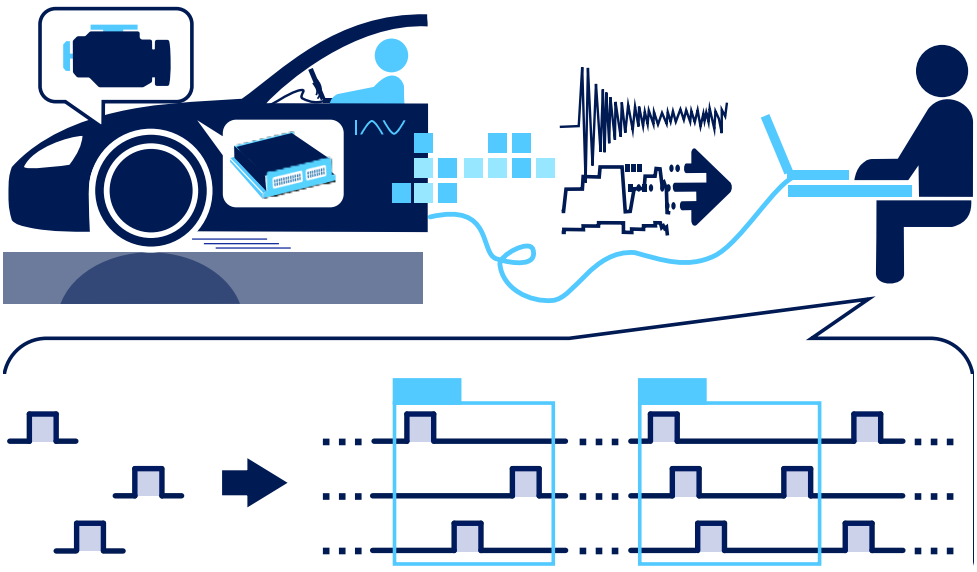
MULTIVARIATE TIME SERIES PATTERN SEARCH



YUNCONG YU

MULTIVARIATE TIME SERIES PATTERN SEARCH

YUNCONG YU

Cover: automotive calibration engineers constantly search for events of interest manifested as fuzzy patterns in sensor measurements in the form of multivariate time series from the engine control unit spontaneously and wish an accurate and prompt retrieval, e.g., for error root diagnosis.

Multivariate Time Series Pattern Search

Yuncong Yu
Ph.D. Thesis

The research for this dissertation was conducted
(1) in the Analytics Platform Team, part of the Department of Data Science and AI Projects, IAV GmbH Ingenieurgesellschaft Auto und Verkehr, Germany and
(2) in the Visualization and Graphics (VIG) group, part of the Department of Information and Computing Sciences (ICS), Faculty of Science, Utrecht University, the Netherlands.

https://doi.org/10.33540/2703

Multivariate Time Series Pattern Search

Zoeken Naar Multivariate Tijdreekspatronen

(met een samenvatting in het Nederlands)

Proefschrift

ter verkrijging van de graad van doctor aan de
Universiteit Utrecht
op gezag van de
rector magnificus, prof.dr. H.R.B.M. Kummeling,
ingevolge het besluit van het college voor promoties
in het openbaar te verdedigen op

dinsdag 21 januari 2025 des ochtends te 10.15 uur

door

Yuncong Yu

geboren op 29 mei 1992
Zhejiang, China

Promotor:
Prof. dr. A. C. Telea

Copromotoren:
Dr. M. Behrisch
Dr. T. Becker

Beoordelingscommissie:
Prof. dr. W. Aigner
Prof. dr. H.L. Hardman
Prof. dr. T. von Landesberger
Prof. dr. A.P.J.M. Siebes
Prof. dr. Y. Velegrakis

ABSTRACT

This Ph.D. thesis focuses on the topic of (visual) pattern search in multivariate time series. On this topic, we developed accurate, efficient, and interpretable algorithms and designed tools for domain users. Traditional methods were designed primarily for univariate time series pattern search with relatively distinctive and unambiguous target patterns. They may not extend naturally to multivariate cases, and their performance may deteriorate significantly in the presence of distortions. If based on machine learning, conventional techniques become inefficient and uninterpretable and the retrieval accuracy may stagnate. Because it is unlikely that a single tool can fit all use cases, we proposed a toolbox of multiple methods, including 1) a scalable, steerable, and interpretable hashing-based representation for pattern search, especially in very high-dimensional time series; 2) an efficient technique capturing various pattern distortions, especially time shifts between tracks; 3) an accuracy-centric model-agnostic machine-learning-based framework that is simultaneously more accurate and more efficient than the prevailing machine-learning-based pattern search framework; and 4) an enhancement of user feedback for active-learning-based feedback-driven pattern search striving for the highest possible retrieval accuracy. All our proposed algorithms and tools work in and some even prefer multivariate cases. Extensive experiments verified the aforementioned benefits regarding accuracy, efficiency, and if necessary the steerability and interoperability of the proposed methods. Moreover, case studies and expert studies validated the usability of the user interfaces accompanying the proposed algorithms. Our tools are helping automotive calibration engineers trace events of interest and enable further domain-specific analysis. They are domain-agnostic and applicable to use cases in other domains.

SAMENVATTING

Dit proefschrift richt zich op het onderwerp (visueel) patroonzoeken in multivariate tijdreeksen. Voor dit onderwerp hebben we nauwkeurige, efficiënte en interpreteerbare algoritmen ontwikkeld en tools ontworpen voor gebruikers in het domein. Traditionele methoden zijn primair ontworpen voor het zoeken naar univariate tijdreekspatronen met relatief onderscheidende en ondubbelzinnige doelpatronen. Het is mogelijk dat ze niet van nature toepasbaar zijn op multivariate gevallen en hun prestaties kunnen aanzienlijk verslechteren in de aanwezigheid van vervormingen. Als ze gebaseerd zijn op machinaal leren, worden conventionele technieken inefficiënt en oninterpreteerbaar en kan de opvraagnauwkeurigheid stagneren. Omdat het onwaarschijnlijk is dat één enkele tool geschikt is voor alle gebruikssituaties, hebben we een gereedschapskist met meerdere methoden voorgesteld, waaronder 1) een schaalbare, stuurbare en interpreteerbare hashing-gebaseerde representatie voor het zoeken naar patronen, vooral in zeer hoog-dimensionale tijdreeksen; 2) een efficiënte techniek voor het vastleggen van verschillende patroonvervormingen, met name tijdsverschuivingen tussen sporen; 3) een nauwkeurigheidsgericht modelagnostisch, op machine-leren gebaseerd raamwerk dat tegelijkertijd nauwkeuriger en efficiënter is dan het gangbare, op machine-leren gebaseerde patroonzoekraamwerk; en 4) een verbetering van gebruikersfeedback voor actief-leren-gebaseerd feedback-gestuurd patroonzoeken dat streeft naar de hoogst mogelijke zoeknauwkeurigheid. Al onze voorgestelde algoritmen en hulpmiddelen werken in multivariate gevallen en sommige hebben zelfs een voorkeur. Uitgebreide experimenten verifieerden de bovengenoemde voordelen met betrekking tot nauwkeurigheid, efficiëntie en indien nodig de stuurbaarheid en interoperabiliteit van de voorgestelde methoden. Bovendien valideerden casestudies en expertstudies de bruikbaarheid van de gebruikersinterfaces bij de voorgestelde algoritmen. Onze tools helpen autokalibratie-ingenieurs bij het traceren van interessante gebeurtenissen en maken verdere domeinspecifieke analyse mogelijk. Ze zijn domein-agnostisch en toepasbaar op use cases in andere domeinen.

PUBLICATIONS

This thesis is a result of the following publications:

PB1 Y. Yu, D. Kruyff, J. Jiao, T. Becker, and M. Behrisch. PSEUDo: Interactive Pattern Search in Multivariate Time Series with Locality-Sensitive Hashing and Relevance Feedback. In: *EuroVis*, 2022. DOI: https://doi.org/10.2312/evp.20221127. (Poster)

PB2 Y. Yu, D. Kruyff, J. Jiao, T. Becker, and M. Behrisch. PSEUDo: Interactive Pattern Search in Multivariate Time Series with Locality-Sensitive Hashing and Relevance Feedback. In: *Visualization in Data Science (VDS) @ the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining and @ IEEE VIS (hybrid)*, 2022. **(Best Paper Award)**

PB3 Y. Yu, D. Kruyff, J. Jiao, T. Becker, and M. Behrisch. PSEUDo: Interactive Pattern Search in Multivariate Time Series with Locality-Sensitive Hashing and Relevance Feedback. In: *IEEE Transactions on Visualization and Computer Graphics (TVCG)* 29 (1), S. 33–42, 2022. DOI: https://doi.org/10.1109/TVCG.2022.3209431.

PB4 Y. Yu, T. Becker, and M. Behrisch. Multivariate Time Series Retrieval with Symbolic Aggregate Approximation, Regular Expression, and Query Expansion. In: *EuroVis Workshop on Visual Analytics (EuroVA)* 2022. DOI: https://doi.org/10.2312/eurova.20221081. **(Best Paper Award)**

PB5 Y. Yu, T. Becker, L. M. Trinh, and M. Behrisch. SAXRegEx: Multivariate Time Series Pattern Search With Symbolic Representation, Regular Expression, and Query Expansion. In: *Computers & Graphics* 112, S. 13–21, 2023. DOI: https://doi.org/10.1016/j.cag.2023.03.002.

PB6 Y. Yu, T. Becker, P. Schichtel, W. Aigner, A. C. Telea, and M. Behrisch. NOOPS: Nonmyopic Multi-Feature-Feedback-Driven Multivariate Time Series Pattern Search. (To be submitted)

PB7 Y. Yu, T. Becker, W. Aigner, A. C. Telea, and M. Behrisch. Time Series Representations: A Survey. (Under preparation)

CONTENTS

LIST OF ABBREVIATIONS

AP Average Precision
A performance metric for measuring object detection results in computer vision.

APCA Adaptive Piecewise Constant Approximation
A time series representation extending PAA with adaptive/variable piece lengths.

AR Autoregressive
A time series model that estimates the current time step value based on a linear combination of previous.

ARCH Autoregressive Conditional Heteroskedasticity
A time series representation for modeling variances (of prediction error) with an AR model of previous errors.

ARIMA Autoregressive Integrated Moving Average
A time series model that replaces time step values with their differences in ARMA models to model trend nonstationarity in time series.

ARMA Autoregressive Moving Average
A time series model that combines AR and MA models.

CBIR Content-Based Image Retrieval
A discipline that studies methods to retrieve images from an image database that are similar to a query image given by the user.

CNN Convolutional Neural Network
A machine learning model featuring a regularized feedforward neural network with convolution kernels.

DBA Dynamic Time Warping Barycenter Averaging
A technique for averaging time series with regard to time shifts between them and is consistent with DTW.

DBSCAN Density-Based Spatial Clustering of Applications with Noise
A clustering algorithm based on the concentration densities of data items distributed in a space.

DFT Discrete Fourier Transform
An integral transform that works as the counterpart of FT for discrete time-limited signals.

DTFT Discrete-Time Fourier Transform
An integral transform that works as the counterpart of FT for discrete infinite-time signals.

DSP Digital Signal Processing
A discipline that studies operations on sampled and quantified signals that can be performed by digital devices like computers.

DTW Dynamic Time Warping
An elastic time series distance measure that tries to align time steps in two time series while calculating the distance score.

DWT Discrete Wavelet Transform
An integral transform that works as the discrete counterpart ofWT for discrete signals.

ECU Engine Control Unit
A device which controls multiple systems of an internal combustion engine in a single unit.

ED Euclidean Distance
A time series distance measure.

ECG Electrocardiogram
A medical graph recording the heart's electrical activity through repeated cardiac cycles.

EEG Electroencephalogram
A medical graph recording electrical activity in the brain.

EMD Empirical Mode Decomposition
An integral transform that decomposes a time series into a set of base / intrinsic mode functions that does not need a prior kernel choice by the user.

FFT Fast Fourier transform
An integral transform for spectrum analysis for time-limited discrete signals that produces the same result as DFT but accelerated.

FT Fourier Transform
An integral transform decomposing a continuous signal to sinusoidal components of different frequencies.

GAN Generative Adversarial Network
a generative machine learning model featuring a generator and a discriminator competing with each other during training, after which the former can be used for synthetic data generation and the latter for data recognition.

GARCH Generalized Autoregressive Conditional Heteroskedasticity
A time series model/representation that extends ARCH models and models time series variances with an ARMA model of the previous errors.

GP Gaussian Process
A time series model where a finite collection of time steps (regarded as random variables) is multivariate normal distributed.

GRU Gated Recurrent Unit
A machine learning model and a variation of RNN that is simpler than LSTM with insignificant performance loss.

HHT Hilbert-Huang Transform
An integral transform that conducts EMD followed by HSA to obtain instantaneous frequencies.

HMM Hidden Markov Model
A model that assumes that the current observation is only determined by the current state, which not directly observable and depends only on the previous state, not on states before the previous state.

HSA Hilbert Spectrum Analysis
An analysis of the amplitude spectrum in the time-frequency domain with the help of the Hilbert transform.

i.i.d independent and identically distributed
A property of a collection of random variables, where the random variables share the same probability distribution and are mutually independent.

IMF Intrinsic Mode Function
One of the multiple functions resulted from Empirical Mode Decomposition (EMD) that satisfies two defining properties and generally represents a simple oscillating mode in the original time series.

IoT Internet of Things
The commutation of devices connected with each other or with the internet.

IoU Intersection over Union
A metric for measuring how much two items overlap.

LLE Locally Linear Embedding
A manifold learning / nonlinear dimensionality reduction technique that prioritizes preserving the local neighborhood among the data items.

LSH Locality-Sensitive Hashing
A hashing algorithm that tries to generate the same hash code for close data items and different hash codes for distant data items.

LSTM Long Short-Term Memory
A machine learning model and a variation of RNN that retains memory longer than plain RNN.

MA Moving-Average
A time series model that estimates the current time step value based on a linear combination of previous estimation errors.

MASS Mueen's Algorithm for Similarity Search
The so far fastest time series similarity search algorithm.

MDS Multidimensional Scaling
A manifold learning / nonlinear dimensionality reduction technique that tries to preserve the pairwise distances as much as possible.

MTS Multivariate Time Series
A time series with more than one tracks.

MVP Minimum Viable Product
A prototype of a product with only core features, e.g., for gathering feedback from early users.

NMS Non-Maximum Suppression
A technique to suppress multiple findings of the same object, mainly in the discipline of computer vision.

NVH Noise, Vibration, and Harshness
A discipline that studies acoustic problems in automobiles.

PAA Piecewise Aggregate Approximation
A time series representation that splits a time series into piece of fixed length and represent each piece with its mean.

PCA Piecewise Constant Approximation
An obsolete name for PAA.

PCA Principal Component Analysis
A projection / linear dimensionality reduction technique that linearly maps data items to a set of bases called principal components that explain the data variances the most.

PIP Perceptually Important Point
A time series representation that represents a time series with the visually salient points, like the major peaks and troughs in the time series curve.

PLR Piecewise Linear Representation
A time series representation that splits a time series into piece of fixed length and represent each piece with a line segment.

PPR Piecewise Polynomial Representation
A time series representation that splits a time series into piece of fixed length and represent each piece with a polynomial.

RNN Recurrent Neural Network
A machine learning model featuring a memory cells for processing sequential data.

SAX Symbolic Aggregate approXimation
A symbolic time series representation based on PAA.

SOM Self-Organising Map
A manifold learning / nonlinear dimensionality reduction technique that learns a special neural network from competitive learning rather than minimizing error and tries to preserve the topological structure in data.

SSM State Space Model
A model relating a set of input, output, and state variables with first-order differential equations or difference equations.

QALSH Query-Aware Locality-Sensitive Hashing
A hashing algorithm based on LSH that generates hash tables suitable centered around the query.

STFT Short-Time Fourier Transform
A time series representation that conducts DFT/FFT piecewise.

SSA Singular Spectrum Decomposition
A time series representation based on Singular Value Decomposition (SVD) that decomposes a time series into interpretable components.

SVD Singular Value Decomposition
A technique for factorizing a matrix into a rotation matrix, followed by a scaling matrix, followed by another rotation matrix.

SVM Support Vector Machine
A machine learning model for classification and regression, which separates data with a hyperplane in the original space (linear), or a hyperplane in an infinite-dimensional space with a kernel trick (nonlinear).

TF-IDF Term Frequency–Inverse Document Frequency
A feature describing the importance of a word to a document in a corpus for text retrieval.

t-SNE t-distributed Stochastic Neighbor Embedding
A manifold learning / non-linear dimensionality reduction technique that tries to reproduce pairwise distances between data items in high-dimensional space in low-dimensional space and is good at preserving local structures.

UMAP Uniform Manifold Approximation and Projection
A manifold learning / non-linear dimensionality reduction technique similar to t-SNE but based on Riemannian geometry and algebraic topology and is good at preserving global structures.

UTS Univariate Time Series
A time series with only one track.

VAE Variational Autoencoder
A generative machine learning model with similar architectures to autoencoders but learns normal

distributions in its embedding layers, capable of generating new data by sampling the distributions.

VAR Vector Autoregressive
A time series model that extends the AR models for Univariate Time Series (UTS) to multivariate cases.

VARIMA Vector Autoregressive Integrated Moving Average
A time series model that extends the ARIMA models for UTS to multivariate cases.

VQS Visual Query System
A tool with a visual interface that allows users to specify and search for desired time series patterns, typically in a line chart.

WoS Web of Science
One of the world's biggest interdisciplinary indexes and search tools for academic literature.

WT Wavelet Transform
An integral transform that decomposes a signal into a series of contracted, expanded, and translated versions of a basis/kernel/wavelet function.

LIST OF SYMBOLS

The symbols listed in this chapter have fixed meanings. Unlisted symbols used in the main text may vary in meaning according to the context. Nonetheless, unlisted symbols are rare cases.

We use uppercase letters for matrices and sets; lowercase letters for scalers or functions; lowercase letters in boldface for vectors and arrays. This rule applies to both Latin and Greek symbols.

