant candidates had lower bars, which also indicated the appropriate confidence threshold. That said, he preferred a global confidence threshold, not one for each data file. Therefore, the histogram could still be a better solution because the added channel worked only within each data file.

- T7: Inspect the results in the other two files freely.

One of the files had too many false negative candidates in the dimensionality reduction scatter plot (there were many candidates closer to the query than the found patterns). Expert A had no patience to check all of them. He had forgotten to use the lasso selection tool to select all false negative candidates. The confidence threshold could be adjusted to include all of them as found patterns. However, even he used the lasso selection tool after our reminder,

the result was unsatisfactory because many false positives also appeared. He checked some potential false negatives and decided to give up.

Our Reflection: this was rather a frustrating experience during the hands-on test. The idea in T4 to show the line charts of the potential false negatives and false positives could potentially alleviate the issue because the user does not need to click the points before checking the line charts, which better caters to the domain user’s impatience.

- T8: Provide feedback on the results. At least three positive labels and one negative label are desired. Please do not forget the possibility of marking spontaneous labels.

Expert A mentioned that the feature of marking spontaneous labels was hidden and somewhat unintuitive. He would like to have the input fields and buttons below the *Overview*, which appeared when marking a spontaneous label, to float above the labels in the *Overview* and close to the user’s focus, rather than below the line chart in the *Overview*. This was especially problematic when there were many channels, as was exactly the case because there were three channels in the use case.

Our Reflection: interactively (i.e., on an ad hoc basis) added UI components, which require immediate user interactions, should ideally be close to the user’s current focus or the position of the mouse cursor. It is our mistake to violate this rule.

- T9: Conduct the feedback-driven search and verify whether the results are improved.

No interesting behavior was observed.

- T10: Point out an imperfect found pattern (the found pattern is relevant, but its position or size is imprecise), if any, and suggest the remedy.

No interesting behavior was observed.

Our Reflection: actually, we did not expect the last task to be solved so effortlessly. On the contrary, we expected this task to be the most demanding as it required understanding of the algorithm. However, perhaps we provided a telling example during the live demonstration, and this was also the last point we demonstrated in the live demo. Both experts had fresh memories and could simply imitate our analysis.

Hereby, we also attach the protocol in the ten-minute interview after the hands-on test.

- Q1: Does the tool cover your use case?

Both affirmative, definitely.

- Q2: Could you imagine other scenarios beyond this use case that you have provided where the tool may be of help?

Expert A: not exactly other scenarios, but we often want to collect multiple samples of a certain event in the data to analyze its cause and effect. Such an event is usually an unexpected behavior of the engine. It is rather spontaneous. I cannot say what we search constantly, but we search for different events. In the old days, we would search for such events in historical

measurements. We have even developed a tool called “Rampensau” (German word meaning passionate stage performer) to detect ramps (“Rampen” in German), which mathematically describes a specific query and can only search for this specific pattern. If we cannot retrieve such events from the historical data, we must commission new measurements, which would incur costs. We appreciate it that you saved us hassle and money.

Expert B: There are cases where the appearance of the target patterns is initially obscure to us. We may start with another anomaly detection tool, also from your team. Then, we can search for similar events with this pattern search tool. That said, how about a bigger tool combining the two, maybe a platform tool where we can concatenate these atomic tools and build our own analysis pipeline?

Q3: Which visualizations and features are the most useful based on your experience?

Expert A: from my perspective, the one-click solution to scale all found patterns to the value range of the query is the most useful one. I would even suggest setting it as the default to show all found patterns in the same value range as the query, because it does not make sense to check the details of a found pattern, if it does not have the correct value range anyway.

Expert B: I like the last added channel in the *Overview* with the confidence of the pattern candidates and a draggable horizontal line for adjusting the confidence threshold.

Our comment: the practical features that are easy to use and best suit the task are rated the highest.

Q4: Which visualizations and features are user-unfriendly based on your experience?