Table 1: Latin Symbols

Symbol	Meaning
$\mathbb{A}$	An alphabet or set of symbols (used in symbolic representations for time series).
C	A sampled found pattern candidate from the found patterns $\{F\}$ for user inspection and feedback.
C_+	An accepted / positively labeled sample.
C_-	A rejected / negatively labeled sample.
$\mathbb{C}$	Sampled found pattern candidates from the found patterns $\{F\}$ for user inspection and feedback.
$\mathbb{C}_+$	Accepted / positively labeled samples.
$\mathbb{C}_-$	Rejected / negatively labeled samples.
F	A found pattern.
$\mathbb{F}$	All found patterns in a time series.
FN	The number of false negatives.
FP	The number of false positives.
G	A ground truth.
$\mathbb{G}$	All ground truths in the time series. Please note that a time series may have target patterns $\mathbb{T}$, regardless of whether ground truths $\mathbb{G}$ are provided or not.
$IoU(.,.)$	Intersection over union between two time series subsequences.
$\mathbb{N}_0$	The set of non-negative integers.
$\mathbb{N}_{>0}$	The set of positive integers.
	$\vdots$

Table 1: Latin Symbols – Continued

Symbol	Meaning
P	A pattern (not necessarily a found pattern, a found pattern candidate, a ground truth, or a target pattern).
Q	Query to search for.
$R_{interim}$	Relevance of patterns starting at every time step with every resampled version of the query Q after averaging C_{raw} over tracks during the rule-based initial search.
R_{raw}	Relevance of patterns starting at every time step in every track with every resampled version of the query Q directly after similarity profile calculation during the rule-based initial search.
$\mathbb{R}$	The set of real numbers.
S	A time series.
$\underset{\sim}{S}$	A multivariate symbolic represented time series.
S_i	The scaler value at the i-th time step in the time series considered as a random variable from the perspective of stochastic processes for univariate time series.
$S(\alpha, \beta)$	Subsequence in S that starts with time step index α inclusive and ends with time step index β exclusive.
T	A target pattern.
$\mathbb{T}$	All target patterns in a time series.
TP	The number of true positives.
$Var(.)$	Variance of a random variable.
X	A time series subsequence; $S(\alpha, \beta)$ is used instead if the start time step index α and end time step β is known.
a_j	Aggregated distance between the j-th track in a positively labeled sample C_+ and the j-th track in the query Q.
a_j^*	Normalized aggregated distance a_j for j-th track.
$\boldsymbol{a}$	Aggregated distances for every track between a positively labeled sample C_+ and the query Q.
⋮	

Table 1: Latin Symbols – Continued

Symbol	Meaning
$\boldsymbol{a}^*$	Normalized aggregated distances $\boldsymbol{a}$.
b	Index for feedback round.
$\boldsymbol{c}_j$	The j-th track in a positively labeled sample C_+.
d	Dimension (number of tracks) of the time series.
$f_{dis}(.,.)$	Distance measure that takes two time series patterns as input and returns their distance.
$f_{rel}(.)$	A function that takes a subsequence $S(\alpha,\beta)$ of S as the input and returns the relevance of $S(\alpha,\beta)$.
$\tilde{f}_{rel}(.)$	A function that takes a subsequence $S(i-\zeta_\alpha, i+\zeta_\beta+1)$ around the time step s_i and returns the relevance of $\boldsymbol{c}_i$, e.g., the confidence/probability that the time step s_i is in a target pattern.
$f_{sim}(.,.)$	Similarity measure that takes two time series patterns as input and returns their similarity normalized to $[0,1]$.
$h(.)$	A hash function in a compound hash function in Locality-Sensitive Hashing (LSH).
i	Index of a time step.
$\underset{\sim}{i}$	Index of a symbol step in a symbolic represented time series V.
j	Index of a track.
k	Number of hash functions in a compound hash function.
l	Number of compound hash functions.
m	Number of resampled versions of Q, we set $m=8$.
n	Length (number of time steps) of the time series.
p	Order of an Autoregressive (AR) process.
q	Order of an Moving-Average (MA) process.
q_{ij}	The value at the i-th time step in the j-the track of the query Q.
$\boldsymbol{q}_i$	The i-th time step in the query Q.
$\boldsymbol{q}_j$	The j-th track in the query Q.
⋮	

Table 1: Latin Symbols – Continued

Symbol	Meaning
$\boldsymbol{q}_{ju}$	The j-th track in the u-th resampled version of Q.
r_i	Relevance of the pattern starting at the time step index i.
r_{iju}	Relevance of the univariate pattern starting at time step index i in j-the track, calculated with j-th track in u-th resampled version of the query Q.
r_{iu}	Relevance of the pattern starting at time step index i, calculated with u-th resampled version of the query Q after averaging over j for r_{iju}.
$\boldsymbol{r}_{merged}$	Relevance of patterns starting at every time step after selecting the best resampled version of the query Q with the highest relevance in $R_{interim}$ during the rule-based initial search.
$\tilde{r}_i$	Relevance of the time step with index i, e.g., the confidence/probability that it is located in a target pattern.
$\tilde{\boldsymbol{r}}$	Relevance of every time step in a time series, e.g, the confidence/probability of each time step in a time series that it is located in a target pattern.
s_{ij}	The value at the i-th time step in the j-th track of a time series S.
$\boldsymbol{s}_i$	The time step at index i in time series S.
$\boldsymbol{s}_j$	The j-th track in the time series S.
u	Index of a resampled version of Q.
$\underset{\sim}{v}_{\underset{\sim}{i}j}$	Symbol at the $\underset{\sim}{i}$-th time step in j-th track in a multivariate symbolic represented time series V.
$\underset{\sim}{v}'_{\underset{\sim}{i}'}$	The $\underset{\sim}{i}'$-th symbolic step in a symbolic represented time series with intertwined tracks $\boldsymbol{v}'$.
$\underset{\sim}{\boldsymbol{v}}'$	Symbolic represented time series V with intertwined tracks.
w_j	The j-th weight in the weights $\boldsymbol{w}$ used to update hash functions, which corresponds to the importance of the j-th track.
⋮	

Table 1: Latin Symbols – Continued

Symbol	Meaning
$\boldsymbol{w}$	Weights to update the parameters $\boldsymbol{\theta}$ in hash functions.
$\boldsymbol{w}_h$	Weights to update the hash function parameters according to relevance feedback on the hash functions.
$\boldsymbol{w}_{h,b}$	Weights in the b-th feedback round to update the hash function parameters according to relevance feedback on the hash functions.
$\boldsymbol{w}_h^*$	Interim unnormalized weights to update the hash function parameters according to relevance feedback on the hash functions.
$\boldsymbol{w}_{s,b}$	Weights in the b-th feedback round to update the hash function parameters according to relevance feedback on the samples.
$\boldsymbol{w}_s$	Weights to update the hash function parameters according to relevance feedback on the samples.
$\boldsymbol{w}_s^*$	Interim unnormalized weights to update the hash function parameters according to relevance feedback on the samples.
y_i	Whether the time step with index i is in a found pattern. If yes, $y_i = 1$, otherwise $y_i = 0$.
$\boldsymbol{y}$	An array of the same length as the time series containing boolean values of whether each time step in the time series is in a found pattern (1 if time step in a target pattern, else 0).
z_i	Whether the time step with index i is in a ground truth. If yes, $z_i = 1$, otherwise $z_i = 0$.
$\boldsymbol{z}$	An array of the same length as the time series containing boolean values of whether each time step in the time series is in a ground truth (1 if time step in a ground truth, else 0).

Table 2: Greek Symbols

Symbol	Meaning
Θ	Parameter vectors of all hash functions.
Θ_+	Parameter vectors of all hash functions accepted by the user.
α	Index of the start time step of a time series subsequence.
$\boldsymbol{\alpha}_{filtered}$	Start time step indexes of potential found patterns after filtering α_{nms} with a relevance threshold τ.
$\boldsymbol{\alpha}_{nms}$	Start time step indexes of potential found patterns after non-maximum suppression in α_{raw}.
$\boldsymbol{\alpha}_{raw}$	Local maximum points in merged relevance profile $\boldsymbol{r}_{merged}$, which are start time step indexes of potential found patterns.
β	Index of the end time step of a time series subsequence.
$\gamma_{\Delta i}$	The coefficient for the Δi-th past value in the MA model. The common symbol is in the literature is θ instead of γ. However, the former is occupied in the thesis.
$\varepsilon_{i-\Delta i}$	The Δi-th past innovation of the time series model.
ζ	Lookaround. To classify a time step s_i, its proximity $S\left(i-\zeta,\, i+\zeta+1\right)$ is used as input data.
ζ_α	Lookbehind, number of historical time steps to consider when classifying whether a time step as in a target pattern or not. In practice, we set $\zeta_\alpha = \zeta$.
ζ_β	Lookahead, number of future time steps to consider when classifying whether a time step as in a target pattern or not. In practice, we set $\zeta_\beta = \zeta$.
$\tilde{\zeta}$	Length of the neighborhood around a positively labeled pattern $S\left(\alpha,\beta\right)$, within which the time steps are considered negative, i.e., all time steps in $S\left(\alpha-\tilde{\zeta},\, \alpha\right) \cap S\left(\beta,\, \beta+\tilde{\zeta}\right)$ are considered negative.
η	Relevance threshold, which the relevance of a found pattern candidate should reach to be recognized as a found pattern.
	⋮

Table 2: Greek Symbols – continued

Symbol	Meaning
θ_j	The j-th parameter in a hash function corresponding to the weight for the j-th track.
$\boldsymbol{\theta}$	Parameters in a hash function.
$\boldsymbol{\theta}'$	Parameters in a retrained hash function after relevance feedback.
ι	Bin size for Piecewise Aggregate Approximation (PAA) and Symbolic Aggregate approXimation (SAX).
λ_{lower}	The lower bound of the horizontal scaling factor for Q.
λ_{upper}	The upper bound of the horizontal scaling factor for Q.
ρ	Learning rate for updating the track weights.
ϕ	Window length (number of time steps in the window).
ϕ_u	Length of the u-th resampled version of Q.
$\varphi_{\Delta i}$	The coefficient for the Δi-th past value in the AR model.
$\tilde{\phi}_i$	Length of the chosen resampled version of Q for the time step with index i.
$\boldsymbol{\phi}$	Lengths of every resampled versions of Q.
$\tilde{\boldsymbol{\phi}}$	Lengths of the resampled version of Q chosen for every time step.
$\underset{\sim}{\phi}$	Word length of the symbolic represented query Q.
$\boldsymbol{\tau}$	Indexes of time steps estimated to be in a target pattern.
υ	Intersection over Union (IoU) threshold.
φ	Radius used in Non-Maximum Suppression (NMS) in the rule-based initial search.
ψ	Hash collision threshold for LSH.
ω	Projection collision threshold (hash bucket size) within which project collisions happen for LSH.

Table 3: Other Symbols

Symbol	Meaning
$\underset{\sim}{\cdot}$	In symbolic space.
$\odot$	Element-wise product, aka Hadamard product, entry-wise product, or Schur product.
$\lfloor . \rfloor$	Floor function.
$[.]$	A matrix.
$\{.\}$	A set.
$\lvert . \rvert$	Cardinality (number of entries) of a set or length of a vector.
$\lVert . \rVert$	L^2 norm.
$\mathrm{Cov}\,(.,.)$	Covariance of two random variables.
$\mathrm{E}\,(.)$	Expectation of a random variable.
$\mathcal{T}\,(.)$	An integral transform.
$\mathrm{Var}\,(.)$	Variance of a random variable.
div	Integer division.
mod	Modulus.

1 INTRODUCTION

In this chapter, we will introduce the domain background that motivated our research, followed by the challenges that prevail in the academic discipline. Thereupon, we will pose our research question and requirements. Based on them, we will derive concrete goals that envision the deliverables of our technical work. Subsequently, we will lay out the contributions that we have made or are making in our completed and ongoing projects. Finally, we will sketch an outline of the rest of this thesis. Table 12 in Chapter 9 summarizes the challenges, requirements, goals, and contributions addressed in each chapter.

1.1 MOTIVATION

Searching for recurrences of a given event in chronologically recorded data is a ubiquitous task in various domains, such as discovering astronomical objects like supernovae or quasars in brightness transient observations [155], locating pathogenic fragments in genomic sequences [156, 239], tracking events in Internet of Things (IoT) data collected from inertial measurement units in mobile devices [153], and detecting similar market behavior in historical stock prices [2, 90, 97].

Our use cases come from our industrial partner IAV GmbH Ingenieurgesellschaft Auto und Verkehr (hereinafter called IAV). They are mainly in the automotive domain, where engineers frequently and spontaneously search for event-induced patterns in measurements from various sensors and control units, e.g., for subsequent error root cause diagnosis. For instance, our engine engineers await an indicating signal to pinpoint the best entry point of their novel nozzle control algorithm for a smoother air-fuel ratio control, which should be activated immediately after the indicating signal; our transmission engineers trace gear change events to examine moments on different shafts to optimize the control strategy for the next-generation dual-clutch transmission and reduce traction loss during gear shifts; Noise, Vibration, and Harshness (NVH) engineers hunt for transition phases when half of the cylinders are deactivated or reactivated for automobiles with a variable displacement engine to analyze and suppress acute vibrations during these periods.

These real-world use cases are, at an abstract level, pattern search in time series, one of the tasks in time series analysis illustrated in Figure 1.1. Time series pattern search addresses the problem of finding patterns along the time axis in a time series that are similar to one or several given examples. It serves as a frequent enabler for downstream domain-specific data analysis that requires multiple examples or as many occurrences of the interesting event as possible.

Our engineers from various departments, especially engine calibration, complain that they need to manually search for events in measurements several times a week, each costing hours of working time. If failed to procure sufficient samples of the interesting event from historical records, they have to commission new test drives, incurring even more time and costs. In this regard, IAV initialized this research project on Multivariate Time Series (MTS) pattern search.

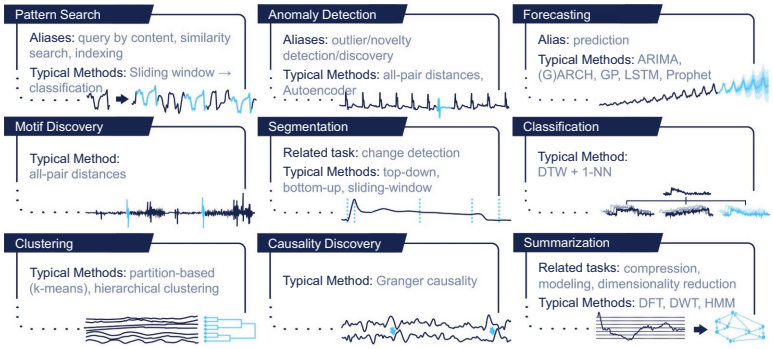


Figure 1.1: **Tasks in Time Series Analysis.** Time series analysis is the discipline that studies methods for extracting information and knowledge from time series data. It subsumes (but not limited to) nine typical tasks [85, 89, 94, 138].

1.2 CHALLENGES

Multivariate time series pattern search faces nine challenges (abbreviated as Cs) illustrated in Figure 1.2.

C1 Temporal Length and C2 High Dimensions are dictated by properties of the data and thus common for all tasks in time series analysis.

C3 Duration Scaling, C4 Inter-Track Time Shifts, and C5 Context Obliviousness relate to the query and the target patterns, and are specific to time series pattern search.

C6 Unexplainable Model and Its Training Process, C7 Inefficient Model (Re)training, and C8 Meager Labels affects model-based time series pattern search with a machine learning model.

C9 Subjective and Task-Dependent Similarity and Relevance denounces rule-based time series pattern search that does not require optimizing parameters in the data processing pipeline and is thus not steerable.

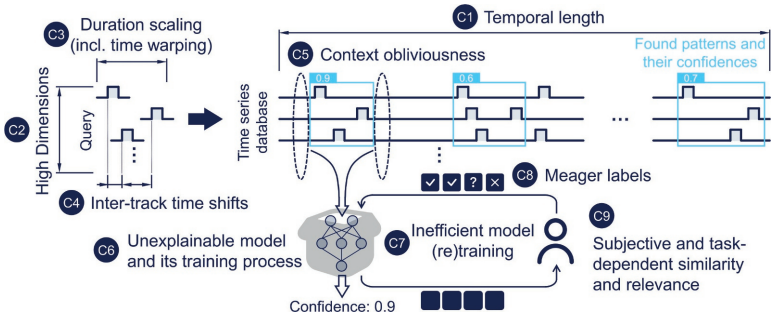


Figure 1.2: **Challenges.** The task, multivariate time series pattern search, faces nine challenges. We will address all of them in this thesis.

C1 Temporal Length. Time series data can span numerous time steps due to fine sampling rates or long measurement durations. For instance, our measurements from Engine Control Units (ECUs) typically last several hours with a sampling rate of 1 millisecond, easily reaching ten million time steps; astronomical and weather observations accumulate every year [166]; IoT and Industrial 4.0 devices generate unprecedented big data, with an estimation by 2025 of over 75 billion connected devices worldwide [183] and a global economic impact of up to 11.1 trillion [64]. Research in time series analysis has entered an era of trillions of time steps [174, 175, 211]. When we conduct time series pattern search on a standard laptop, processing a time series with one million time steps typically costs several minutes for most algorithms implemented in Python. Occasionally, out-of-memory errors occur. In addition, most visualization libraries like PlotlyJS and BokehJS become sluggish, making interactive exploratory search user-unfriendly. Our work in Chapter 6 scans the time series data with a regular expression along the time axis instead of the de facto standard sliding window.

C2 High Dimensions. A time series can record multiple variables over time, each revealing the state of the observed object or system from one perspective. Sometimes it is only after syn-

thesizing information distributed across dimensions that it becomes possible to retrieve the events of interest [275, 276]. For instance, a typical Electroencephalogram (EEG) uses no less than 64 electrodes to measure the electrical activity in the brain [221]. Our ECU data comprise more than nine thousand signals/dimensions, although a specific analysis task may require only a subset of them. Moreover, multiple dimensions also increase processing time many times. All our works take multivariate suitability into account. In addition, our work in Chapter 5 merges all tracks into one with LSH, so that subsequent data processing scales sublinearly/constantly with respect to the number of dimensions.

C3 Duration Scaling. The time spans of the target patterns may differ. For instance, in the EEG Eye State dataset [221], the longest periods of the target event (when the subject's eyes are closed) are more than ten times longer than the shortest ones. This is a nontrivial challenge due to the de facto standard usage of sliding windows to scan the time series [153, 156, 182]. A fixed-sized sliding window can only retrieve patterns of the same length. Conversely, a range of sliding windows of various lengths multiplies the processing time. Our work in Chapter 6 exploits quantifiers in regex to capture duration scaling. Furthermore, our work in Chapter 7 classifies time steps instead of time series windows, which scans the time series only once with machine learning models and retrieves duration scaled patterns.

C4 Inter-Track Time Shifts. In practice, an event may delay its manifestation in the data, and such delay may vary in different tracks [276]. This is often the case with CAN-Bus data, where the delays and their differences may be even greater than the pattern duration in each track. Our work in Chapter 6 combines regex techniques like quantifiers, character classes, and wildcards to capture various time series pattern distortions, including inter-track time shifts.

C5 Context Obliviousness. In some of our use cases, the target patterns do not show distinctive internal characteristics, while they form prominent patterns together with their immediately preceding and succeeding data. This is especially the case when the event is a phase in a multiphase process. Our unique approach of time-step-based classification in Chapter 7 is aware of the pattern context during retrieval.

C6 Unexplainable Model and Its Training Process. It is ubiquitous in academic work to train a machine learning model to represent time series, estimate time series properties, and measure the relationships between time series [184, 190, 278]. How-

ever, domain users hesitate to apply machine learning due to the inaccessibility to its mechanism. As a result, they are unable to deduce the causes of the errors and are skeptical of its reliability. Our work in Chapter 5 turns LSH into a model and trains it with an explainable mechanism that returns feature/track importance as a by-product.

C7 Inefficient Model (Re)training. Another challenge faced by model-based time series pattern search is the computational costs caused by expensive mathematical optimization for model training [275]. In our case, calibration engineers query measurements spontaneously and often wish for an immediate answer. They expressed concern about waiting several hours for model training, let alone hyperparameter tuning, before consuming the model. Our work in Chapter 5 updates the model according to data variance instead of mathematical optimization, which allows instantaneous model adaptation.

C8 Meager Labels. Machine learning models are notorious for their hunger for labeled data [81, 147, 177, 283]. In practice, our engineers may not and usually cannot provide many examples of the target patterns. Hence, the feasibility of machine learning is in question. Our work in Chapter 5 and Chapter 8 solicits labels from user feedback to train the model actively. Moreover, the technique "nonmyopic search" proposed in Chapter 7 indulges the model's gluttony for labels because a user-labeled pattern will be translated to hundreds and thousands of training labels.

C9 Subjective and Task-Dependent Similarity and Relevance. Pattern similarity and relevance are fuzzy and hard to describe. Worse still, they may contain domain knowledge or information specific to use cases [147]. Consequently, by no means can one set of fixed rules capture all kinds of pattern similarity/relevance [65]. Our work in Chapter 5, Chapter 7, and Chapter 8 studies model-based pattern search which adapts to the needs of the task and the user.

We acknowledge that a single solution to all of the challenges is unrealistic. Therefore, we will try to address a subset of them each time and propose a toolbox where our domain users can choose methods according to the dominating challenges in concrete use cases.

1.3 RESEARCH QUESTION

We interviewed 23 domain engineers and solicited 40 user stories. After analyzing their user cases, we posed the following overarch-

ing research question: **How to query patterns in MTS accurately, efficiently, and understandably?**

This research question imposes four (business) requirements conveyed by the adjectives and adverbs therein.

R1 Multivariate Suitability. It is an indispensable functional requirement because the phenomena that interest our engineers require consideration of multiple signals, as mentioned in C2 High Dimensions.

R2 Sufficient Accuracy. It is a standard non-functional requirement. Unlike safety/security-critical scenarios, e.g., medical diagnosis and security surveillance, our domain users typically need a couple of examples of a target event, which does not need to be exhaustive. Accordingly, we use the word "sufficient" instead of "high", implying diminishing user satisfaction growth after the accuracy exceeds a certain threshold. Nonetheless, this requirement is nontrivial with C4 Inter-Track Time Shifts, C5 Context Obliviousness, C8 Meager Labels, and C9 Subjective and Task-Dependent Similarity and Relevance in the way.

R3 High Efficiency. Like accuracy, it is also a self-explanatory non-functional requirement. C1 Temporal Length, C2 High Dimensions, C3 Duration Scaling, C4 Inter-Track Time Shifts, C5 Context Obliviousness and C7 Inefficient Model (Re)training may call for extra computational resources. This requirement is especially relevant if we expect the query system to be interactive. R3 High Efficiency is often in conflict with R2 Sufficient Accuracy. Our engineers are constantly under stress and wish for a prompt outcome. Considering their moderate requirement of retrieval accuracy, we incline slightly towards efficiency in the trade-off between R2 Sufficient Accuracy and R3 High Efficiency. Specifically, we endeavor to be faster than the state-of-the-art methods with comparable accuracy.

R4 Explainability. It is an optional functional requirement that addresses C6 Unexplainable Model and Its Training Process. This requirement urged our tool to expose its working mechanism to our engineers so that they can reason the retrieval process, predict its behavior, explain its failures, and build trust in it. Among others, our domain engineers wish that the algorithm can hint at and rank the most important tracks/features that contribute most to detecting the target event. For instance, our engineers search for anomaly examples in measurements, followed by an error root analysis based on the tracks that exhibit the strongest causal relationship with the anomaly.

1.4 GOALS AND SCOPE

After deriving the research question and the business requirements from the 40 user stories from the 23 domain engineers in Section 1.3, we broke them down into tangible and manageable technical requirements or features. Then, we converted them into actionable and measurable goals, each catering to an atomic technical requirement that cannot or had better not be broken down. After that, we categorized them according to the MoSCoW method [25, 59] into must-have goals, should-have goals, could-have goals, and won't-have goals.

The **must-have goals** (abbreviated to "MG") aim to lay a foundation for academic contributions and business impacts. We must attain all of them to arrive at the Minimum Viable Product (MVP).

MG1 **STAR Analysis.** The Ph.D. student must review state-of-the-art algorithms on time series pattern search.

MG2 **Baseline Algorithms.** The Ph.D. student must select and implement representative state-of-the-art algorithms as baselines if no suitable libraries of the selected algorithms exist.

MG3 **Basic Algorithm.** Design and implement novel algorithms for time series pattern search that are not necessarily suitable for multivariate cases.

MG4 **Algorithmic Benchmark.** Benchmark the developed algorithms with the baselines in terms of retrieval accuracy and speed.

MG5 **Basic App.** Design and implement web applications to assist domain users with no coding experience in applying the developed algorithms, which, at its bare minimum, must support inputting data, calling the algorithm code, and exporting found pattern intervals.

MG6 **Basic Visualizations.** The developed web application must be able to visualize the whole measurements and the found patterns.

MG7 **Query-by-Example.** The developed web application must support query definition by marking an interval in a line chart of a data file (query-by-example).

MG8 **Case Study.** Evaluate the usability of the web application with case studies (without the involvement of users).

The **should-have goals** (abbreviated to "SG") are natural extensions of the MGs, as illustrated in Table 4. They target the

requirements in Section 1.3 and strive for significant academic contributions as well as business impacts.

SG1 **Multivariate Suitability.** The developed algorithms should apply to multivariate cases.

SG2 **Single Example.** The developed algorithms should support beginning with one example of the target events, as the user may not be able to provide many examples of the target event.

SG3 **Performance Gain.** The developed algorithms should bring significant performance improvement (functional, accuracy, speed) against the state-of-the-art baseline algorithms under certain nontrivial circumstances.

SG4 **Adaptive Algorithm.** The developed algorithms should be adaptive and steerable to converge to the user's notion of pattern relevance.

SG5 **Streamlined Workflow.** The developed web application should have a distinctive and streamlined workflow that the user can follow easily.

SG6 **Click & Run.** The developed web application should have as few as possible *mandatory* configurations and interactions (ideally click & run), but optional ones (e.g., collapsible panels or an expert mode) are allowed.

SG7 **Expert Studies.** The usability of the web application should be validated through expert studies with domain users.

Like the SGs, the **could-have goals** (abbreviated to "CG") also extend the MGs, as illustrated in Table 4. Although they also contribute to academic research and business valorization, they approach the border of our scope. Some CGs study alternative and complementary techniques, e.g., query-by-sketch in contrast to our query-by-example query definition approach. Some CGs consolidate our side work, e.g., a survey for time series representations and an evaluation framework for time series pattern search. Others are speculative and nice-to-have features that do not affect the essential functionality of our method, e.g., track importance estimation and intelligent assistance. Due to the time constraint, we consider them optional.

CG1 **Transformation Survey.** The Ph.D. student could survey problem space transformations for time series and create a taxonomy for them, discussing their merits and drawbacks, as well as suggesting their proper use cases.

CG2 **Track Importance.** The developed algorithms could be explainable, especially in terms of track importance (which tracks contribute more to the detection of the target event).

CG3 **Evaluation Framework.** The Ph.D. student could design and implement a unified evaluation framework.

CG4 **Query-by-Sketch.** The developed web application could support query definition by drawing a prototype of the desired patterns in a canvas (query-by-sketch).

CG5 **Intelligent Assistance.** The developed web application could provide helpful, ideally intelligent, suggestions to the user.

Similar to the situation in Section 1.2, we cannot fulfill some goals simultaneously. Instead, we provide a toolbox with multiple methods designed to meet different subsets of the goals. Combined, the toolbox strives to cover all MGs, SGs, and as many CGs as possible.

We consider the following points (**won't-have goals**, abbreviated to "WG") out of the scope of our work and will forgo them.

WG1 **Engineering.** Heavy engineering work, product design, or application of existing technology, e.g., indexing with Apache Lucene, parallel computing with Hadoop or Spark, or user management, since they do not contribute academically.

WG2 **Models for Marginal Gain.** Devising new machine learning models, improving the architectures of existing models, or testing new combinations of existing machine learning models and existing data processing pipelines, for the sole purpose of marginal performance gain (as in Kaggle competitions or publications resulting from competitions for specific datasets), unless the changes enable new functions useful in our use cases.

WG3 **Hyperparameter Optimization.** Excessive hyperparameter optimization during algorithm evaluation, because extensive offline model training before inference/production is infeasible in and the optimal performance achieved this way does not apply to our spontaneous and highly dynamic use cases. Pertaining to SG6, the performance should be robust to hyperparameter setting, and a set of default values should generally work well.

Apart from the goals, our work must comply with regulatory conditions in IAV irrelevant to the research work. For instance,

	MG1: STAR Analy.	MG2: Baselin. Alg.	MG3: Basic Alg.	MG4: Alg. Benchm.	MG5: Basic App	MG6: Basic Vis.	MG7: Query-by-Ex.	MG8: Case Studies
SG1: Multivar. Suitability			■					
SG2: Single Example			■					
SG3: Performance Gain				■				
SG4: Adaptive Alg.			■					
SG5: Streamlined Workflow					■			
SG6: Click & Run					■			
SG7: Expert Studies								■
CG1: Trans. Survey	■							
CG2: Track Importance			■					
CG3: Eval. Framework				■				
CG4: Query-by-Sketch							■	
CG5: Intelligent Assistance						■		

Table 4: **Relationships between SGs/CGs and MGs.** One dark blue square indicates the extension of a must-have goal by a should/could-have goal. The MGs lay the foundation of the research and their fulfillment leads to the MVP. In comparison, the SGs and CGs target significant academic contributions and business impacts. The goals focus more on the algorithmic side and less on visualizations and interactions.

choices of programming language (e.g., Python instead of MATLAB) and libraries (e.g., Django instead of FastAPI), interface design (e.g., formats of input/output artifacts), corporate identity design (e.g., color scheme), software tests and coverage, and license checks (e.g., restricted usage of copy-left open-source software), which we omit to enumerate in this academic thesis.

1.5 CONTRIBUTIONS

In the course of working towards the goals laid out in Section 1.4, we completed three major research projects on novel algorithms (distributed from Chapter 5 to Chapter 8), are working on the fourth project on a taxonomy for time series representations (Chapter 3), and planned a fifth project on an evaluation framework for time series pattern search (discussed briefly under the point "Unified Evaluation" as future work in Chapter 9). They deal with all challenges in Section 1.2; they center around the same research question posed in Section 1.3 and cover all four requirements prescribed by it; they have attained all MGs and SGs plus CG2 and are working on CG1 as well as CG3. The remaining CG4 and CG5 are in our backlog as future work.

Our work intended to contribute to algorithm development and visual analytics for multivariate time series pattern search, with emphasis on the algorithmic side. The contributions and their significance in the three aspects are listed in Table 5.

Our tools are deployed in IAV and Volkswagen and are helping automotive calibration engineers trace events in massive measurements.

1.6 THESIS OUTLINE

We organize the subsequent chapters in this doctoral thesis according to the contributions listed in Section 1.5. Please refer to Table 12 for the addressed challenges, requirements, and goals in each chapter.

Chapter 2 provides necessary technical background and reviews the state-of-the-art methods in this research field. It goes from the data (time series), over the task (time series analysis and pattern search), to essential techniques.

Chapter 3 addressing CO 1 establishes a taxonomy for time series representations and provides guides for effective choices among them in practice.

No.	Ch.	Contributions	Alg.	Vis.	Int.
CO1	3	A taxonomy for time series representations (ongoing)	+++	-	-
CO2	4	A unified evaluation framework for time series pattern search (ongoing)	++	+++	+
CO3	5	An efficient, steerable, and interpretable representation for very high-dimensional time series	+++	+	+
CO4	6	Description of pattern distortion & retrieval invariance, especially for inter-track time shifts	+++	+	+
CO5	7	Nonmyopic Machine-Learning-based pattern search capturing duration scaling and pattern context	+++	+	+
CO6	8	Multi-feature feedback for capturing user's notion of relevance	+	-	++

Table 5: **Contributions.** This thesis makes six distinctive contributions. We assessed their significance in terms of algorithms, visualizations, and interactions (abbreviated as Alg., Vis., and Int., respectively, in the table). "-" means no noteworthy overlap, "+" means application of established techniques but no contributions, "++" means tangential or minor contributions, and "+++" means major contributions.

Chapter 4 originates from the ongoing CO 2 and introduces the evaluation metrics that we propose for benchmarking retrieval accuracy. We will use them from Chapter 5 to Chapter 8 throughout. The visual analytics part of the metrics proceeds as an ongoing master thesis.

Chapter 5 is devoted to CO 3 and presents our extension of a hashing-based data representation for very high-dimensional time series. It steers the representation to align with the user's notion of relevance with an understandable model updating mechanism while retaining the efficiency of the original representation.

Chapter 6 dedicated to CO 4 proposes our extension of a symbolic representation for UTS to multivariate cases. Based on the extended representation, we formulate the query, i.e., the target event, as a regular expression and exploit techniques in regex to describe various distortions in time series patterns, especially heterogeneous duration scaling and inter-track time shifts.

While Chapter 5 and Chapter 6 aim to push back the frontiers of rule-based pattern search that does not involve machine learning, Chapter 7 and Chapter 8 attempt to advance (machine-learning)-model-based pattern search.

Chapter 7 tackling CO 5 put forward a data processing pipeline called "nonmyopic search" that combines time-step-based classification and density-based-clustering. It alters how (the same) machine learning models are used for time series pattern search to capture duration scaling and pattern context, ultimately leading to higher accuracy and speed simultaneously.

Chapter 8 concerning CO 6 enhances the existing relevance feedback used in feedback-driven pattern search with other features of the retrieved patterns besides relevance. It collaborates closely with nonmyopic search and drives accuracy to an even higher level.

Finally, Chapter 9 concludes our work and discusses future work.

2 RELATED WORK

This chapter systematically covers the technical background and reviews state-of-the-art methods related to the central topic MTS pattern search, while we defer related work that only pertains to a specific chapter to the related work section of that chapter. For instance, we will introduce time series representation in this chapter because it is an essential technique for the central topic. Whereas, we will elaborate on SAX, a special time series representation, in Section 6.2, because it is fundamental to Chapter 6 but would disturb the information flow in this chapter.

This chapter will begin with dissecting the studied data type, i.e., time series in Section 2.1. Then, Section 2.2 introduces the research field time series analysis, followed by Section 2.3 that zooms in on the task time series pattern search. The remaining sections focus on essential techniques for this task, including data representations, similarity measures, traversing approaches, retrieval invariance, active learning, and user interface design.

2.1 TIME SERIES

Time series, like tabular data, graph data, image data, text data, etc., is one of the most common and basic data types. Subsequently, we will define time series and dissect its two dimensions.

2.1.1 *Definition of Time Series*

A time series is a sequence of real-valued observations recorded chronologically [32, 94, 109, 186]. It is one of the most ubiquitous data types which describes, for example, sales development [10, 200], energy consumption [70, 241], stock price fluctuations [78, 97], IoT measurements [64, 152], audio recordings [86, 135], concentration changes during chemical processes [16, 60], genetic sequences [17, 156], medical graphs like EEG [195, 280] and Electrocardiogram (ECG) [178, 245].

Formally, we define a time series

$$S = \left[\boldsymbol{s}_i \in \mathbb{R}^d\right]_{0 \leq i < n} = \left[s_{ij}\right]_{\substack{0 \leq i < n \\ 0 \leq j < d}}$$

of length n (number of time steps) and dimension (number of tracks) d as a permutation of n real-valued vectors, each of which has d entries. We adopt the matrix notation with square brackets for an array of indexed elements with duplicate allowed. We impose simple constraints (e.g. "$\in \mathbb{R}^d$" in this case) directly with a representative element in the square brackets. The subscription describes the ranges of the indexes for the elements. We use the set notation with curly brackets $\{.\}$ for a group of not necessarily ordered elements without duplicates, such as all found patterns. Please refer to List of Symbols for all symbols used consistently in this thesis.

2.1.2 *Two Axes in Time Series*

The data values in a time series span two axes, i.e., the time axis and the attribute axis.

Time Axis

Along the time axis lies the essential property of time series, i.e., the time steps / data samples are dependent or ordered [31, 250], and the dependency or order carries information, as indicated by the word "permutation" instead of "combination" in our definition of a time series. This property differentiates time series from tabular data containing a combination of rows / data samples. Interestingly, this order does not necessarily depend on the notion of time, let alone the existence of time stamps; and the data values are not strictly required to be real numbers readily. For instance, researchers also regard genomic sequences [156] and sequences representing two-dimensional shapes [141, 269] as time series. It is reasonable and beneficial to subsume such sequential data under the data type time series, because they respect the essential property of time series, and methods for time series analysis often apply to them as well.

Data processing along this axis is prone to C1 Temporal Length. To avoid scanning along the time axis, one may consider applying a transform like Fast Fourier transform (FFT) or model the temporal dynamics to work outside the time domain, providing the time series is stationary, i.e., its statistics like mean, variance, etc. do not change over time [230, 238]. If a time series is non-stationary but the dynamic of the non-stationary component is known, it is possible to remove the non-stationarity or extend the model. If such transformations or modeling are infeasible, one may consider methods without many assumptions of the an-

alyzed time series, e.g., indexing [2, 44, 46] and computational techniques [174, 188] to alleviate C1 Temporal Length.

We will review techniques for traversing the time axis and mention our work in this respect briefly in Section 2.6.

Attribute Axis

If $d > 1$ and the time series comprises multiple dimensions, it is multivariate, high-dimensional, or multichannel. Various disciplines have diverse terminologies referring to a dimension in a time series. Literature on time series analysis favors the terms "channel" and "multivariate time series". "Channel" is also the language of our engineers. Consequently, we use it in the user interfaces designed for them. Audio data are recorded in "tracks". This thesis adopts this terminology, following the tradition in our research group studying multimedia data. The graph library `Plotly` uses the word "trace" Other aliases include "signal", "variable", "attribute", "feature", "dimension", and "series".

Data processing across this axis is subject to C2 High Dimensions [166]. Each track may depict the measured object or process from one perspective [250]. Synthesizing information distributed across multiple tracks can proceed, succeed, or even accompany data processing along the time axis. For instance, [237] suggested two possibilities to extend Dynamic Time Warping (DTW), one of the most popular distance measures between two UTS, to MTS: 1) by regarding each time step, that was previously a scalar, as a point or vector, and performing the very same calculations on them, replacing scalar subtraction with Euclidean Distance (ED); 2) by computing conventional univariate DTW for each track independently, and averaging them subsequently. They call the first extended version DTW_D (the subscript "D" stands for "dependent") and the second DTW_I (the subscript "I" stands for "independent"). Furthermore, their experiments concluded that neither version is universally superior. Another example is the extension of AR models to Vector Autoregressive (VAR) models, where the observations / time steps are vectors instead of scalars, and the previously scalar coefficients before the historical observations in the equation become matrices, allowing a description of temporal dynamics and inter-track relations simultaneously.

R1 Multivariate Suitability dictates that we confront C2 High Dimensions face to face. All our proposed algorithms apply to MTS. Particularly, we designed `PSEUDo` for very high-dimensional time series and `SAXRegEx` for simultaneously analyzing temporal and inter-track information.

2.2 TIME SERIES ANALYSIS

The term "time series analysis" [32, 111, 238], also known as "time series data mining" [85, 89, 94], refers to the process of information and knowledge discovery from time series data. Intrinsically, it is studied roughly in nine tasks; extrinsically, it is closely related to several other disciplines, which we will present in the subsequent subsections.

2.2.1 *Tasks in Time Series Analysis*

There are many established problems/tasks in the research field of time series analysis [94, 138]. We pick the nine most typical ones illustrated in Figure 1.1. Based on the use case, analysts may want to conduct

PATTERN SEARCH / QUERY BY CONTENT / INDEXING, ETC. Locating patterns that are similar to a given one, which represents an event of interest and is worthy of further investigation [6];

ANOMALY DETECTION Detect extraordinary events that may relate to errors or novelties [26, 157, 234];

PREDICTION / FORECASTING Extrapolate future development of the data [21, 50, 184];

MOTIF DISCOVERY Unearth previously unknown recurrent behavior, e.g., for association rule learning [4, 212, 249];

SEGMENTATION / CHANGE DETECTION Split the data into consecutive pieces that are homogenous internally and heterogeneous with each other, e.g., to analyze phases individually [102, 171];

CLASSIFICATION Categorize time series and thereby the items that they measure to known groups [189, 190];

CLUSTERING Divide multiple time series into (previously unknown) groups, each with characteristics shared by group members but unique to other groups, in order to reveal patterns in common behavior [5, 121, 163];

CAUSALITY DISCOVERY Recognize causal relationships between two tracks or two time series with the same time axis [13, 191], which is less mentioned in many surveys on tasks in time series analysis [85, 89, 138]; and

SUMMARIZATION Summarize and usually visualize the essential features of long time series [138, 196, 243].

During our literature review, we found that a systematic analysis of the state-of-the-art algorithms for time series segmentation and pattern search is missing, especially the latter, albeit with an abundance of methods proposed.

Our work falls into the category pattern search. Strictly speaking, we consider indexing a (preprocessing) step for pattern search. However, many works use the term time series indexing to address pattern search and even define this term explicitly this way. Therefore, we include it as an alias for pattern search. We will introduce the task time series pattern search and review its state-of-the-art methods in Section 2.3.

2.2.2 *Intersections with Other Disciplines*

Observed in a broader scope, time series analysis intersects multiple other disciplines, where (temporal) sequential data analysis is of concern. Specifically, a considerable portion of the methods in time series analysis originate from other disciplines. They interpret time series dynamics differently and sometimes make domain-specific assumptions. Figure 2.1 shows four disciplines closely related to time series analysis and time series pattern search. We will review these interpretations and assumptions in detail in Chapter 3.

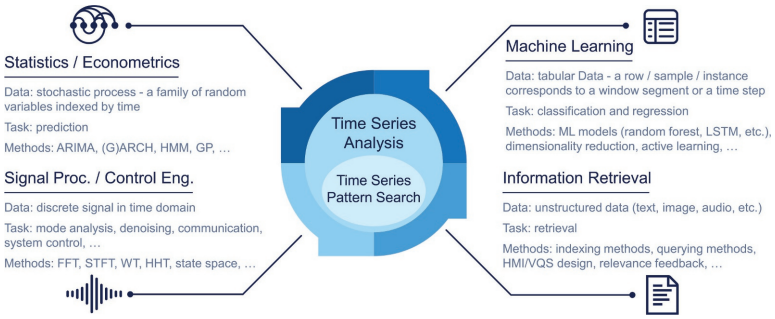


Figure 2.1: **Intersections of Time Series Analysis Including Pattern Search With Other Disciplines.** Observed in a broader scope, time series analysis and time series pattern search assimilate methods from multiple intersecting disciplines, which observe the data from different perspectives, solve distinctive issues, and propose unique techniques.

2.3 TIME SERIES PATTERN SEARCH

As illustrated in Figure 1.1, time series pattern search addresses the problem of locating sub-sequences, which are similar to a target example, in a time series. It appears in the literature under the aliases of pattern search [156, 219], time series indexing [44, 109, 138], query by content [86, 104, 129, 132, 136, 138, 140, 143, 210], similarity search [104, 193], subsequence matching [90, 106, 139], subsequence search [92], twin search [46, 47], time series querying [182], and so on. In our reviewed literature, "similarity search" appears most frequently. However, we slightly prefer "pattern search" because it is most clear and accessible to us.

For a precise description, we define

- a subsequence $S(\alpha, \beta)$ of S, starting at α-th time step inclusive and ending at β-th time step exclusive as

$$S(\alpha, \beta) = [\boldsymbol{s}_i]_{\alpha \leq i < \beta} \tag{2.1}$$

- a query as

$$Q = \left[\boldsymbol{q}_i \in \mathbb{R}^d\right]_{0 \leq i < \phi} = [q_{ij}]_{\substack{0 \leq i < \phi \\ 0 \leq j < d}} \tag{2.2}$$

  of length ϕ; and

- a relevance function

$$f_{rel} \colon \mathbb{R}^{(\beta-\alpha) \times d} \to [0, 1] \tag{2.3}$$

  which takes a subsequence $S(\alpha, \beta)$ of S as its input and returns the relevance of $S(\alpha, \beta)$, i.e., the confidence/probability that it is a desired pattern (we will use "relevance" in the thesis, which is from our perspective more precise, and "confidence" in our user interfaces preferred by our users).