Expert A: as mentioned earlier, I find the visualizations except the confidence histogram, though undoubtedly useful, not immediately accessible for me as a mechanical engineer. In terms of research, I understand the different circumstances. However, as a product, being simple to use is of paramount importance. In my view, Google search is so successful because there is only one input field. But this example is perhaps excessive, as Google search is designed for everyone, while experts deserve more complex tools for deeper insights. If it is indeed helpful, as I can say in this case, we can organize trainings.

Expert B: I find the visualizations in the *Analysis Views* useful but require too much learning.

Our comment: as mentioned, especially in the reflection for T4, much to our surprise, even standard techniques in visual analytics can be intimidating for domain users. We came to realize the frustrating reason why many delicate visual systems thrive in the greenhouses of laboratories but fail to survive in the wildness of industry.

Q5: Do you miss a feature? If so, please elaborate.

Expert A: No, I cannot think of any.

Expert B: It is common for calibration engineers to add or remove channels during inspection. It might be a good idea to allow adding channels after the

initial search and before the feedback-driven search. Because even though target events may reveal themselves visually as patterns in certain channels, other channels may be related according to domain knowledge and may contribute to the retrieval of such events, which is possible with the machine learning techniques behind the feedback-driven search.

Q6: Other criticisms, suggestions, and comments.

Expert A: I'm generally pleased with everything I have seen. I look forward to closer cooperation between our teams, because I believe that our use cases and requirements inspire and drive your work, and your research and development simplify and expedite our tedious routines.

Expert B: Not really. Generally, the UI looks beautiful and professional.

L.3 Screenshots

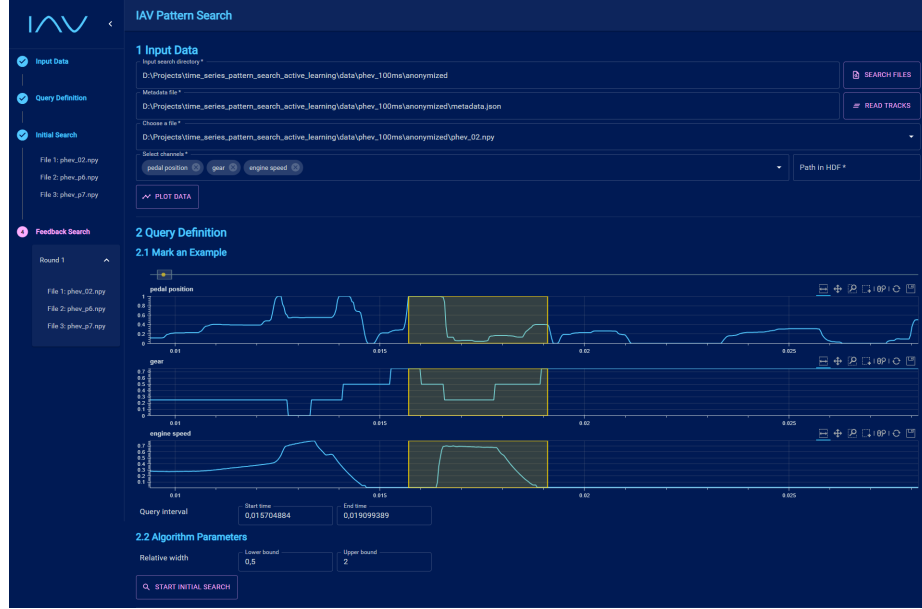


Fig. S56: **Query.** The physical background of the data and the query is described in detail in App. G.5. In a nutshell, the car is driven by the electric motor at the beginning of the event. The driver then depresses the gas pedal (first channel for the gas pedal position). In reaction, the car switches to a lower gear to produce higher acceleration (second channel for the current gear). Simultaneously, the combustion engine is activated and suddenly revolves unexpectedly fast (third channel for the engine speed). After that, the state of the transmission and the engine resume. This use case is much more formidable than the **Variable Displacement** use case in the case study in App. K because the desired features in the target patterns are much more subtle, with many conditions and constraints that are specified only implicitly. For instance, at the beginning of the event, the combustion engine should be idle or operating at low speed, and the gas pedal should not be depressed to a large extent. The increase in the position of the gas pedal and the engine speed should be significant. Additionally, causality must be guaranteed. Namely, the abrupt change in pedal position should occur first. The query does not contain the abrupt increase of the pedal position at the beginning of the event. This query was marked by the expert on purpose, knowing that the abrupt increase of the pedal position would still be taken into account because nonmyopic search is aware of the contexts of the patterns.

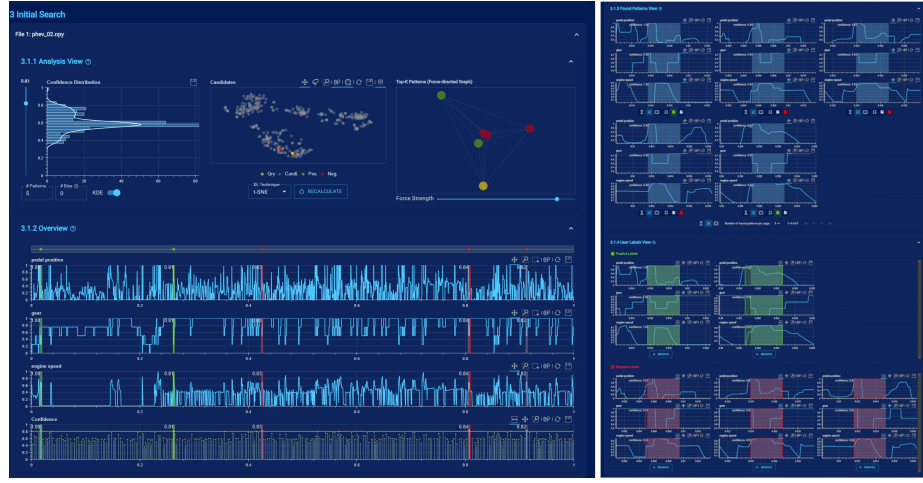


Fig.S57: **Initial Search Result (File 1).** The initial search result in File 1 was not particularly good, as the rule-based initial search only retrieved the query and one more true positive in the top-5 found patterns with the highest confidence. The true positives and false positives were marked as positive labels and negative labels, respectively. Worse still, the confidence distribution histogram and the force graph did not seem to contribute much to the analysis. The dimensionality reduction scatter plot helped, albeit with many false candidates suggested for examination. The result could be improved with the feedback-driven search because, on the one hand, there were three false negatives (marked as false negatives) with higher confidence than a true positive (marked as positive label). Therefore, it was impossible to filter out all true positives without introducing many false positives. On the other hand, it was very likely that the data contained false negatives that the feedback-driven search might be able to discover.

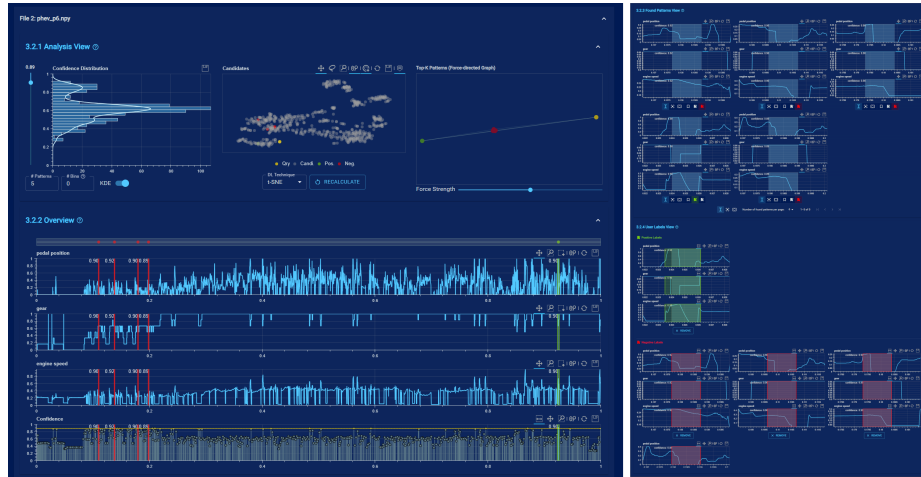


Fig. S58: **Initial Search Result (File 2)**. The initial search result in File 2 was even gloomier. The rule-based initial search found one true positive in the top-5 found patterns. Other found patterns were false positives. It was also interesting that the true positive was farther to the query than the false positives in the force graph, indicating the erroneous usage of ED as the distance measure between two patterns in the force graph (potentially also inappropriate for the dimensionality reduction scatter plot).