We define the task time series pattern search formally as follows: given a time series S and a query Q, return all subsequences $S(\alpha, \beta)$ of S whose relevance is no less than a threshold η, namely, return

$$\{S(\alpha, \beta) \mid f_{rel}(S(\alpha, \beta)) \geq \eta\}$$

Researches on time series pattern search mainly focus on two directions: 1) novel relevance functions f_{rel}, including rule-based similarity measures and machine learning models, which better

describe the notion of relevance/similarity and focus on accuracy [103, 107, 201, 284]; 2) indexing techniques focusing on efficiency [2, 47, 202].

We made contributions in both directions and will mention them in Section 2.5 and Section 2.4, respectively, among other contributions.

2.4 TIME SERIES REPRESENTATIONS

Time series representations are ubiquitous during time series analysis. General data preprocessing includes data cleansing, feature & sample selection, outlier removal, normalization, and data transformation [230]. In time series analysis, data transformation is synonymous with time series representation. It transforms a time series into another form to compress data volume, reduce dimensionality, or reveal latent features, ultimately enabling more effective or efficient data analysis.

We define a time series representation as a transformation technique (rather than, rigorously speaking, the outcome of this technique) that converts a time series into another form while retaining information on its salient dynamics, with which it is viable to reconstruct the original data or synthesize data possessing similar characteristics.

It is inconsequential that the transformed time series stays in the time domain, as with most piecewise representations [44, 136]. Naturally, it is also plausible for a representation to transform a time series into another domain, like the frequency domain, as with FFT, or an embedding space, as with a dimensionality reduction technique. A representation is valid, as long as the information loss or distortion does not prohibit an inverse transformation to reproduce the original time series. Such reproduction extends to the generation of synthetic data with similar characteristics, as with generative machine learning models like Variational Autoencoder (VAE) and Generative Adversarial Network (GAN), and with models describing system dynamics like Autoregressive Integrated Moving Average (ARIMA) and Hidden Markov Model (HMM). It is also irrelevant whether such an inverse transformation formally exists. For instance, inverse transformations for many manifold learning / nonlinear dimensionality reduction techniques may not have been proposed or recognized officially. Nonetheless, they are theoretically possible [87].

A counterexample would be indexing techniques like R-Tree and its variants, also commonly used in time series analysis, especially in retrieval tasks [2, 90]. We are skeptical of qualifying

them as time series representations because they may not preserve sufficient information on data characteristics, obstructing a meaningful data reconstruction. Similar to our consideration, Esling et al. treat the time series representation and indexing as two separate topics. Together with the similarity measure, they are the three major issues in time series data mining [85]. However, this is admittedly controversial. For instance, Fu does not differentiate representation and indexing [94], and Koegh explicitly includes trees as a time series representation [138].

Due to the same reason, we refrain from regarding transformations like Box-Cox transform [29], detrending, and differencing as time series representations, which Salles et al. feature in their survey on non-stationary time series transformation methods [230].

Furthermore, a time series representation can

- prescribe a set of operations, as with integral transforms, piecewise representations, and dimensionality reduction techniques, or model the time series, as with stochastic processes and machine learning models;
- be deterministic as with a linear regression model, or stochastic as with a stochastic process model;
- make assumptions about the data like normal distribution, as with SAX [165], and stationarity as with Autoregressive Moving Average (ARMA) [32], or agnostic about the data properties, as with most machine learning models;
- be geared towards certain tasks like ARCH only for predicting variance/heteroscedasticity/volatility [82], or be general-purpose like dimensionality reduction techniques;
- serves in complex pattern discovery or rule mining, or just for data smoothing or compression spontaneously.
- incur consequential overhead like model training, or basically computationally free to use like piecewise representations.

We will comprehensively survey time series representations in Chapter 3 and hereby omit a deeper related work analysis.

Time series representations play a cardinal role in our work. In Chapter 5, we will employ and enhance the time series representation LSH (to be elaborated in Section 5.2) as the foundation for the high efficiency, especially for very high-dimensional time series. In Chapter 6, we will utilize and extend the time series representation SAX to capture pattern distortions in time series.

2.5 TIME SERIES SIMILARITY MEASURES

Measuring similarity (alternatively distance) between two time series is a common subroutine in many time series analysis algorithms. In the task time series pattern search, the rule-based (not involving machine learning models) algorithms heavily rely on a similarity measure to gauge the similarity between a pattern and the query Q. The similarity measure is decisive for retrieval accuracy.

Subsequently, we will define the term "similarity measure" in time series analysis and explain its two forms (rule-based and model-based).

2.5.1 *Definition of Time Series Similarity Measure*

We define a similarity measure as a function

$$f_{sim} : \mathbb{R}^{\phi_1 \times d} \times \mathbb{R}^{\phi_2 \times d} \to [0, 1]$$

which takes two time series (segments) P_1 of length ϕ_1 and P_2 of length ϕ_2 and outputs their similarity. In this thesis, we assume that the similarity score is normalized to $[0, 1]$.

In general, applications favor distance measures (e.g., ED and DTW) over similarity measures (e.g., cosine similarity), especially in rule-based cases. We define a distance measure as

$$f_{dis} : \mathbb{R}^{\phi_1 \times d} \times \mathbb{R}^{\phi_2 \times d} \to [0, +\infty)$$

The conversion between f_{sim} and f_{dis} can be nontrivial. Undoubtedly, (normalized) similarity 1 is equivalent to distance 0. Both of them denote a pair of identical patterns. However, it is controversial what similarity 0 means. For instance, it could correspond to distance $+\infty$, in which case the conversion is nonlinear. We choose to map the highest distance (between the query and the pattern in the time series that is most dissimilar to the query) to similarity 0 and keep the conversion linear. Moreover, we prefer the notion of similarity measure because it aligns well with the output from machine learning classification models used to estimate the relevance of patterns. Nonetheless, as we will mention later, the scores of similarity or distance are secondary; what matters is the ranking of the found patterns. As such, we will use the term "similarity measure" by default and use "distance measure" if the context makes it more appropriate.

As mentioned in Section 2.3, time series pattern search needs a relevance function f_{rel}. This is where f_{sim} enters the data pro-

cessing pipeline for pattern search. Depending on whether a machine learning model is involved, we can divide time series pattern search approaches coarsely into rule-based and model-based techniques. Rule-based pattern search often begins with the single target pattern example, i.e., the query Q. In this case, we can regard f_{rel} as a partial function of f_{sim}, i.e., $f_{rel}(P) = f_{sim}(P, Q)$. Normally, the user does not know the optimal relevance threshold η (equivalently the desired number of top hits) a priori and needs to adjust η after seeing the search results. In contrast, model-based pattern search is often driven by multiple target patterns, either ground truths or labels from user feedback. In this case, f_{rel} takes the form of a model and does not rely on a similarity measure directly, while η is usually 0.5.

As a side note, we call it "measure", instead of "metric", because in mathematics, a metric (in a metric space) has to possess the following three properties:

1. Non-negativity:

$$f_{dis}(P_1, P_2) \geq 0\ ,\ P_1 = P_2 \Leftrightarrow f_{dis}(P_1, P_2) = 0$$

2. Symmetry:

$$f_{dis}(P_1, P_2) = f_{dis}(P_2, P_1)$$

3. Triangle inequality:

$$f_{dis}(P_1, P_2) + f_{dis}(P_2, P_3) \geq f_{dis}(P_1, P_3)$$

Please note that the properties were proposed for distance metrics, not similarity measures. There are issues translating them to similarity measures, especially regarding the triangle inequality. These properties are desirable for indexing and fast querying. Many popular similarity/distance measures do not satisfy these criteria. For instance, one of the most popular distance measure DTW violates the triangle inequality. Nonetheless, we use phrases like "evaluation metric" or "quality metric", e.g., in Chapter 4, because they are widely accepted, and it is clear that the word "metric" in this context does not have the mathematical connotation.

2.5.2 *Rule-Based Similarity Measure*

Rule-based similarity measures do not have trainable parameters in f_{sim}. Common basic options of the rule-based similarity measure include cosine similarity [66], Mahalanobis distance [208],

and variants of the L^p norm, such as the L^2 norm aka ED [36, 188] and the L^∞ norm aka Chebyshev distance [47]. To capture temporally warped patterns, researchers proposed elastic similarity measures, where DTW and its variants [23, 237] are the most widely used and proven hard to beat among 20 methods in the task time series classification [15]. Lines and Bagnall benchmarked eleven rule-based similarity measures, also for time series classification, and found no significant performance differences among the elastic similarity measures [167]. Therefore, we will use DTW if our method requires a rule-based similarity measure, as in Chapter 5; or use DTW to benchmark our rule-based method if our method prescribes another similarity measure or works without an explicit similarity measure, as in Chapter 6. ED, DTW, etc. are distance measures. In our experiments, we invert the distances and normalize them, as indicated earlier. In practice, the absolute value of the similarity is unimportant; what matters is the ranking of the pattern candidates. Hence, any manipulation of the similarity or distance, that does not change the ranking of the pattern candidates, also does not change the performance of the method. For instance, theoretically, we can scale the similarity heterogeneously so that the top hits stand out. This will not affect the pattern-interval-based metrics (Section 4.3) at any rate; also not the time-step-based metrics (Section 4.2) if the threshold v is transformed along with the similarity scores. In Chapter 6, we even eliminate the need for an explicit similarity measure in rule-based pattern search and innovatively use a regular expression to capture similarity in the presence of various distortions.

2.5.3 *Model-Based Similarity Measure*

The notion of similarity is fuzzy and hard to describe concretely and mathematically. Worse still, it can be subjective and differ in each use case, as C9 Subjective and Task-Dependent Similarity and Relevance states. Research has already acknowledged that different use cases favor different similarity measures [15], and "no single algorithm accounts for human judgments of time series similarity" [65]. Rule-based similarity measures lack trainable parameters to model complex similarity notions and adapt to changing use cases. To solve this problem, many works propose machine learning models in place of fixed rules to assess time series similarity, i.e., granting f_{sim} trainable parameters. Recent representative model-based similarity measures include Siamese networks [122, 201] and NeuralWrap [107]. Some time

series representations (Section 2.4) bring their unique similarity measures, like SAX's distance measure between symbols [165] and ShapeSearch's scoring system defined upon their shape primitives [239]. We have not used model-based similarity measures or machine learning models as f_{sim}. Instead, we use models as f_{rel} directly, so that we do not need to input Q every time we use the model, and the model learns from multiple samples, not only Q.

2.6 TIME SERIES TRAVERSING APPROACHES

Time series pattern search algorithms require a traversing approach to scan the time series data. We coined the term "traversing approach" because this problem reminded us of traversing data structures (trees or graphs). This is an essential yet less visited area.

The majority of the methods in the literature scan the time series data sequentially, while some try to improve the accuracy or speed issues associated with this approach. In the following subsections, we will briefly discuss them.

2.6.1 *Sequential Scanning*

The classic traversing approach uses a sliding window to check every temporal position in the time series. At each position, it calculates the relevance of the pattern that starts from the position and spans the length of the sliding window. Faloutsos et al. called this traversing approach "sequential scanning" [90]. It converts the problem to the more studied task time series classification, a common practice among similar works [153, 156, 182].

2.6.2 *Accuracy-Oriented Traversing Approaches*

Before introducing the techniques, let us first discuss two weaknesses of sequential scanning related to accuracy.

Hard-to-Set and Fixed Window Length. Finding the optimal window length is nontrivial. Ermshaus et al. reviewed six algorithms for configuring the window length [83]. However, they only work for anomaly detection, motive discovery, and segmentation. Apart from the issue with the optimal window length, this approach cannot directly retrieve patterns of variable lengths. It relies on a series of sliding windows of different lengths, adding a new dimension of computational complexity and increasing ex-

ecution load multiple times. As a result, the trade-off between accuracy and efficiency becomes acute when the pattern lengths vary significantly. Specifically, adding differently-sized sliding windows increases accuracy at the cost of more processing time, as mentioned in C3 Duration Scaling. In practice, if our proposed algorithm (i.e., Section 5.3.1 and Section 7.3.1) or benchmark algorithms require sequential scanning, we used a series of eight sliding windows of logarithmically increasing lengths to capture patterns with varying duration.

Context-obliviousness. A typical sliding window assesses the relevance of a pattern based on the characteristics of the pattern located in the window. It cannot consider the pattern context. Sometimes, the target event does not manifest itself in rich features within the time period during which it happens but may precede or succeed distinctive signals, of which the user may not be aware. In this case, considering the data surrounding the pattern to examine contributes to the retrieval.

Researchers have proposed various solutions for rule-based algorithms to address the first weakness. For instance, some works match individual pieces in the query with pieces in the time series with the same characteristic (e.g., monotonicity) and a matching pair of pieces may differ in length [136, 182]. It is also possible to formulate the query as a regular expression that scans a symbolic representation of the time series, where quantifiers in the regex can capture pattern scaling along the time axis [219]. There are also works proposing computational techniques [104, 229]. However, these approaches work only with rule-based pattern search. In model-based pattern search, sequential scanning with a sliding window is currently the only option in the literature [156, 275].

The second weakness remains untouched.

Our work in Chapter 6 extends the technique that queries symbolic represented time series with a regular expression, thus inheriting its capability to deal with the first weakness. Moreover, our work in Chapter 7 breaks the yoke that model-based pattern search must use sliding windows to traverse data, addressing both weaknesses without tampering with the models, raising accuracy and speed simultaneously.

2.6.3 *Speed-Oriented Traversing Approaches*

In terms of speed, Mueen's Algorithm for Similarity Search (MASS) is so far the fastest similarity search algorithm [187, 188]. It uses computational techniques to accelerate the calculation of

similarities between all subsequences of a certain length in a time series. Nonetheless, it still suffers from the aforementioned two weaknesses regarding accuracy. We used it constantly for benchmarking speed with our rule-based algorithms. Admittedly, MASS outperforms our methods in speed in general. However, under certain circumstances, like pattern search in very high-dimensional time series (Section 5.5.2) or in the presence of duration scaling among other distortions (Section 6.5.3), our algorithms overtake.

2.7 PATTERN DISTORTIONS AND RETRIEVAL INVARIANCE

Though less considered this way, time series pattern search is, from our perspective, an instance of information retrieval, which addresses the problem of retrieving information in unstructured data like text documents or images according to a search query [43, 56]. In time series pattern search, the query is usually a target example extracted from the data or a prototype of the target drawn by the user (please refer to Section 2.9.2 for more details). The goal is to retrieve patterns in the time series similar to the given query. This implies that the target patterns in time series differ in some aspects. We call these dissimilarities pattern distortions. Algorithms that contrive to retrieve relevant patterns despite these distortions are said to have retrieval invariance [101, 199].

Besides random noise, patterns in the time series exhibit various forms of distortions. MTS inherits all distortions affecting UTS while plagued by unique distortions. As such, subsequent subsections explain distortions in UTS first, followed by that in MTS.

2.7.1 *Pattern Distortions in Univariate Time Series*

Horizontal translation, i.e., earlier arrival or delay of a pattern along the time axis, is trivial in time series, when the de facto standard traversing approach, sequential scanning (Section 2.6), is applied.

Horizontal scaling, i.e., dilation and contraction of the pattern duration along the time axis, is only addressed in a few works, such as [137, 181]. They are tightly bound with certain time series representations or methods that may not collaborate with the techniques that the user wants to use in the first place. As mentioned in Section 2.5, model-based pattern search relies on sequential scanning and does not possess horizontal scaling in-

variance directly. Our work in Chapter 7 solves this problem irrespective of the used model.

Vertical translation and scaling, i.e., bias and amplitude expansion/compression along the value axis, is often handled by normalizing time series windows so that each window spans the value range $[0, 1]$ in case of min-max normalization (sometimes simply called normalization) [156] or has mean 0 and standard deviation 1 in the case of Z-score normalization (also called standardization) [47].

Time warping, which can be regarded as a heterogeneous resample of different fragments of a pattern with unequal scaling factors, is captured by a variety of elastic similarity measures [48, 244], among which DTW is the most popular [23, 237, 284]. In Chapter 6, we propose a unique way to address warping with quantifiers in a regular expression.

2.7.2 *Pattern Distortions in Multivariate Time Series*

In multivariate cases, most methods handle the temporal dynamics and inter-track relationships independently and assume that the tracks are synchronized, i.e., a pattern begins and ends in all tracks simultaneously. Like the cases for general time series analysis mentioned in Section 2.1, in time series pattern search, some methods like DTW_I calculate distance profiles along the time axis for each track individually and merge them subsequently for each time step [237]; other methods like DTW_D and LSH merge tracks first, followed by univariate processing along the time axis [272]. We have not found a time series pattern search method, that naturally integrates the temporal and inter-track information, as with VAE mainly for time series prediction. All approaches that we reviewed and claimed to be suitable in multivariate cases assume that every target pattern is synchronized across all tracks. Driven by our domain problems, especially those related to CAN Bus data, we discard this assumption and scan the temporal dynamics as well as inter-track relations jointly in Chapter 6.

2.8 ACTIVE LEARNING FOR TIME SERIES PATTERN SEARCH

Model-based pattern search requires training machine learning models with sufficient labels. In practice, there are usually insufficient ground truths of the target patterns to use as training labels. To deal with this problem, the technique relevance feedback [226]

from information retrieval and the active learning [216, 247] from machine learning invite users to label the retrieved items interactively, possibly in multiple iterations. During this interaction, domain knowledge flows into the data processing pipeline. Active learning has witnessed success in processing of tabular [20], text [226, 235], and image data [74, 160]. Since its introduction in time series analysis in [142], this technique has gained attention in time series pattern search [156, 232, 275], where researchers advocate the acquisition of labels by inviting users to verify found patterns.

Classic active learning in information retrieval consists of four techniques, i.e., feature extraction, classifier, query strategy, and balance strategy [235]. For instance, [235] shows that a typical constellation for text retrieval may use Term Frequency–Inverse Document Frequency (TF-IDF) for feature extraction [255, 277], Support Vector Machine (SVM) as the classifier [51, 197, 209], (un)certainty-based query strategy [197, 209], and undersampling as the balance strategy [235, 255, 277].

However, we cannot attribute our contributions to these four techniques. Instead, we are more interested in the relatively lower-level prerequisites and machinery of active learning, which has three essential components from our perspective.

Firstly, trainable parameters, such as those in a model, form the foundation. We show in Chapter 5 that parameters can even be as broad as initially random numbers in hash functions that no work has thought of training.

Secondly, it requires understandable features of the outcome as an interface between users and the model. Until now, active learning has only considered the relevance of the found pattern by asking users to accept relevant found patterns and reject irrelevant ones. It is understandable because traditional relevance feedback techniques serve mainly in text and image retrieval, and there are not many understandable features on which users can provide feedback. It remains a question if feedback on relevance alone contains sufficient information for capturing the user's notion of relevance and steering the algorithm to converge to relevant time series patterns. We propose to include other features specific for time series pattern search, namely positions and sizes of the found patterns in Chapter 8, which leads to significant accuracy improvement. Accordingly, we differentiate in this thesis the often interchangeable terms "active learning" and "relevance feedback", i.e., relevance feedback acquires user feedback only on relevance and is a subset of active learning.

Thirdly, active learning needs a mechanism to steer the trainable parameters based on the user feedback to the understandable features of the outcome. Traditionally, this mechanism is based on mathematical optimization, which is inefficient and not interpretable. We show a mechanism in Chapter 5 that works both instantaneously and interpretably.

We call model-based pattern search driven by human-in-the-loop active learning "feedback-driven pattern search". Whereas, general model-based pattern search may begin with sufficient labeled data and may not involve active learning.

2.9 VISUAL QUERY SYSTEMS

Data scientists design time series pattern search algorithms mainly for domain experts, who do not have coding skills and, according to our survey, are also reluctant to learn the algorithm. Therefore, there is a need for tools that execute the developed algorithm and that users interact with. Such tools are called Visual Query Systems (VQSs).

In the following, we will define the term VQS, introduce its dominant categorization, and present representative VQSs from the literature.

2.9.1 *Definition of Visual Query System*

According to [155], the term VQS was first introduced by [227] to refer to "systems that allow users to specify and search for desired line chart patterns via visual interfaces". For clarification, the term Visual Query System (VQS) has been used even earlier, e.g., by [9, 42], but in a broader sense. We have designed a dedicated user interface for each proposed algorithm and delivered them together as a VQS to assist the user in applying the developed algorithm, including functions like query definition, parameter tuning, result inspection, and primitive error diagnosis.