Fig. S59: **Initial Search Result (File 3)**. The result in File 3 was also far from ideal, as the initial search found only one true positive and four false negatives with higher confidence. Considering the results in all files, it could be concluded that the initial search did not perform well, and the feedback-driven search should be resorted to. Nonetheless, the rule-based initial search did retrieve some true positives that could serve as positive labels.

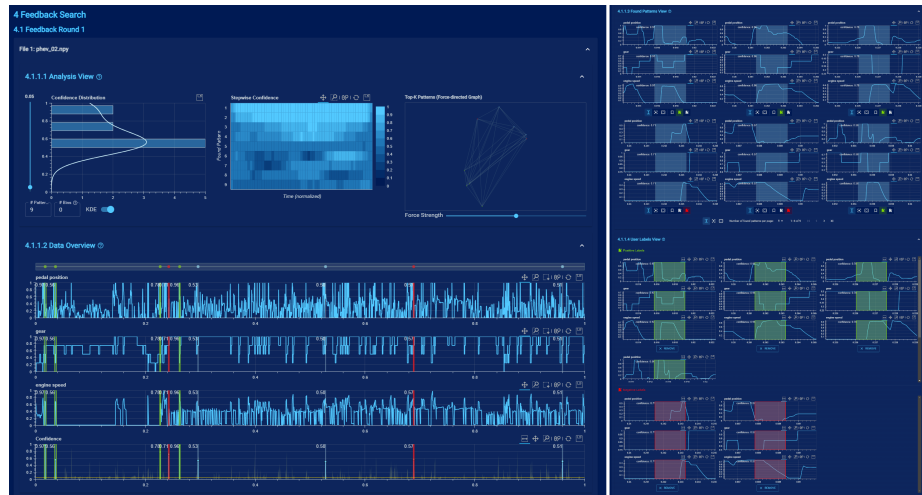


Fig. S60: Feedback-Driven Search Result (File 1). A major improvement was witnessed as two more true positives were found compared to the result from the rule-based initial search. If zoomed-in substantially, the four found patterns / green points built a cluster at the bottom of the force graph and are far from the false positives, an indicator suggesting a good result.

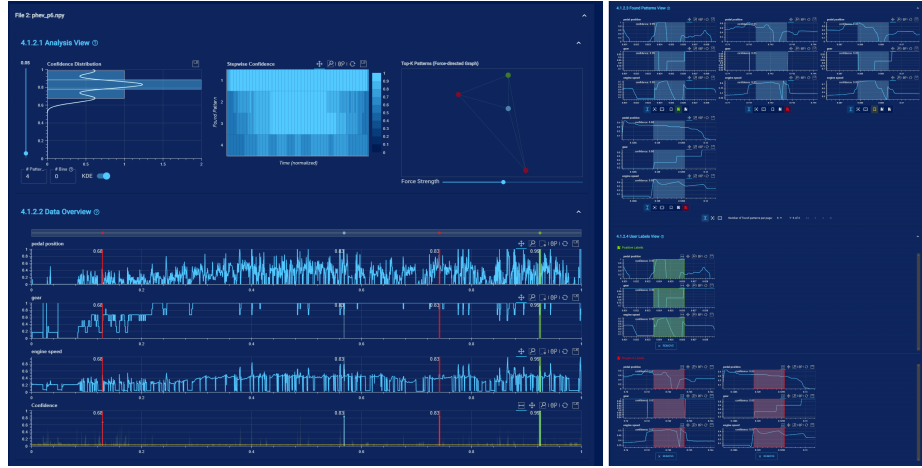


Fig S61: **Feedback-Driven Search Result (File 2)**. The result in File 2 was not noticeably improved. The feedback-driven search found the same true positive, an indecisive pattern, and two false negatives. A minor point, the true positive had a more precise starting and end time than the one after the rule-based initial search, thanks to the multi-feature feedback (the position and size of the found pattern was corrected before used as a positive label during model training).