2.9.2 *Categorization of Visual Query System*

Literature categorizes VQSs primarily based on the approach to specifying the query. There are two prevailing approaches, viz. query-by-example and query-by-sketch. The former defines a query by providing an example, e.g., marking an interval in the line chart showing the time series, like in the VQS design pioneer TimeSearcher [119] and the recent PEAX [156]. This query defini-

tion approach is preferable if the query is complex and an example is accessible [156]. On the debit side, query-by-example can be brittle [129]. Query-by-sketch specifies a query by drawing a prototype, e.g., on a canvas. Examples range from QuerySketch [259] over QueryLines [227] to Zenvisage++ [155]. It gives users more freedom, especially when the initial example is hard to find [225]. It is an active research area, as capturing the unbiased concept from the user's drawing is challenging [155]. Other query definition approaches include formulas, regular expressions [239], and natural language [129, 239]. All our VQSs adopt the query-by-example approach because it is too overwhelming for our users to draw or describe in other ways their patterns with complex temporal dynamics, let alone considering multiple tracks and sometimes with time shifts between tracks. On the other hand, our engineers usually can provide at least one example of the target event.

2.9.3 *Pioneering and Recent Visual Query Systems*

We list pioneering (old), recent, and our VQSs in Table 6. This table is based on Table 3 in [275] (PB3) and Table F.1 in [276] (PB5).

Table 6: Recent VQSs

VQS[1]	Year[2]	Query Definition	Representation / Indexing	Similarity Measure / Classifier	MV[3]	FB[4]	VD[5]	CA[6]	Focus / Contributions
TimeSearcher[7] [36, 119]	2004	Example	None	ED	No[8]	No	No	No	VQS design general
MASS[9] [188]	2016	N/A	None	ED	No	No	No	No	Speed
Qetch [182]	2018	Sketch	Trend (up / down)	Own local distortion error and shape error	No	No	**Yes**	No	Query-by-sketch
STSS [219]	2019	Regex	Symbolic representation for value, trend, 2nd order derivative, and value range	regex	No	No	**Yes**	No	Semantic shape description
PEAX [156]	2020	Example	Convolutional autoencoder	Random Forest	No	**Yes**	No	No	time series representation, feedback-driven pattern search
ShapeSearch [239]	2020	Sketch / natural language / regex	Position + trend (up / down / flat / degree) + dedicated operators (concat / and / or)	Scoring system defined on the representation	No	No	No	No	Semantic shape description
				⋮					

Table 6: Recent VQSs – Continued

VQS[1]	Year[2]	Query Definition	Representation / Indexing	Similarity Measure / Classifier	MV[3]	FB[4]	VD[5]	CA[6]	Focus / Contributions
Zenvisage++ [155]	2020	Sketch	None	ED / DTW / Segmentation / MVIP	No	No	No	No	Query-by-sketch
Qute [129]	2021	Natural language	Predefined intuitive features	Sum of feature closeness scores	**Yes**	**Yes**	**Yes**	No	Query by natural language
PSEUDo [275]	2022	Example	LSH	ED / DTW	**Yes**	**Yes**	No	No	Speed for high-dimensional time series retrieval, feedback-driven pattern search
SAXRegEx [276]	2023	Example	SAX	Regex	**Yes**	No	**Yes**	No	(Multivariate) pattern distortions, speed
NOOPS (to be submitted soon)	2025[10]	Example	None	Agnostic (best model in experiments: random forest)	**Yes**	**Yes**	**Yes**	**Yes**	feedback-driven pattern 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 MTS up to eight tracks. However, pattern search is conducted in each track independently. Found patterns are also independently highlighted in each track.

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

[10] `NOOPS` has not been published yet. The publication year is estimated.

We have designed three VQSs, PSEUDo (Chapter 5), SAXRegEx (Chapter 6), and NOOPS (Chapter 7 and Chapter 8). They are tailored to the proposed algorithms. For instance, PSEUDo visualizes the specific hashing-based classifiers to explain its behavior; SAXRegEx supports the definition of multivariate queries with time shifts between tracks, during which users also see the reconstructed query in the symbolic space; NOOPS visualizes the relevance of each time step (important interim results of the data processing pipeline) for error diagnosis and supports feedback on features more than pattern relevance, among other visualizations and interactions for parameter tuning and error diagnosis.

Until now, our work has focused primarily on the algorithmic side, while our VQSs do not intend to make major academic contributions to user interface design. They mainly serve to assist the user in applying the proposed algorithms.

3

TIME SERIES REPRESENTATIONS: A SURVEY

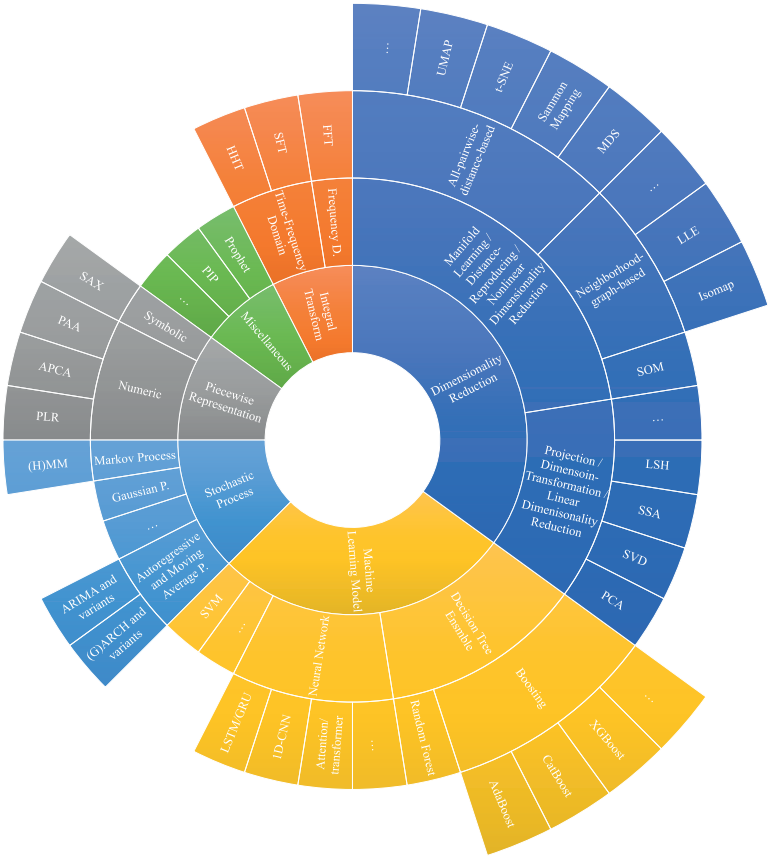


Figure 3.1: **Taxonomy of Time Series Representations.** We categorize time series representations into six categories based on their essential techniques.

In this chapter, we present a taxonomy of time series representations. As far as we know, it is more comprehensive than state-of-the-art surveys on the same topic in terms of academic disciplines (e.g., statistics, signal processing, machine learning, where the methods originate), application domains (e.g., economics/finance, meteorology, engineering, where the methods serve), and tasks (e.g., compression, prediction, anomaly detection, which

the methods solve). The taxonomy categorizes the representations based on essential technical affinities and respects existing disciplines. [1]

3.1 INTRODUCTION

In retrospect, during our four-year research in time series pattern search and time series analysis in general, we have reviewed, analyzed, utilized, and proposed several time series representations. We would like to share our knowledge and experience in the form of a survey on time series representations and discuss their effective choices.

Time series embraces many kinds of data in various domains, as enumerated in Section 1.1. Likewise, perhaps half of the algorithmic advancements in time series analysis emerge from related disciplines, as indicated in Section 2.2.2 However, various domains view time series data from diverse perspectives, make distinctive assumptions, and serve different tasks, which restricts method transfer. Therefore, it is meaningful to survey interdisciplinary time series representations while issuing their caveats. During our literature review for time series representations in Section 2.4, we found that most surveys [72, 230, 250, 264] of time series representations concentrate on one or a selection of disciplines and research fields. Moreover, they emphasize the state-of-the-art analysis while focusing less explicitly on the constraints and limitations of the representations. Lastly, they sometimes present methods as isolated knowledge nodes instead of a graph with clusters of nodes, which we strive to achieve.

Please refer to Section 2.4 for our definition and clarification of the term "time series representation".

In this chapter, we will propose a taxonomy of time series representations with six categories. Table 7 compares the existing reviews with ours. In general, our taxonomy is much broader than the state-of-the-art analyses in terms of the involved disciplines. It also does not focus on one downstream task.

Our review features a unique ontology in the following three aspects.

First, unlike, e.g., [72] primarily for stochastic processes, [264] solely for time-domain representations, or [154, 250] mainly for deep learning methods, we subsumed a broader genre of time

1 This chapter is largely based on our ongoing publication PB7 with textual modifications for a coherent information flow.

series representations and categorized them based on their essential technical affinities and named the categories accordingly.

Second, unlike, e.g., [111] chiefly from the aspect of econometrics, [31] with an emphasis on system and control engineering, [238] from the perspective of statistics, our domain-agnostic work surveyed techniques from various domains, from econometrics and meteorology, over signal processing and control engineering, to data science and computer science.

Third, unlike, e.g., [76] for pattern search, [230] for prediction, or [53] for compression, we reviewed methods for multiple tasks, mainly those explained in Section 2.2.

Publication	Year	Discipline						Downstream Task								
		Stochastic Process	Integral Transform	Piecewise Representation	ML Model	Dimenionality Reduction	Miscellaneous	Pattern Search	Anomaly Detection	Prediction	Motif Discovery	Segmentation	Classification	Clustering	Causality Discovery	Summarization
Koegh [138]	2008		✓	✓		✓		(✓)[1]	(✓)	(✓)	(✓)	(✓)	(✓)	(✓)	(✓)	(✓)
Ding et al. [76]	2008		✓	✓				✓								
Längkvist et al. [154]	2014				✓					✓			✓			
Box et al. [32]	2015	✓							✓	✓						
Wilson [264]	2017			✓			✓[2]	(✓)	(✓)	(✓)	(✓)	(✓)	(✓)	(✓)	(✓)	(✓)
Salles et al. [230]	2019	✓	✓				✓[3]			✓						
Hamilton [111]	2020	✓	✓							✓						
Deistler et al. [72]	2022	✓	✓							✓					✓	
Triat et al. [250]	2024				✓			(✓)	(✓)	(✓)	(✓)	(✓)	(✓)	(✓)	(✓)	(✓)
Our work	2024	✓	✓	✓	✓	✓	✓	(✓)	(✓)	(✓)	(✓)	(✓)	(✓)	(✓)	(✓)	(✓)

[1] (✓) means that the review is agnostic of the downstream tasks and therefore theoretically covers all of them.
[2] PIP.
[3] Diverse simple transformations like logarithmic transform, Box-Cox transform, and simple differencing.
[4] This is a technique paper, instead of a review or monograph. However, it benchmarked many representations and is worth mentioning.
[5] Features extracted with the python library `tsfresh` for time series feature extraction.

Table 7: **Representative Reviews or Monographs on Time Series Representations.** While most works focus on one or two technical disciplines and address a few downstream tasks, our work has a much broader scope.

In this chapter, we present a survey on time series representations in the literature, combining them with our experience. We will begin by explaining our literature selection in Section 3.2. Then, Section 3.3 will introduce our broad taxonomy for time series representations. Next, Section 3.4 will discuss the factors to consider when choosing a time series representation. It follows a comprehensive list of time series representations with their data assumptions, limitations, and typical use cases in Section 3.5. Finally, Section 3.6 will suggest directions for future research.

3.2 LITERATURE SELECTION

This section explains the criteria and queries for our literature selection.

3.2.1 *Inclusion Criteria*

According to the definition in Section 2.4, we include works that convert time series into a representation and solves the downstream task by working on this proxy.

Otherwise, we did not confine:

- **Data Format and Application domain**: it can be stock prices in finance, precipitation in meteorology, genetic sequences in biology, and audio signals in engineering, as explained in Section 2.2.2.
- **Downstream Task**: whether it is anomaly detection, classification, prediction, as presented in Section 2.2.1, or task-agnostic.
- **Publication Year**: because we would like to include classic methods and exploit established knowledge systems; whereas, recent publications are preferred.
- **Venue**: the most prominent are ACM SIGKDD, IEEE ICDM, Springer DMKD, SIAM SDM, and ECML PKDD.

We used Scopus and Web of Science (WoS) core collection as the literature search tools. They provide the world's biggest interdisciplinary scientific citation indexes covering multiple venues.

3.2.2 *Queries*

Our literature selection began with reviews on time series representations, including works on general time series analysis. The

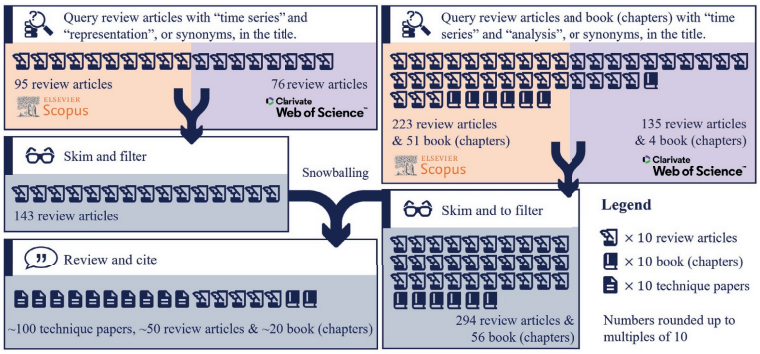


Figure 3.2: **Queries and Results During Literature Selection.** Our literature selection began with review articles on time series representations and works on general time series analysis. Through snowballing, we cover the essential technique papers for individual methods.

technique papers for individual methods proliferate after snowballing, as Figure 3.2 illustrates.

To begin with, we searched for survey papers on time series representations. We started with survey papers to avoid rediscovering existing typologies. Our scope is much broader than other surveys on time series representations. As a result, we would have lost the overview had we begun by reviewing individual technique papers.

In the first query, we searched for survey papers, whose title contains both "time series" and "representation" (or their variants and synonyms). Specifically, the query for Scopus reads "TITLE (time-series AND representation OR transform* OR model) AND (LIMIT-TO (DOCTYPE , "re"))". In WoS, we used the query "TI=(time-series AND (representation OR transform* OR model))" and refined the document type to "Review Article" (WoS does not support specification of document types in the query string directly). In both queries, "time-series" subsumes "time series"; "transform*" also captures "transformation". Scopus and WoS take care of lemmatization like plural forms. Please note that the boolean operator OR precedes AND in Scopus but the other way around in WoS. Scopus retrieved 95 documents and WoS 76.

There are three potential problems with these first queries. Firstly, it is likely that we would overly rely on existing knowledge introduced in the survey papers instead of proposing our own. However, due to our much broader scope, no existing taxonomy exists on this level. In fact, it would be even more effi-

cient and thus preferable to reuse existing reviews or taxonomies as sub-systems in our larger one. Second, survey papers may not cover up-to-date publications and recent advancements. Whereas, we focus primarily on established methods and knowledge systems. Thirdly, this query misses survey papers and monographs on general time series analysis, which also present time series representations. Therefore, we reviewed also publications for general time series analysis with the query "TITLE (time-series AND analysis OR mining) AND (LIMIT-TO (DOCTYPE , "re") OR LIMIT-TO (DOCTYPE , "bk"))" in Scopus and "TI=(time-series AND (analysis OR mining))" (refined to review articles and book chapters) in WoS. They returned 274 and 139 entries, respectively.

After tidying and merging, we manually inspected the documents and selected the ones describing representative time series representations/transformations/models. During the review, we used snowballing to get even more publications, including those introducing one method, but pioneering, impactful, and established.

Finally, we include publications based on our experience gained during the Ph.D. study, such as [44, 156], which benchmarked various important representations while presenting their own.

3.3 A NEW TAXONOMY OF TIME SERIES REPRESENTATIONS

We propose a new taxonomy for time series representations, illustrated in Figure 3.1. Our categories include **stochastic process**, **integral transform**, **pieceswise representation**, **machine learning model**, **dimensionality reduction**, and **miscellaneous**. When introducing the representations in each category, we emphasize the relationships between them to create a system of connected rather than disjoint knowledge nodes. They are categorized according to the dominant fundamental technique. There are two reasons for this choice of classification criterion. First, please consider the alternatives like functions (e.g., classification model, regression model, generation model), tasks (e.g., forecast model, data compression technique, denoising technique), or certain properties (e.g., deterministic/stochastic model, time-domain / frequency-domain representation). A representation may have several functions, serve various tasks, and possess multiple properties. They may have variants and evolve as the technique develops. In contrast, the fundamental technique or mathematical operation for a representation is relatively unique and stable. Consequently, there is less ambiguity when assigning a representation to a cate-

gory. Second, the resulting categories are mostly already treated as established and self-contained disciplines that are studied as individual subjects anyway. There is no need for defining new concepts or explaining extensively. Subsequent divisions can also simply inherit the existing taxonomy in each discipline.

In order to avoid obscuring the main thread of this section, we eschew recent variations based on these root ones because there are overwhelmingly many new but less proven developments. For instance, there are more than 100 extensions of the Generalized Autoregressive Conditional Heteroskedasticity (GARCH) model alone, and none of them seems to dominate. Nor do we claim to be exhaustive in the root representations, because our goal is to establish a taxonomy, into which most time series representations including the unlisted, unnamed, and unborn can fit according to their technical affinity. For instance, the stochastic process representation HMM is omitted. Nonetheless, we try to mention as many influential time series representations. Please refer to Section 3.5 for a comprehensive list of common time series representations.

3.3.1 *Stochastic Process*

Statistics, especially econometrics, often view a time series as the realization of a stochastic process. Roughly speaking, a stochastic process, also known as a random process, is a sequence of random variables, whose index is usually interpreted as time [39, 110].

Introduced in the monumental work from the first edition [30] to the fifth [32], the most influential model in this discipline is ARIMA together with its multivariate extension Vector Autoregressive Integrated Moving Average (VARIMA). In a nutshell, it draws on the information from past data and errors to model the current value, usually in the form of a (possibly high-dimensional) difference equation. ARIMA estimates the mean of time steps and assumes a constant variance for each time step, also called homoscedasticity in econometrics. To estimate a variable variance, also known as heteroscedasticity or volatility in econometrics, [82] introduced Autoregressive Conditional Heteroskedasticity (ARCH) to model current variance with past data, similar to AR, which [27] later extended to GARCH to include past variances and reduce the number of past steps needed. Stochastic volatility models bring it to the next level and assume that the variance itself is randomly distributed [19, 99], which is beneficial when modeling highly dynamic derivative securities like options. The

State Space Models (SSMs) provide a more generalized representation than ARIMA among others, that introduces hidden states in addition to observed data. It allows handling missing data and incorporating time-varying parameters [11, 63]. If quantification of uncertainty is desired, Gaussian Process (GP) is a promising choice.

Most stochastic process representations are used for prediction/regression [32, 77].

3.3.2 *Integral Transform*

Signal processing and control engineering often view a time series as the signals in the time domain and prefer processing it in the frequency domain for more manipulative operations. Integral transform is a mathematical operation that maps a function (in our case a time series) from its original space (usually the time domain) into another space (typically the frequency domain or time-frequency domain) [71], where properties of the system processing these time series can be better described, typically in the context of signal processing and control engineering.

Fourier Transform (FT) with its variants and Wavelet Transform (WT) are the most widely adopted integral transforms in time series analysis. FT uses harmonic waves as the integral kernel and converts the time series into its frequency spectrum, where the manipulations become more straightforward, e.g., for data compression and denoising. FT was invented for continuous functions/data, while time series, as well as digital signals, are discrete in time. Therefore, we need the discrete version of FT called Discrete-Time Fourier Transform (DTFT) [214]. However, DTFT still requires knowing the entire and thus infinite data. In practice, we can assume that the measurements that are limited in time repeat iteratively and arrive at Discrete Fourier Transform (DFT), or its accelerated version FFT [214]. FT assumes a time-invariant frequency spectrum. This assumption does not always hold, and analysts sometimes want to know the temporal context of an event. Short-Time Fourier Transform (STFT) approaches this problem by conducting FT in a sliding window of fixed size along the time axis so that the result lies in a time-frequency domain. Unfortunately, STFT suffers from the mathematically insolvable trade-off between the resolutions in time and in frequency [214]. WT fills this gap by conceptually scanning the data with a scalable and shiftable prototype wavelet that enables multi-resolution analysis in time-frequency domain [217]. While WT relies on the practitioner's expertise in choosing a base/wavelet

function, EMD decomposes a time series into a set of Intrinsic Mode Functions (IMFs) determined in a data-driven way without prior setting of the user [126, 179, 279]. Unlike WT, EMD does not leave the time domain. Hilbert-Huang Transform (HHT) extends EMD with an Hilbert Spectrum Analysis (HSA) on the IMFs to get the amplitudes/energy of various frequency components at each time so that a time-frequency analysis is possible [124, 242].

Most integral transform representations are mainly used for data compression, filtering, denoising, and feature extraction since they originate from signal processing, but they are also seen in other tasks like anomaly detection [62], clustering [80], and pattern search [207].

During our review, we found that the dominating transformation in Digital Signal Processing (DSP), Z-transform, is seldom mentioned in time series analysis. In the future, we expect more applications and development of Z-transform in time series analysis. For example, Z-transform may serve in causality discovery explained in Section 2.2 because it excels at analyzing the relationships between output and input signals through a system. It is also likely that it can serve to check the stationarity of time series required by some stochastic process models like ARIMA and its variants. Because we find that the stationarity check in time series analysis is mathematically similar to stability concerns during digital filter design.

3.3.3 *Piecewise Representation*

Various piecewise representations are inventions of the discipline of time series analysis itself, which remains in the time domain. Piecewise representations assume piecewise homogeneity along the time axis in the data and describe each piece with a simpler representation like a constant, a line segment, a polynomial, and so on.

The simplest piecewise representation is Piecewise Constant Approximation (PCA) [143], later renamed as PAA [140], which uses the mean value to represent each piece. As a kind reminder, down-sampling the data seems to have a similar effect, but it is subject to aliasing effect [34]. The piece length in PCA is fixed. Adaptive Piecewise Constant Approximation (APCA) [140] extends PAA by making the piece length variable and adaptive so that it can fit the temporal shapes better. Instead of mean values, Piecewise Linear Representation (PLR) uses a linear segment to represent each piece. The next logical extrapolation is Piecewise Polynomial Representation (PPR) [98], which uses a polynomial

to represent each piece. Based on PAA, SAX [164, 165] assumes normal distributions of the data values and quantifies them. Then, it assigns each value range bin a symbol to convert a time series track into a string. It soon became one of the most popular symbolic representations for time series and witnessed many extensions [258]. Some surveys on time series representations feature a separate category called "symbolic representation" [264]. However, in our opinion, the dominant technique behind SAX is PAA and quantization, and it is secondary whether the quantified data in each quantization bin are assigning a symbol or left numerical.

Piecewise representations make few assumptions on the data and are general-purpose. They are extremely efficient in that they can compress data massively and have an edge in capturing temporal dynamics [136, 140]. Therefore, piecewise representations can be used all the time, especially when analysts need to smooth the data, remove outliers, or compress the data.

3.3.4 *Machine Learning Model*

According to [149], [Machine learning models are] statistical algorithms that can learn from data and generalize to unseen data, and thus perform tasks without explicit instructions.

Among machine learning models, Recurrent Neural Network (RNN) and its variants like Long Short-Term Memory (LSTM) [120] and Gated Recurrent Unit (GRU) [54] are designed for sequential data like time series. RNN and its variants are typically used for prediction [172, 266, 282], but are also capable of tackling other tasks like classification [128] Otherwise, one-dimensional Convolutional Neural Network (CNN) is an alternative to RNN with considerably less computational costs [55]. In 2017, [252] proposed the transformer model based solely on the attention mechanism without convolution or recurrence, which achieved state-of-the-art performance for language translation. Because the essential concept of transformer/attention also applies to time series data, researchers have tried it on time series analysis like anomaly detection, prediction, and classification [3, 260]. As the representative representation learning model in machine learning, autoencoder [150] learns an identity function with an encoder-decoder structure. The former learns to represent the data with an embedding in a latent space, and the latter tries to recover the data from the embedding. Autoencoder and its variants are typically used for anomaly detection [248, 270] and denoising / missing data imputation [123, 228], though not precluded in other tasks like prediction [194], classification [273],

and pattern search [156] after modifications or with the help of other techniques. Noteworthy is an autoencoder variation called VAE [146], which tries to encode the data as normal distributions in the embedding by learning the means and standard deviations of these normal distributions in its embedding layer (output layer of the encoder or the central hidden layer of the whole autoencoder). Not only are the latent features more explainable, but the VAE becomes particularly suitable for synthetic data generation [75, 112, 159]. It also inherits from the vanilla autoencoder the talent for anomaly detection [49, 161, 271] and is occasionally seen in time series prediction [38, 114]. VAEs have witnessed great success in synthetic image generation especially, but was soon overshadowed by GANs [54]. It features a generator and a discriminator competing with each other during training, after which the former can be used for synthetic data generation and the latter for data recognition. In time series analysis, GANs are used primarily for generating synthetic time series, but also in anomaly detection and denoising [35, 281], like autoencoders.

Machine learning models make few assumptions on the data, but generally require sufficient training data, and even ample labels depending on the task. They often also specialize in certain tasks, as we will discuss in Section 3.4.3. Therefore, choosing a machine learning model usually depends on the task and the availability of ample (labeled) training data.

3.3.5 *Dimensionality Reduction*

Dimensionality reduction techniques transform data points from a high-dimensional space to a lower-dimensional space for revealing patterns in data that are conceived by the curse of dimensionality and for data visualization that are only possible when data points are in a 2D or 3D space [100]. There are two types of dimensionality reduction techniques, namely projection and manifold learning [8, 100] The first class, also known as linear dimensionality reduction, maps the data space linearly to another; while the second, also known as nonlinear dimensionality reduction, endeavors to reproduce the pairwise distances between data items.

The most popular project technique Principal Component Analysis (PCA) [134] reduces and rearranges the coordinate system axes called principal components so that the axes capture as much data variance as possible. PCA has been a popular time series representation in various tasks, including but not limited to prediction [198], segmentation [16], clustering [158],

and pattern search [132]. The mathematically more general technique SVD also has a long history serving in time series analysis [37, 145, 262]. Koegh et al. proposed the first implementation of SVD for time series indexing, where each column in the matrix to decompose is a segment in a univariate time series segmented by a sliding window [140]. Based on SVD and well established in meteorology [224], Singular Spectrum Decomposition (SSA) [224], which decomposes a time series into interpretable components (e.g., trend, periodic components, and noise), has gained much popularity in general time series analysis [28, 113]. The time series representation LSH that we use in Section 5.3 and explain in Section 5.2 is essentially also a linear projection technique that emphasizes efficiency.

Manifold learning techniques, e.g., Multidimensional Scaling (MDS) [170, 215, 254], Isomap [67, 261], Locally Linear Embedding (LLE) [180, 285], Self-Organising Map (SOM) [95, 115], and so on, also contribute actively to time series analysis. Interestingly, t-distributed Stochastic Neighbor Embedding (t-SNE) [117] and Uniform Manifold Approximation and Projection (UMAP) [185], perhaps the most popular dimensionality reduction techniques for data visualization, are less used in time series analysis and their applications are still mostly in visualization [73, 265]. In fact, [156] benchmarked seven representations involving UMAP on their data and conjectured that while useful for visualization, t-SNE and UMAP, are not effective in capturing visual patterns in time series.

Because time series can be multivariate, analysts may apply dimensionality reduction to the temporal domain, possibly piecewise (without overlapping) or with a sliding window (with overlapping) [95], as well as to multiple tracks within each time step [16, 132], or to both simultaneously [268].

As a side note, autoencoders are also good at manifold learning. However, we categorize them as machine learning models because dimensionality reduction is not its single and dominant function. Please also note that while virtually all piecewise representations and Perceptually Important Point (PIP) (Section 3.3.6) claim explicitly that they conduct dimensionality reduction, we only register linear projection or distance-preserving techniques in this category, because, as mentioned at the beginning of this section, our categorization is based on technical affinities, rather than functions, which each technique can serve many.

Dimensionality reduction techniques are relatively general-purpose and serve in many tasks. However, techniques in this category have different mathematical or technical backgrounds,

leading to different data assumptions. For instance, projections like PCA assumes that the linear combination of the axes in the original data space explains data variances, but real data may exhibit nonlinear interrelationships between dimensions. There seems to be no clear hints about how to choose the best dimensionality reduction technique to represent a time series. Therefore, it makes sense to try several and choose the empirically best in specific use cases.

3.3.6 *Miscellaneous*

Lastly, we subsume representations that are hard to categorize, e.g., because they combine techniques from multiple previous categories, under the category "miscellaneous".

One of the most impactful representations in this category is Prophet from Meta (Facebook back then) [246]. It is basically the combination of PLR for describing the trends, Fourier series for capturing the seasonal fluctuations, and a binary "holiday" component accounting for impulses in the data due to short events like big limited-time discounts when modeling sales. As already implied, it is most active for prediction [1, 231].

Another influential representation in this category is PIP [58]. It represents a time series with visually salient points like the major peaks and troughs in the time series curve. PIP centers around the temporal dynamics. It is versatile and finds applications in prediction [251], classification [267], and motif discovery [96].

3.4 FACTORS TO CONSIDER DURING REPRESENTATION SELECTION

According to our literature review and experience, there are four primary factors when considering employing a time series representation: 1) physical dynamics, 2) data assumptions, 3) task suitability, 4) technique transfer, and 5) computational resources. We will explain them individually and provide examples.

3.4.1 *Physical Dynamics*

The first factor suggests that the best data representations ideally model the physical dynamics, e.g., derived from differential equations backed up by physical rules behind the data. This factor is especially relevant for integral transforms (Section 3.3.2).

Nature tends to reward representations that respect the underlying physical dynamics of the observed system. For instance, the physical properties of audio signals encourage the use of FFT because the sound is by nature a linear combination of vibrations of a range of time-invariant frequencies that may trace back to different physical sources. The energy consumption data typically consists of and can be thus represented with ARIMA [30, 32] or Prophet [246] as a trio of 1) a trend implying the technological advances, behavior shifts, and long-term environment changes; 2) a seasonal fluctuation reflecting the cyclic alternations of days and nights as well as the seasons; and 3) a "holiday" component or random residuals capturing unusual events or unexplained errors.

Since physical dynamics are primarily domain-specific, it follows that there may not be a universally (i.e., domain-agnostic) optimal representation for all tasks in time series analysis, just like that no time series similarity measure consistently outperforms the other, as stated in [65] and verified by our experiments in Section 5.5.2 and Section 6.5.2. As [44, 140] show, simple time series representations like PAA may outperform complex ones like FFT, Discrete Wavelet Transform (DWT), and SVD in capturing temporal dynamics / shapes in the time domain in the time series. In these cases, the usage of the two integral transforms is physically not justified and thus not many benefits cannot be expected.

3.4.2 *Data Assumptions*

The second factor requires that the analyst check the fulfillment of the data assumptions made by the representation and make adjustments if necessary. Compared with the first factor, physical dynamics, which entails essential assumptions on the domain-specific physics of the observed system or process, the second factor assumes certain properties of the data itself, e.g., stationarity, linearity, etc. This factor is most prevalent among stochastic process models (Section 3.3.1).

A typical example is when a model type has a parametric variant with a fixed number of parameters and a nonparametric variant that adds parameters as data increase, e.g., linear SVMs vs. nonlinear SVMs, or neural networks vs. GPs [192, 263]. When possible, the former is preferable due to its efficiency and ease of use, while the more computationally expensive latter may be resorted to when the performance of the former is insufficient.

In our work, we encountered the constraint that the values in the time series should be normally distributed, as required by SAX [164, 165]. In essence, SAX's breakpoints for quantization are based on this assumption so that each quantization bin has roughly the same number of data points. Unfortunately, our data deviated much from this assumption. We could have, e.g., simply applied Box-Cox transformation [29] to bring the value distribution closer to a normal distribution. However, considering that the goal is to distribute the data points roughly evenly into the bins, we can arrange the breakpoints according to the real data distribution, as we have proposed in Section 6.3.2.

We extracted eight data assumptions and examined them for each time series representation to create Table 8, a collection of the most representative time series representations in each category of our taxonomy. We select these data assumptions because they are the most common and essential ones influencing the choice and usage of time series representations. These common data assumptions are

STATAIONARITY (When regarding a time series as a stochastic process) the time series has a time-invariant mean, and the autocovariance of the time series depends only on the time lag.

LINEARITY (When regarding a time series as a stochastic process) a value in the time series can be described as a linear combination of values at other time steps plus independent and identically distributed (i.i.d) random variables; (regarding dimensionality reduction techniques) data are explained by a linear combination of latent variables.

MARKOV PROPERTY The value(s) of the next step time step depends only on the values(s) of the current time step. Namely, the observed system is memoryless.

(A)PERIODICITY The values in the time series repeat after a fixed number of time steps (or the time series should have this property).

UNIVARIANCE There is only one track in the time series.

NORMALITY The values, errors, or other components are normally distributed.

KERNEL FUNCTION The time series representation requires a prior choice of a kernel function, which, in turn, assumes certain properties of the time series data.

T+S+X The time series can be decomposed as the sum of trend components, seasonal components, and some other components (noise, residual, holiday, etc.)

Most of the selected common data assumptions are unequivocal. However, the definition of stationarity and linearity requires delineation because they are overloaded with various meanings, especially when crossing the borders of disciplines.

Stationarity

We adopt the weak stationarity stating that the time series has a time-invariant mean and the autocovariance depends only on the time lag [77, 238].

Stationarity is mainly assumed by some stochastic process models. Stochastic processes conceive a time series as one realization of a series of random variables $[S_i \in \mathbb{R}]_{0 \leq i < n}$, where S_i is a random variable. The means and autocovariance refer to the random variables, namely, $\mathrm{E}(S_i) = \mathrm{E}(S_0)$ and $\mathrm{Cov}(S_i, S_{i+\Delta i}) = \mathrm{Cov}(S_0, S_{\Delta i})$. Otherwise, even the notion of "mean" and "variance" does not make sense. Let the time lag be 0, it follows that the variance of the time series should also be time-invariant.

In contract to the weak stationarity, the strict stationarity requires that the joint distribution of the random variables in any same-sized subsequence of the time series to be the same / time-invariant [77, 238]. To be precise,

$$P(S_i \leq c_0, S_{i+1} \leq c_1 \ldots S_{i+\Delta i} \leq c_{\Delta i}) = P(S_{i'} \leq c_0, S_{i'+1} \leq c_1 \ldots S_{i'+\Delta i} \leq c_{\Delta i})$$

where P represents probability, and $c_0, c_1, \ldots, c_{\Delta i} \in \mathbb{R}$ [238]. Strict stationarity is less used since real-world applications do not often see data with this property, and its verification is also costly.

Many stochastic processes requires the data to be stationary. Therefore, it is necessary to conduct statistic tests like the Dickey–Fuller test for AR model to examine the existence of a unit root [238]. In non-stationary cases, the time series must be specially treated to remove the non-stationarity, e.g., through differencing [230].

Please note that the term "stationarity" is overloaded with multiple meanings. For instance, it is common to see the comparison between FT, STFT, WT in the literature, where the first only applies to "stationary" data and the other two to "non-stationary" data [125]. The stationarity here means time-invariant spectrum.

In fact, DFT can be applied to non-stationary (in the sense of our adopted meaning) time series. Consequently, time series representations like DFT are not marked as requiring data stationarity in Table 8.

Linearity

Linearity in this thesis means that the time series representations can only describe linear dynamics or linear relationships between latent factors. This data property is mainly assumed by some stochastic process models and dimensionality reduction techniques.

Linear stochastic process models assume that the random variable at a time step can be expressed by a linear combination of i.i.d random variables, each corresponding to a time step, i.e.,

$$S_i = \sum_{\Delta i=-\infty}^{+\infty} c_{\Delta i} Z_{i-\Delta i}$$

where $[Z_i]$ is zero-mean and i.i.d, the coefficients $c_{\Delta i}$ are square-summable with $c_0 = 1$ [22]. As a typical case, a linear time series is modeled as a linear combination of historical values and sometimes also historical and current innovations (an innovation is the difference between the observed value at a time step and its predicted value), i.e.,

$$S_i = \sum_{\Delta i=1}^{p} \varphi_{\Delta i} S_{i-\Delta i} + \sum_{\Delta i=0}^{q} \gamma_{\Delta i} \varepsilon_{i-\Delta i}$$

where $\varepsilon_{i-\Delta i}$ represents the Δi-th past innovation (ε_i is the current innovation), $\varphi_{\Delta i}$ and $\gamma_{\Delta i}$ are coefficients with $\gamma_0 = 1$, p and q are the order of the autoregressive process and the order of the moving average process, respectively.

Linear dimensionality reduction techniques assume that the time series to represent can be explained by a linear combination of latent components. Linear dimensionality reduction can be regarded as a linear projection of the data from a higher-dimensional space to a lower-dimensional one. However, the data man form complex manifolds that cannot be disentangled linearly from any perspective.

The additive linear decomposition of a time series into trend, seasonal, and other components, as described by the last data assumption, does not automatically count as linear because the determination of these components may describe nonlinear behavior themselves.

Similar to the case with stationarity, many integral transforms, like FT and WT are considered linear. However, this linearity refers to the transform operation $\mathcal{T}$ themselves, i.e.,

$$\mathcal{T}(ax(t) + by(t)) = a\mathcal{T}(x(t)) + b\mathcal{T}(y(t))$$

The linearity of the operation does not imply the linearity of the time series data. In fact, DFT and WT can be applied to nonlinear (in the sense of our adopted meaning) time series. Consequently, time series representations like DFT and WT are not marked as requiring data linearity in Table 8.

3.4.3 *Task Suitability*

The third factor implies that representations may not be general-purpose. Namely, they may (be verified to) perform well in some tasks (Section 2.2) but are unproven or possibly even fail in others. In the latter case, we may not know because the failed results tend to remain undisclosed and potentially found repeatedly by different researchers unaware of each other's work. This factor influences stochastic process models (Section 3.3.1) and machine learning models (Section 3.3.4) most. However, we attribute different reasons to their flair for particular tasks.

For instance, many representations/models emerging from econometrics are geared to prediction, like ARIMA for forecasting the mean, and ARCH for the variance/heteroscedasticity/volatility. They are seldom used in other tasks like classification [257]. Such representation-task-combination could result from historical backgrounds or domain needs and may not reflect the general competence of the representations themselves. It is likely a promising research direction to benchmark the aptness of various representations commonly used for one task also in others, which are conceptually not exclusive to one task. The deliverable can be in the form of a skill matrix, where each column denotes a task and each row a time series representation, which also shows cells of ineffective matches and warns other researchers of potential resource waste in these directions.

Different machine learning models may favor different tasks. Therefore, we mention the suitable tasks for each model in Section 3.3.4. For instance, we mentioned in Section 3.3.4 that autoencoders are typically for anomaly detection and denoising while VAEs for data generation, though not really prohibited in other tasks. In contrast to the stochastic process models, in our view, the reason for the concentrated usage of some machine learning

models in certain tasks lies in the model function and structure itself. We believe that the structure of the model determines its function and ultimately influences its suitable tasks. Pertaining to the same example with autoencoders, its output layer reproduces the data, whose dissimilarity to the original data measured by the reconstruction error reflects the novelty of the input data and thus naturally relates to anomaly detection. The embedding layer of an autoencoder has much fewer neurons than the input, which can be interpreted readily as dimensionality reduction and potentially used for data visualization after further dimensionality reduction into a two-dimensional space if necessary. Should the analyst stick to the model for other tasks due to its certain merits that the alternatives lack, either the task needs to be formulated or the model modified so that they align with each other.

3.4.4 *Technique Transfer*

The penultimate factor is to enable techniques that solve similar problems in another discipline.

For instance, describing time series as a difference equation or state space representation is a common practice in econometrics [111]. They are standard models in signal processing and control engineering and enables analysis akin to techniques in these disciplines, e.g., stability analysis [32].

In Chapter 6, we utilized a symbolic representation, SAX, with nice properties like an ordered alphabet and the numerosity reduction [165], which can be exploited by text retrieval techniques, in this case, regular expression, to describe the distortions in time series patterns.

3.4.5 *Computational Resources*

The last factor considers execution time and memory consumption. It is especially relevant in sensor monitoring and especially IoT, where the data volume is huge. For instance, a Boeing 787 may generate half a terabyte of sensor data per flight [220]; Moreover, data processing is sometimes partially performed on portable devices with limited computing resources. On one hand, this factor concerns the representation itself, especially for machine learning models (Section 3.3.4) when model training should be taken into consideration (Problem 1). On the other hand, analysts may want to accelerate subsequent operations on the representation (Problem 2).

There are three avenues to alleviate this problem. To begin with, many time series representations have parameters controlling the data compression rate, e.g., the resolution of the frequency spectrum for FFT, the number of the states and transitions in a HMM, the number of hidden units in an autoencoder, and so on. This may ameliorate Problem 2. Secondly, there are techniques designed to boost efficiency instead of effectiveness, like various indexing [6, 46, 202]. This may also mitigate Problem 2. Last but not least, dimensionality reduction techniques (Section 3.3.5) and piecewise representations (Section 3.3.3) can compress the data significantly, drastically lower the needs for computational resources. The LSH used in our Chapter 5 into this category. This tackles Problem 2 without introducing Problem 1.

3.5 LIST OF REPRESENTATIVE TIME SERIES REPRESENTATIONS

Based on our literature review and experience, we provide the following table listing the most representative time series representations, their data assumptions/limitations, as well as their typical use cases.

The common data assumptions are explained in Section 3.4.2. We elaborate on them together with unique data assumptions in free-text form next to the common data assumptions.

The use cases reflect classic and established applications, rather than the newest research. We emphasize but do not limit the use cases to time series analysis, because 1) many representations do not have typical use cases in time series analysis, rather, their typical use cases lie in, e.g., image or language processing; 2) listing a broader portfolio of use cases inspires their application in time series analysis since a major portion of methods in time series analysis are transferred from other disciplines themselves, as indicated in Section 2.2.