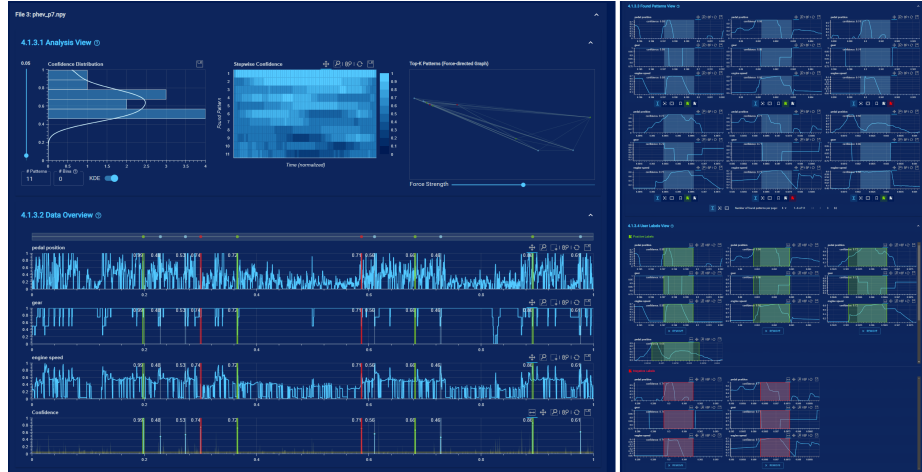


Fig. S62: **Feedback-Driven Search Result (File 3)**. A substantial improvement was observed as three more true positives were discovered.

References

1. Buono, P., Aris, A., Plaisant, C., Khella, A., Shneiderman, B.: Interactive Pattern Search in Time Series. *Visualization and Data Analysis 2005* **5669**, 175–186 (2005). <https://doi.org/10.1117/12.587537>
2. Fan, C., Matković, K., Hauser, H.: Sketch-Based Fast and Accurate Querying of Time Series Using Parameter-Sharing LSTM Networks. *IEEE Transactions on Visualization and Computer Graphics* **27**(12), 4495–4506 (2021). <https://doi.org/10.1109/TVCG.2020.3002950>
3. Folgado, D., Barandas, M., Antunes, M., Nunes, M.L., Liu, H., Hartmann, Y., Schultz, T., Gamboa, H.: TSSEARCH: Time Series Subsequence Search Library. *SoftwareX* **18** (2022). <https://doi.org/10.1016/j.softx.2022.101049>
4. Giao, B.C., Anh, D.T.: 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). <https://doi.org/10.1007/s40595-016-0062-4>
5. Hochheiser, H., Shneiderman, B.: Dynamic Query Tools for Time Series Data Sets: Timebox Widgets for Interactive Exploration. *Information Visualization* **3**(1), 1–18 (2004). <https://doi.org/10.1057/palgrave.ivs.9500061>
6. Imani, S., Alaei, S., Keogh, E.: Qute: Query by Text Search for Time Series Data. In: Arai, K., Kapoor, S., Bhatia, R. (eds.) *Proceedings of the Future Technologies Conference*, pp. 412–427. Springer, Cham (2021)
7. Keogh, E.J., Pazzani, M.J.: An Enhanced Representation of Time Series Which Allows Fast and Accurate Classification, Clustering and Relevance Feedback. In: *The 4th International Conference on Knowledge Discovery and Data Mining*, pp. 239–243 (1998)
8. Lee, D.J.-L., Lee, J., Siddiqui, T., Kim, J., Karahalios, K., Parameswaran, A.: You Can’t Always Sketch What You Want: Understanding Sensemaking in Visual Query Systems. *IEEE Transactions on Visualization and Computer Graphics* **26**(1), 1267–1277 (2020). <https://doi.org/10.1109/TVCG.2019.2934666>
9. Lekschas, F., Peterson, B., Haehn, D., Ma, E., Gehlenborg, N., Pfister, H.: PEAX: Interactive Visual Pattern Search in Sequential Data Using Unsupervised Deep Representation Learning. *Computer Graphics Forum* **39**(3), 167–179 (2020). <https://doi.org/10.1111/cgf.13971>
10. Mannino, M., Abouzied, A.: Expressive Time Series Querying with Hand-Drawn Scale-Free Sketches. In: *Proceedings of the CHI Conference on Human Factors in Computing Systems*, pp. 1–13. ACM, NY, USA (2018). <https://doi.org/10.1145/3173574.3173962>
11. Michael, Y.C.-C., Yan, Z., Liudmila, U., Nurjahan, B., Yifei, D., Anh, D.H., Furtado, S.D., Abdullah, M., Eamonn, K.: 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*, pp. 1317–1322 (2016). <https://doi.org/10.1109/ICDM.2016.0179>
12. Ren, P., Xiao, Y., Chang, X., Huang, P.-Y., Li, Z., Gupta, B.B., Chen, X., Wang, X.: A Survey of Deep Active Learning. *ACM Computing Surveys* **54**(9) (2021). <https://doi.org/10.1145/3472291>
13. Rodrigues, J., Folgado, D., Belo, D., Gamboa, H.: SSTS: A Syntactic Tool for Pattern Search on Time Series. *Information Processing & Management* **56**(1), 61–76 (2019). <https://doi.org/10.1016/j.ipm.2018.09.001>

14. Ryall, K., Lesh, N., Lanning, T., Leigh, D., Miyashita, H., Makino, S.: QueryLines: Approximate Query for Visual Browsing. In: CHI Extended Abstracts on Human Factors in Computing Systems, pp. 1765–1768. ACM, NY, USA (2005). <https://doi.org/10.1145/1056808.1057017>
15. Schlegl, T., Schlegl, S., Tomaselli, D., West, N., Deuse, J.: Adaptive Similarity Search for the Retrieval of Rare Events from Large Time Series Databases. *Advanced Engineering Informatics* **52**, 101629 (2022). <https://doi.org/10.1016/j.aei.2022.101629>
16. Siddiqui, T., Luh, P., Wang, Z., Karahalios, K., Parameswaran, A.: ShapeSearch: A Flexible and Efficient System for Shape-Based Exploration of Trendlines. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, pp. 51–65. ACM, NY, USA (2020). <https://doi.org/10.1145/3318464.3389722>
17. van de Schoot, R., de Bruin, J., Schram, R., Zahedi, P., de Boer, J., Weijdem, F., Kramer, B., Huijts, M., Hoogerwerf, M., Ferdinands, G., et al., An Open Source Machine Learning Framework for Efficient and Transparent Systematic Reviews. *Nature Machine Intelligence* **3**(2), 125–133 (2021). <https://doi.org/10.1038/s42256-020-00287-7>
18. Yu, Y., Becker, T., Trinh, L.M., Behrisch, M.: SAXRegEx: Multivariate Time Series Pattern Search with Symbolic Representation, Regular Expression, and Query Expansion. *Computers & Graphics* **112**, 13–21 (2023). <https://doi.org/10.1016/j.cag.2023.03.002>
19. Yu, Y., Kruffy, D., Jiao, J., Becker, T., Behrisch, M.: PSEUDO: Interactive Pattern Search in Multivariate Time Series with Locality-Sensitive Hashing and Relevance Feedback. *IEEE Transactions on Visualization and Computer Graphics* **29**(1), 33–42 (2022). <https://doi.org/10.1109/TVCG.2022.3209431>
20. Zhang, E., Zhang, Y.: Average Precision. In: Encyclopedia of Database Systems. Ed. by L. LIU and M.T. Özsu, pp. 192–193. Springer US, Boston, MA (2009). https://doi.org/10.1007/978-0-387-39940-9_482