Due to the space limit, we abbreviated the categories, i.e., SP: stochastic process, IT: integral transform, PR: piecewise representation, ML: machine learning model, DL: dimensionality reduction technique, Mi: miscellaneous.

Table 8: Representative Time Series Representations

Representation	Category	Stationary*	Linear**	Markov	(A)periodic	Univariate	Normal	Kernel	T+S(+X)***	Annotations to Data Assumptions and Other Limitations	Typical Use Cases in Time Series Analysis
Autoregressive Integrated Moving Average (ARIMA)	SP	$\times^1$	$\times^2$		$\times^1$	$\times^3$	$\times^4$			[1] **Stationarity:** constant mean and variance over time after detrending with the integral part, otherwise special treatment for known non-stationarity. For seasonal non-stationarity, use seasonal ARIMA (SARIMA). [2] **Linearity:** relationship between variables and between current value and historical values are linear. [3] **Univariance:** the original ARIMA is developed for univariate time series; use vector ARIMA (VARIMA) for multivariate time series. [4] **Normality:** the error terms in the moving average part are often assumed to be i.i.d samples from a zero-mean normal distribution.	Forecast in economics / finance / business / meteorology
(Generalized) Autoregressive Conditional Heteroskedasticity ((G)ARCH)	SP	$\times^1$	$\times^1$		$\times^1$	$\times^1$	$\times^1$			[1] **Mean Process:** (G)ARCH only models the variances of the error terms. It requires a model / mean process to model the mean, e.g, an AR model. As a result, the assumptions primarily refer to the error terms. **Conditional Heteroskedasticity:** the variances of errors follow an AR model (ARCH) or ARMA model (GARCH).	Forecast of volatility of stock prices

...

Table 8: Representative Time Series Representations – Continued

Representation	Category	Stationary*	Linear**	Markov	(A)periodic	Univariate	Normal	Kernel	T+S(+X)***	Annotations to Data Assumptions and Other Limitations	Typical Use Cases in Time Series Analysis
(Hidden) Markov Model ((H)MM)	SP			$\times^1$						[1] **Markov Property:** the next future state (state is hidden for HMM) depends only on the current state and not on the states that occurred before it. Observation Independence (for HMM): the current observation (time series value) does not depend on previous or future observations but only on the current state. **Finite States and Observations:** the number of states and possible observations are finite. **Time-Invariant Transition Probabilities:** the probability of moving from one state to another does not change over time. It can also be regarded as a kind of stationarity.	Handwriting / speech / guesture recognition, text / audio generation, object tracking, stock price forecast, DNA sequence modeling
State Space Model (SSM)	SP		$\times^1$	$\times^2$						[1] **Linearity:** not strictly required, but it is often assumed that the observed system is linear or can be linearized (and time-invariant, i.e., LTI system), namely, 1) transition equation is linear, i.e., relationship between the state variable in the next future step and the state variable in the current step is linear); 2) observation equation is also linear, i.e., relationship between the observed variable in the current step and state variable in the current step is linear. [2] **Markov Property:** the next future state depends only on the current state and not on the states that occurred before it.	forecast in economics / finance / meteorology, noise filtering in signal processing, sensor fusion (Kalman Filter), system analysis and control in control engineering
...											

Table 8: Representative Time Series Representations – Continued

Representation	Category	Stationary*	Linear**	Markov	(A)periodic	Univariate	Normal	Kernel	T+S(+X)***	Annotations to Data Assumptions and Other Limitations	Typical Use Cases in Time Series Analysis
Gaussian Process (GP)	SP						×[1]	×[2]		[1] **Normality:** values at any subset of time steps follow a multivariate normal distribution. [2] **Kernal Function:** assumptions made when choosing the kernel function / covariance function (how strongly points in the process should correlate with each other)	General regression, prediction, filling gaps in data, bayesian optimization (e.g., hyperparameter optimization in ML), learning control policies, modeling system dynamics
Discrete Fourier Transform (DFT)	IT	[1]			×[2]	×[3]				[1] **Time-Invariant Spectrum:** the frequency components do not change over time. This is why the literature claims DFT requires stationarity. But the column "stationary" refers to time-invariant mean and variance. This is not an assumption of DFT. [2] **Periodicity:** the input signal is periodic and continues indefinitely. The input of DFT is a period. A finite aperiodic time series is thus assumed to repeat itself infinitely. [3] **Univariance:** though no problem with multidimensional data (e.g. images), DFT for multivariate time series is less established, though there are research in this direction, like [206].	Signal filtering, compression, spetrum analysis, system identification, identifying/modeling cyclic patterns
Short-Time Fourier Transform (STFT)	IT					×[1]				[1] **Univariance:** we did not find any multivariate version of STFT. Resolution Trade-Off: time and frequency resolutions cannot be high simultaneously. Larger window size leads to higher frequency resolution but lower time resolution, and vice versa.	Speech recognition, detect and track target in radar/sonar signals, anomaly/novelty detection in machine vibration and seismic data

...

Table 8: Representative Time Series Representations – Continued

Representation	Category	Stationary*	Linear**	Markov	(A)periodic	Univariate	Normal	Kernel	T+S(+X)***	Annotations to Data Assumptions and Other Limitations	Typical Use Cases in Time Series Analysis
Wavelet Transform (WT)	IT					×[1]		×[2]		[1] **Univariance:** though no problem with multidimensional data (e.g. images), WT for multivariate time series is less established, though there are research in this direction, like [162]. [2] **Kernal Function:** assumptions made when choosing the kernel function.	Continuous wavelet transform (CWT): time-frequency analysis Discrete wavelet transform (DWT): compression (e.g. JPEG 2000 and MPEG-4), denoising, signal filtering, feature extraction, pattern recognition
Hilbert-Huang Transform (HHT)	IT					×[1]				[1] **Univariance:** we did not find any multivariate version of HHT. **Sufficient Oscillatiory Behavior:** HHT's first step, Empirical Mode Decomposition (EMD) reuiqres sufficient oscillatory behavior in the time series for the extraction of meaningful intrisic mode functions. It may not work well on monotonic or very smooth data.	Revealing patterns in science (e.g., seismic and meteorological, astronomical data), medical (e.g., anomaly detection ECG and EEG), engineering (e.g., fault analysis for revolving machine, e.g., for bearing); as preprocessing step for prediction in financial data; image enhancement
Piecewise Aggregate Approximation (PAA)	PR					×[1]				[1] **Univariance:** PAA can be applied to each track individually. However, the interrelationships between tracks are not considered.	Time series smoothing, compression, and indexing
						...					

Table 8: Representative Time Series Representations – Continued

Representation	Category	Stationary*	Linear**	Markov	(A)periodic	Univariate	Normal	Kernel	T+S(+X)***	Annotations to Data Assumptions and Other Limitations	Typical Use Cases in Time Series Analysis
Adaptive Piecewise Constant Representation (APCA)	PR					×[1]				[1] **Univariance:** same as above.	Time series smoothing, compression, and indexing
Piecewise Linear Representation (PLR)	PR					×[1]				[1] **Univariance:** same as above.	Time series smoothing, compression, and indexing
Piecewise Polynomial Representation (PPR)	PR					×[1]				[1] **Univariance:** same as above.	Time series smoothing, compression, and indexing
Symbolic Aggregate approXimation (SAX)	PR					×[1]	×[2]			[1] **Univariance:** same as above. [2] **Normality:** SAX assumes that the values in the time series follows a normal distribution.	Time series smoothing, compression, and indexing
Recurrent Neural Network (RNN)	MI									No noteworthy assumptions	General time series prediciton, text/audio generation, speed recognition, anomaly detection, video description

...

Table 8: Representative Time Series Representations – Continued

Representation	Category	Stationary*	Linear**	Markov	(A)periodic	Univariate	Normal	Kernel	T+S(+X)***	Annotations to Data Assumptions and Other Limitations	Typical Use Cases in Time Series Analysis
One-Dimensional Convolutional Neural Network (1D-CNN)	ML									No noteworthy assumptions	General time series prediciton, audio recognition, text embedding generation, anomaly detection
Attention	MI									No noteworthy assumptions	Text / speech / image translation / recognition / summerization / captioning
Autoencoder	ML									No noteworthy assumptions	Anomaly detection, denoising, dimensionality reduction
Variational Autoencoder (VAE)	MI									No noteworthy assumptions	Text / sequence (e.g., music, molecular structure) / image generation, anomaly detection, image denoising
Generative Adverserial Network (GAN)	ML									No noteworthy assumptions	Image / sequence(e.g. molecular structure) generation, super-resolutioning images, image/text-to-image, style transfer, data augmentation, anomaly detection
										...	

Table 8: Representative Time Series Representations – Continued

Representation	Category	Stationary*	Linear**	Markov	(A)periodic	Univariate	Normal	Kernel	T+S(+X)***	Annotations to Data Assumptions and Other Limitations	Typical Use Cases in Time Series Analysis
Principal Component Analysis (PCA)	DL		×[1]							[1] **Lineariry:** the majority of the data variance can be explained by the linear combination of the first few principal components.	Data visualization, compression, denoising, feature extraction
Singular Value Decomposition (SVD)	DL		×[1]			[2]				[1] **Lineariry:** the data can be explained by the linear combination of left sigular vectors that are scaled by singular values and weighted by right sigular vectors. [2] **Multivariance:** according to the SVD implementation from Koegh et al for time series indexing, each column in the matrix to decompose is a segment in a univariate time series segmented by a sliding window [140] Namely, the matrix to decompose is the transpose of the trajectory matrix of the time series. Accordingly, it only works for univariate time series. Nonetheless, we believe one can 1) use the flattened sgements as the columns of the matrix to decompose in multivariate cases; or 2) use the whole multivariate time series directly as the matrix to decompose.	Latent semantic analysis, data compression, denoising
Singular Spectrum Analysis (SSA)	DL		×[1]						×[2]	[1] **Lineariry:** since SSA uses SVD as the first step, the linearity assumption from SVD should still apply. [2] **Decomposition:** the time series consists of trend, periodic compoents, and noise	Forecast in meteorology, economics, finance, analyzing population dynamics in ecology

...

Table 8: Representative Time Series Representations – Continued

Representation	Category	Stationary*	Linear**	Markov	(A)periodic	Univariate	Normal	Kernel	T+S(+X)***	Annotations to Data Assumptions and Other Limitations	Typical Use Cases in Time Series Analysis
Locality-Sensitive Hashihng (LSH)	DL		×[1]			[2]				[1] **Lineariry:** LSH linearly maps multivaiate time steps into univariate scalars [2] **Multivariance:** the implementation proposed in [272] reduces the dimensions (number of tracks) for each time step. Consequently, it only works for very high-dimensional multivariate time series and not for univariate time series. However, we believe that it should also work on segments of univariate time series (trajectory matrix of the time series).	Nearst neighbor search, near duplicate search, fingerprinting
Various Manifold learning techniques	DL									**Single Manifold:** the high-dimensional data lies on or near a lower-dimensional manifold embedded within the higher-dimensional space; and data reside in a single manifold. **Smoothness:** small changes in the high-dimensional data should only result in small changes in the low-dimensional representation. **Uniform sample density:** uniform distribution of points along the manifold, rather than dense in some regions and sparse in other. Individual techniques may have their own data assumptions, for instance, Local Linear Embedding (LLE) assumes local linearity, isomap assumes isotropy (properties change in one direction on the manifold is the same as in any other direction), and Uniform Manifold Approximation and Projection (UMAP) assumes uniformly distributed data on a locally connected Riemannian manifold and that the Riemannian metric is locally constant or approximately locally constant	Data visualization, denoising, feature extraction, anomaly / novelty detection, pattern recognition
Prophet	Mi								×[1]	[1] **Decomposition:** a piecewise linear represented trend, seasonalities described by a Fourier series, and user provided holidays.	Forecast In business, finance, economics, energy consumption, meteorology
						...					

Table 8: Representative Time Series Representations – Continued

Representation	Category	Stationary*	Linear**	Markov	(A)periodic	Univariate	Normal	Kernel	T+S(+X)***	Annotations to Data Assumptions and Other Limitations	Typical Use Cases in Time Series Analysis
Perceptually Important Point (PIP)	Mi					×[1]				[1] **Univariance:** we did not find any multivariate version of PIP. **Key Points:** there exist "key" points in the time series data that captures most of its significant characteristics.	Time series summarization in meteorology and IoT, pattern recognition (e.g., "head-and-shoulders" and "'cup-and-handle") in finanal maket analysis and healthy monitoring, anomaly detection

Table Footnotes

* "Stationarity" in this table means weak stationarity, namely time-invariant mean, and auto-covariance only depends on the time lag, as delineated in Section 3.4.2.

** "Linearity" in this table means that the values in the time series (e.g., (multivariate) value of a time step, values in a time interval, or even multivariate values in a time interval) are assumed to be able to be expressed by a linear combination of other values in the time series or some latent components. It does not mean that the transformation operation is linear, e.g., in the case of DFT, which is rather a desired property, instead of a data assumption / limitation.

*** "T+S(+X)" in this table means that the time series is assumed to be able to be decomposed to trend components, seasonal/cyclic/periodic components, and possibly other components (noise/residual/holiday).

3.6 DISCUSSION

The topic of this thesis is MTS pattern search. However, we do not confine the scope of this chapter to this task alone. Instead, we surveyed as broadly as possible disciplines and tasks regarding general time series representations deliberately. On the one hand, we would like to share our experience that exceeds this task. On the other hand, it is beneficial to have a toolbox of diverse methods originally designed for various tasks. As Section 2.2 mentioned, the majority of methods in time series analysis are "not invented here".

Our taxonomy claims to reflect and respect the essential technical affinity and mathematical operations of the time series representations. Ironically, in this chapter, we have not used many formulas. However, mathematical descriptions would probably more than double the number of pages in this chapter. Had more space, we would especially like to mathematically describe stochastic processes, integral transforms, and piecewise representations. Because the formulas also reflect the advancements of the methods in each of these three categories that we described in Section 3.3 in natural language.

We include a category called "miscellaneous" accommodating time series representations that do not fit into other categories. It implies that the other categories fail to cover all time series representations. This is especially the case when 1) some representations like Prophet are ensembles of equally important components from multiple categories in the taxonomy; 2) some representations like PIP are unique, and it is trivial to define a singleton category for it. Moving forward, new representations may derive from such methods to form new categories.

There is some ambiguity concerning the categorization of some models. For instance, we categorize GP under stochastic processes. However, it is also a valid machine learning model. In fact, we are not entirely satisfied with the category machine learning models because it fails to address technical essentials directly. Nonetheless, we could not find a better way to incorporate general-purpose models like Autoencoder, CNN, and LSTM.

Some time series representations in our taxonomy solve the task directly, e.g., LSTM for time series prediction, while others are essentially preprocessing, whose output undergoes another method dedicated to the task. For instance, `SAXRegEx` in Chapter 6 uses SAX as the data representation. SAX does not retrieve the patterns, which is the responsibility of regular expression. Although this could be controversial, we do not consider it a prob-

lem for such representations bundled with certain functions, as also mentioned in Section 2.4. Please consider the example of large language models. They are widely recognized as representations of languages. However, they are technically prediction models, whose basic job is to fill missing words in sentences. Whereas, they can be adapted to hold conversations.

Finally, while this chapter has reviewed many time series representations and introduced their typical use cases, admittedly, practitioners may still feel overwhelmed by the myriad representations at their disposal. For instance, should the user prefer ARIMA, LSTM, or Prophet for time series prediction? The user can only try a limited number of them. It would be much more helpful if there are a set of rules for choosing the best representation, ideally in the form of a decision tree, but each node may have more than two children. Moreover, we envision this decision tree to begin with the task selection (pattern search, anomaly detection, etc.) in the root node. Because the task (explained in Section 2.2) is usually clear to the user and influences the choice of representations greatly.

3.7 CONCLUSION

In this chapter, we surveyed time series representations systematically. We created a taxonomy of time series representations based on their essential technical affinity, analyzed the factors to consider when choosing a time series representation, and presented a list of common time series representations with their assumptions/limitations as well as typical use cases. Interestingly, we found a divergence of time series pattern search research in statistics/econometrics favoring stochastic processes and machine learning. We believe that they are complementary, and we are eager to see their reconciliation in research. Our taxonomy should help data scientists and others working on time series analysis choose an effective data representation that solves or simplifies their problems. While such guidance does exist, it tends to focus on a narrower and sometimes domain-specific perspective. Our taxonomy bridges the gap between time series representations from diverse disciplines for various tasks. In prospect, we can expect the enrichment of each category in our taxonomy and even the formation of new categories. In the future, we would like to examine and propose a set of practical criteria and rules for practitioners to choose the optimal time series representations for their use cases.

4

EVALUATION METRICS FOR TIME SERIES PATTERN SEARCH

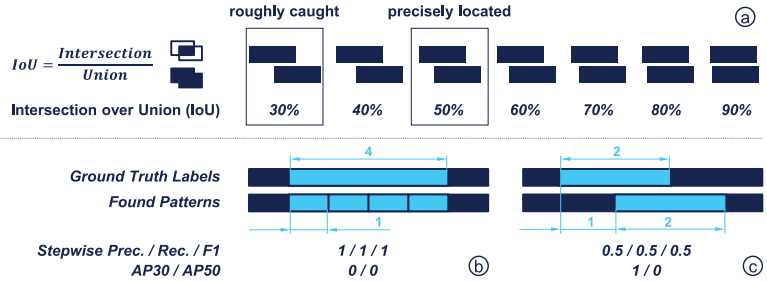


Figure 4.1: **Time-Step-Based vs. Pattern-Interval-Based Metrics.** We propose two types of accuracy metrics for evaluating the performance of time series pattern search algorithms. The time-step-based metrics are easier to understand and use; while the pattern-interval-based metrics align better with the goal of the task. In some cases, the found patterns do not match the target patterns well, yet the time steps are all correctly classified, as in ⓑ. In this case, the pattern search algorithm fails to recognize a target pattern as a whole piece. The time-step-based metrics will arrive at high scores despite this problem. Pattern-interval-based metrics punish such errors.

We would like to preface the exposition of our work on devising new techniques by introducing our evaluation metric. There are two reasons for this unconventional arrangement. For one thing, it is the major evaluation metric in Chapter 5, Chapter 6, Chapter 7, and Chapter 8. Whereas, we cannot explain the metric in Chapter 2 because we are the first to transfer it from object detection in computer vision to pattern search in time series. For another, pedagogically, it may evoke an intuitive feeling of what time series pattern search outputs, and what a good retrieval looks like. It primes the reader with our major work. [1]

1 This chapter is based on the evaluation metrics used in our (released or ongoing) publications PB1-PB6 with textual modifications for a coherent information flow.

4.1 INTRODUCTION

Interestingly, although researchers have proposed numerous algorithms and tools for time series pattern search, a unified evaluation approach is missing. In particular, it is imperative to establish an accuracy metric on the quality of the found patterns.

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

To be rigorous, we differentiate the notion of ground truths $\mathbb{G}$ from target patterns $\mathbb{T}$. The ground truths are labels provided by domain experts according to empirical evidence. A time series may contain target patterns, regardless of whether ground truths are provided. In practice, we assume $\mathbb{G} = \mathbb{T}$, if our engineers claim that they have labeled the whole dataset.

It faces two challenges. For one thing, there are few labeled public datasets for time series pattern search. So, G is often unavailable. Luckily, this problem does not affect our case. Nonetheless, we would like to mention this challenge that exists in general. For another, the matching degree of G and F is not straightforward. Unlike classic classification, the correctness of the found patterns is not black and white. We cannot simply subtract the outcome F from the ground truth G to measure their difference, as in the case of regression / time series prediction; nor can we directly apply standard classification metrics like precision, recall, or confusion matrix.

In the case of the first challenge, researchers resort to visual inspection of the found patterns [2, 92, 136, 156]. Another possibility is to measure performance decrease using found patterns from a more accurate baseline method, kind of as pseudo-ground-truths [104, 239, 272], which is suitable when the proposed algorithm is an accelerated version of the baseline method. Fortunately, our engineers have provided ground truths, so, the first challenge does not stand in our way. However, we still need to face the second challenge.

Our solutions embrace the classic classification metrics like precision and recall. However, they necessitate the nontrivial definition of true positives, false positives, and false negatives. We view this problem (measuring the aligning between $\mathbb{F}$ and $\mathbb{G}$) from two perspectives and propose two classes of metrics, namely time-step-based metrics and pattern-interval-based metrics. The former is more straightforward, while the latter aligns better with

the alignment problem. The subsequent two sections will elaborate on the two classes of metrics.

4.2 TIME-STEP-BASED METRICS

The time-step-based metrics formulate the problem as the binary classification of each time step, i.e., whether the time step is inside or outside a target pattern. Specifically, we first need to derive whether each time step in S is located in a found pattern, i.e.,

$$\boldsymbol{y} = \left[y_i = \begin{cases} 1, & \text{if } \alpha \leq i < \beta,\ S(\alpha, \beta) \in \mathbb{F} \\ 0, & \text{otherwise} \end{cases} \right]_{0 \leq i < n}$$

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

$$\boldsymbol{z} = \left[z_i = \begin{cases} 1, & \text{if } \alpha \leq i < \beta,\ S(\alpha, \beta) \in \mathbb{G} \\ 0, & \text{otherwise} \end{cases} \right]_{0 \leq i < n}$$

Moreover, we can count

- the number of true positives defined as time steps both in a ground truth and a found pattern

$$TP = |\{i \mid y_i = z_i = 1\}|$$

- the number of false positives defined as time steps in a found pattern but not in a ground truth

$$FP = |\{i \mid y_i = 1,\ z_i = 0\}|$$

- the number of false negatives defined as time steps in a ground truth but not in a found pattern

$$FN = |\{i \mid y_i = 0,\ z_i = 1\}|$$

where the operator $|.|$ returns the cardinality of a set. Finally, we can calculate

$$precision = \frac{TP}{TP + FP}$$

$$recall = \frac{TP}{TP + FN}$$

$$F1 = \frac{2 \cdot precision \cdot recall}{precision + recall}$$

Pattern search algorithms usually return multiple found pattern candidates with their relevance. To obtain the actual found patterns $\mathbb{F}$ required by the time-step-based metrics, the user has to provide a relevance threshold η. Only found patterns candidates with relevance higher than this threshold are recognized as found patterns. Alternatively, the user may specify the desired number of found patterns. This configuration of the strictness of the metric is nontrivial because it depends on the data and task, which is usually only possible after the user has inspected the found patterns. It has a major impact on the scores of time-step-based metrics and must be configured sensibly. In our experiments, we tuned the relevance threshold to achieve the best possible F1-score.

Asides from the above inconvenience, time-step-based metrics may lead to the pitfall illustrated in Figure 4.1 ⓑ, where the algorithm fails to recognize the target pattern as a single piece, despite the correct classification of every time step and perfect scores according to the metrics.

4.3 PATTERN-INTERVAL-BASED METRICS

As mentioned in Section 2.2, time series analysis heavily draws on expertise from other disciplines. To solve the problems of the time-step-based metrics, we propose to transfer the metric Average Precision (AP) from object detection in computer vision to pattern search in time series analysis in favor of a perspective of the intervals.

This metric requires a threshold v for IoU between a found pattern F and a ground truth G for judging whether F is relevant or a true positive, based on the ground truth G^* that overlaps (the most among all ground truths $\mathbb{G}$) with it. This IoU of two patterns P_1 and P_2 is defined as

$$IoU\left(P_1, P_2\right) = \frac{|P_1 \cap P_2|}{|P_1 \cup P_2|}$$

The metric AP pronounces a found pattern F^* true positive if

- F^* overlaps with G^* by at least v, i.e., $IoU\left(F^*, G^*\right) \geq v$; and
- the ground truth G^* does not overlap with another found pattern F' of higher relevance by at least v (otherwise, the more confident F' has already consumed G^*), i.e., $\forall F' \in \mathbb{F}$ such that $IoU\left(F', G^*\right) \geq v$, it follows $f_{rel}\left(F'\right) < f_{rel}\left(F^*\right)$.

The IoU threshold v can regulate the strictness of the metric. The choice of v not standardized. In computer vision, v for AP is usually at least 50% and can be as high as 90%. For instance, PASCAL challenge chooses 50% and COCO challenge a range from 50% to 95% with 5% increment. However, we found the degree of overlapping between F and G in time series pattern search, at least in our cases, is generally much smaller than in the cases in computer vision. Accordingly, these thresholds are too strict for time series pattern search, and the scores of most results (including and especially results from baseline methods) would be too low and not well separated in $[0, 1]$, making method comparison difficult. Therefore, We decided to use primarily 30% and 50% and denote the metrics AP30 as well as AP50, respectively. Moreover, we interpret the more lenient metric AP30 as "roughly caught" and the stricter AP50 "precisely located". Figure 4.1 ⓐ shows a series of IoUs. In the example Figure 4.1 ⓒ, 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. Figure 4.2 further clarifies the data science terms true positive, false positive, and false negative in our special problem, time series pattern search.

Please also note that AP does not necessarily need the found patterns, which survive the filtering with a relevance threshold η. Instead, it requires the ranked pattern candidates before filtering. Pattern candidates of low quality but also low relevance will not pollute the high-quality candidates or diminish AP. This is desirable because the decision on η or the number of top hits is not necessarily a responsibility of the pattern search method, especially for rule-based pattern search, where the user needs to adjust η upon seeing the tentative result. Thus, η is better absent from the metric gauging the performance of the methods.

In general, pattern-interval-based evaluation metrics better respect the desired outcome format of the task time series pattern search. Namely, the outcome should be found patterns in whole pieces rather than individual time steps. Pattern-interval-based evaluation metrics also satisfy the target of the evaluation metric defined in Section 4.1 (measuring the aligning between $\mathbb{F}$ and $\mathbb{G}$). In practice, we talk about correct/relevant found patterns, incorrect/irrelevant found patterns, and missing patterns heuristically by measuring their overlapping. Accordingly, the definitions of true positives, false positives, and false negatives for pattern-interval-based metrics match these heuristic notions. We omit perhaps the more useful domain-specific definitions of rele-

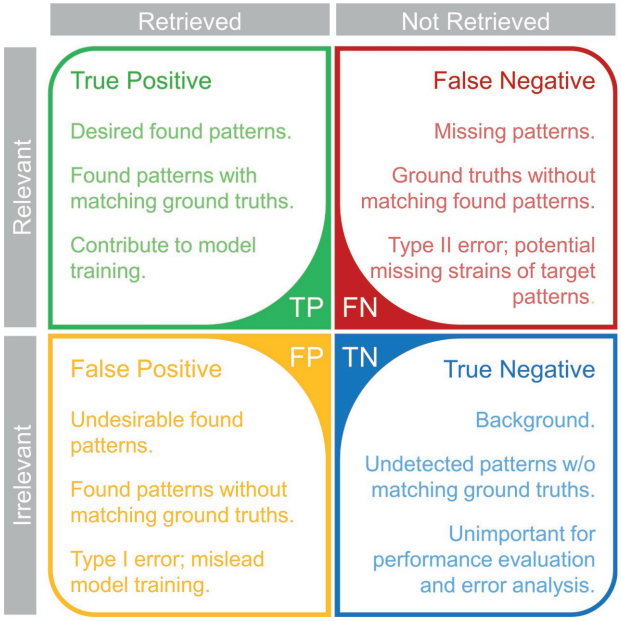


Figure 4.2: **True/False Positive/Negatives.** pattern-interval-based metrics define true positives as found patterns with matching ground truths; false positives as found patterns without matching found patterns; and false negatives as ground truths without matching found patterns. Compared with comparable definitions with time-step-based metrics, they align better with the notion of desired found patterns for true positives, undesirable found patterns for false positives, and missing patterns for false negatives.

vance because it is often unavailable. Otherwise, we prefer to use it as the similarity measure to conduct the search.

Paradoxically, we will criticize pattern-interval-based classification and advocate time-step-based classification when using in model-based time series pattern search later in Chapter 7. Whereas, for evaluation, we prefer a pattern-interval-based metric to measure retrieval accuracy, as explained in this chapter.

5

EFFICIENT, STEERABLE, AND INTERPRETABLE DATA REPRESENTATION FOR HIGH-DIMENSIONAL TIME SERIES

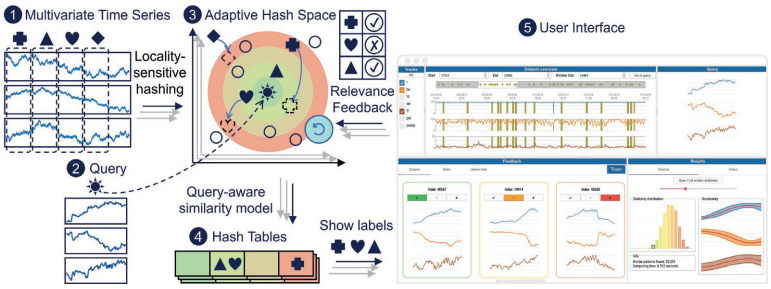


Figure 5.1: **PSEUDo.** We present PSEUDo for pattern search in very high-dimensional time series. It creates a representation model for MTS based on locality-sensitive hashing, conducts scalable pattern retrieval with few initial labels, and evolves with interpretable relevance feedback to capture subjective pattern similarity.

In this chapter, we present PSEUDo. It features a speed-oriented hashing-based time series representation. We designed it to be adaptive and steerable. Besides improving the retrieval, it provides the added benefits of estimating track importance. PSEUDo is especially efficient for very high-dimensional time series. [1]

5.1 INTRODUCTION

Very high-dimensional (e.g., with more than ten tracks) time series emerge on an unprecedented scale due to increasing sensor usage and data storage. Visual pattern search is one of the most frequent tasks on them. The general problem is well understood and studied. However, very few solutions focus on the practical, real-life challenge that oftentimes multiple sensors or observation angles have to be taken into consideration in unison to understand a problem. In accordance, we formulate the research question for this chapter: **how to retrieve patterns in very high-**

1 This chapter is based on our publications PB1, PB2 (awarded Best Paper in VDS 2022), and PB3 with textual modifications for a coherent information flow.

dimensional MTS? Moreover, driven by our domain requirement R4 Explainability, we propose an additional research question: **how to assess the importance of dimensions/tracks?**

On one hand, this problem is interesting and important, because relevant information is often distributed in multiple attributes. In extreme cases, our ECU data contains around 10 thousand tracks, though our engineers can often select 10 to 20 relevant ones before analysis. They also wish to know the feature/track importance, as mentioned in R4 Explainability. On the other hand, this problem is challenging, because high dimensions mean multiple times of data volume for efficient processing, as well as the challenge of interpreting distributed information. In general, automatic pattern retrieval methods suffer from efficient and transparent training algorithms, as mentioned in Section 1.2.

In this chapter, we propose PSEUDo (Pattern Search, Exploration and Understanding in Multivariate Time Series Data), a tool for visual pattern retrieval in MTS, especially very high-dimensional time series. It aims to overcome the uneconomic and uninterpretable (re-)training problem accompanying deep-learning-based methods. PSEUDo is based on Query-Aware Locality-Sensitive Hashing (QALSH) [272] to create a representation of the MTS. In a nutshell, LSH linearly maps all tracks into one with groups of hash functions, achieving sublinear training and inference time with respect to data dimensions. This performance gain allows an instantaneous relevance-feedback-driven adaption to converge to users' relevance/similarity notion. Our major contribution is making LSH trainable and extending it with an efficient, steerable, and interpretable relevance feedback mechanism. We have also implemented a prototypical user interface to assist users with the algorithm.

As shown in Figure 5.1, the overall pipeline works as follows: Stage ❶ in Figure 5.1: preprocessing the time series with sliding windows and window normalization; Stage ❷: marking by the user a pattern in the time series as the query to search for Stage ❸: initial search based on the standard LSH; Stage ❹: sampling results for relevance feedback; Stage ❺: inspecting results and providing relevance feedback by the user; back to Stage ❸: updating the LSH model and rerunning search. This process iterates between Stage ❸ and Stage ❺ until the user is satisfied with the result.

We evaluated PSEUDo's performance in terms of accuracy, speed, and steerability, through quantitative benchmarks with representative time series retrieval methods, and its usability through a case study. Firstly, we benchmarked PSEUDo's "open-

loop" accuracy and speed without relevance feedback against four representative benchmarking methods on four labeled datasets with different characteristics. Disappointingly, we found that the methods worked differently on different datasets, and there was no universally best method in terms of accuracy. In the speed benchmark, PSEUDo was slightly worse than MASS (the state-of-the-art algorithm for time series pattern search in terms of speed) on univariate and low-dimensional datasets. Still, it started to overtake the latter on a relatively high-dimensional dataset. To verify this finding, we benchmarked the scalability with respect to the number of tracks with an unlabeled very high-dimensional dataset. Next, we evaluated the effectiveness of the relevance feedback mechanism, where we used an agent simulating sensible user feedback as the experimental group and two agents simulating extreme user feedback as control groups. We witnessed an increasing accuracy within five feedback rounds, which happened instantaneously. Furthermore, we noticed that PSEUDo attaches higher weights to informative tracks and lower to less helpful ones, confirming its feature selection mechanism. Finally, we invited an expert to test PSEUDo in his use case. In conclusion, PSEUDo detected patterns in high-dimensional time series efficiently, improved the result with relevance feedback through feature selection, and allowed an understandable as well as user-friendly retrieval process.

5.2 RELATED WORK

This chapter centers around an active-learning-based time series representation. We have elaborated on time series representations in Section 2.4 and active learning in Section 2.8. In this chapter, we use Locality-Sensitive Hashing (LSH) as the time series representation due to its high efficiency when dealing with very high-dimensional time series facing C2 High Dimensions. Subsequently, we will briefly introduce this base technique.

Generally, hashing-based algorithms aim for a significant speed boost with tolerable information loss. Classic hashing algorithms highlight the smallest differences in objects, e.g., for data integrity check [52]. First introduced in [130], LSH inherits the efficiency while differing from other hashing algorithms in that it maps similar objects to close hash codes and induce hash collision. This feature enables its application in a wide range of data mining problems, such as nearest neighbor search [127, 130], hierarchical clustering [61, 148], and near-duplicate detection [68]. Conceptually, LSH conceives data objects in the database as points

in a high-dimensional space. This space is cut into sub-regions by a group of random hyperplanes. Each hyperplane is described by a hash function [130]. Each sub-region corresponds to a bucket in the hash table, and the object points in the same sub-region fall into the same bucket in the hash table. To reduce false negatives, multiple hyperplane groups or multiple hash function groups are used. All objects colliding with the query under at least one hash function group are treated as candidates. Subsequently, the candidates can be filtered with a similarity measure to suppress false positives. Since LSH is considerably fast, it can be repeated for every new query with query-centric bucketing, namely QALSH, yielding better accuracy[127]. Recently, LSH received attention for time series indexing, where it achieves up to 20 times faster processing speeds with a minor accuracy sacrifice [176]. Later, the query-centric QALSH is introduced for MTS [272].

Our work in this chapter is based on QALSH (hereinafter called LSH for brevity). We extend the idea by making LSH trainable and explainable while maintaining its high efficiency for processing very high-dimensional time series even with iterative model retaining.

5.3 METHOD

This section describes PSEUDo's relevance-feedback-driven learning algorithm based on LSH. LSH allows querying large MTS efficiently for real-time user interaction. We extended it with an also efficient and interpretable relevance feedback algorithm.

We derive our conceptual model, depicted in Figure 5.2, from FDive, the feedback-driven preference learning method [20]. Inspired by the conceptual model, we assume that the randomly initialized parameters in the LSH hash functions are trainable to combat the ambiguity of time series similarity.

In the conceptual model, input MTS data are modeled with LSH functions and stored in query-aware hash tables, significantly speeding up the processing. Striving for a sensible data modeling on top of LSH's probabilistic nature and converging to the subjective similarity, we draw a sample set from the hashed time series. The user then checks the samples as well as their "average" shape and variance per bucket, i.e., how much uncertainty/volatility the time series windows in a hash bucket expose. This mechanism actively contributes to the querying process in that PSEUDo learns to understand the user's similarity notion and interprets it as feature/track importance. Due to the size and complexity of MTS data, such an open and adaptable exploration process was

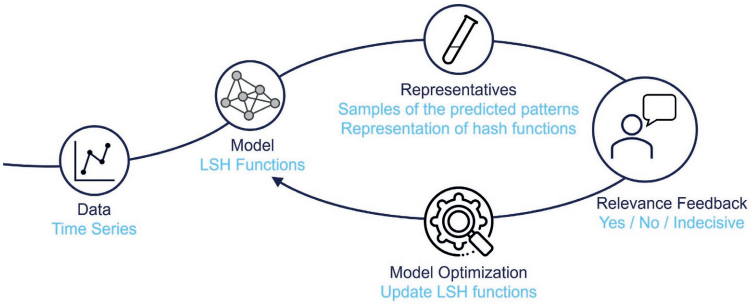


Figure 5.2: **Conceptual Model.** PSEUDo is (1) an instantiation of FDive [20] for MTS retrieval. (2) It models data with LSH, (3) samples the outcomes, (4) invites the user to review the model's similarity understanding, and (5) optimizes the LSH functions based on user feedback.

unfeasible before within our self-imposed performance limits. Moreover, we can see that such a human-in-the-loop active learning approach improves the overall retrieval performance and remains explainable.

For a detailed and precise description, we will use the notations defined in List of Symbols. Namely, we use lowercase letters for scalars and uppercase for matrices. i, j, and r denote time step index, track index, and feedback round index, respectively. We denote vectors as lowercase letters with arrows, like $\vec{\boldsymbol{\theta}}$. Complex vector or matrix operations may require the vectors and matrices denoted as their entries, like $\{\theta_j\}$. we use $\mathbb{C}$ to denote sampled candidates from all found patterns, with $\mathbb{C}_{+/-}$ being positively or negatively labeled found patterns. Next, we will explain how to transform the conceptual model into the concrete PSEUDo pipeline in Figure 5.3.

5.3.1 *Initial Modeling: A High-Speed Foundation*

The initial modeling follows the procedures proposed in the work from [272]. As Figure 5.3 shows, the input data include a query

$$Q = \left[q_{ij}\right]_{\substack{0\le i<\phi \\ 0\le j<d}} = \left[\vec{\boldsymbol{q}}_i \in \mathbb{R}^d\right]_{0\le i<\phi}$$

of length ϕ aligning with Equation 2.2 and a MTS

$$S = \left[s_{ij}\right]_{\substack{0\le i<n \\ 0\le j<d}} = \left[\vec{\boldsymbol{s}}_i \in \mathbb{R}^d\right]_{0\le i<n}$$

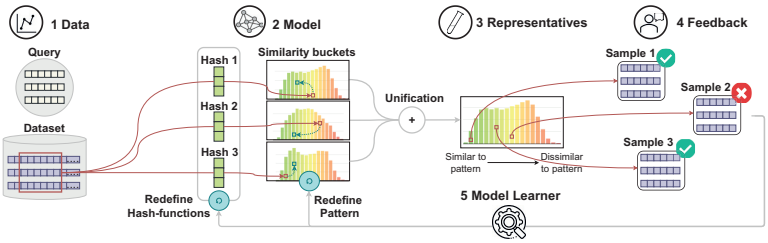


Figure 5.3: **PSEUDo's Data Processing Pipeline.** Following the conceptual model in Figure 5.2. (1) PSEUDo receives a query and preprocessed time series windows as inputs, (2) hash them into hash table buckets representing the distribution of similarity to the query (green depicts similar, red dissimilar), (3) draw representatives from both similar and dissimilar buckets, (4) receive user feedback, and (5) update the hash functions and the query pattern accordingly.

of length n in accordance with Equation 2.1. The output data are the time series windows $\mathbb{F}$ filtered by LSH and by their similarity to Q. Because the method is extensive and established, we describe only the essentials in this section.

As the red box in Figure 5.3 indicates, the method preprocesses S by sequentially scanning (Section 2.6) it with a sliding window of length ϕ (length of Q). Optionally, we can use a range of differently sized sliding windows for patterns with variable duration. Subsequently, the windows and the query are normalized and prepared for hashing.

In the next step, LSH initializes l compound hash functions. Each compound hash function consists of k hash functions $h(\boldsymbol{x})$, $\boldsymbol{x} \in \mathbb{R}^d$. Each hash function h is independently initialized with a parameter vector $\boldsymbol{\theta} = [\theta_j]_{0 \leq j < d}$ containing d elements drawn independently from the standard normal distribution, i.e, $\theta_j \sim \mathcal{N}(0,1)$.

Figure 5.4 illustrates the details during the modeling process. Each h calculates the dot product between its $\boldsymbol{\theta}$ and every time step $\boldsymbol{x}_i$ in a time series window $X = [\boldsymbol{x}_i]_{0 \leq i < \phi}$, thus merging the d tracks in X to a univariate hash code of length ϕ. A projection collision for $\boldsymbol{x}_i$ happens when $|h(\boldsymbol{q}_i) - h(\boldsymbol{x}_i)| \leq \frac{\omega}{2}$ holds with a given error band / hash bucket size ω. Further, a hash collision for X happens when the number of projection collisions between the query Q and X under an h exceeds a threshold ψ. A group hash collision for X happens when X and Q have hash collisions under all k hash functions in a compound hash function. Finally, X is considered a candidate similar to Q when X and Q have

compound hash collision under at least one of the l compound hash functions. How l, k, and other parameters can be optimally configured is referred to [272].

In the next step, the pipeline uses a similarity measure like DTW or ED to calculate the similarity of all candidates to Q. PSEUDo modifies the pattern relevance assessment [272] by applying the similarity measure on the hash code of the candidates rather than on their original MTS form in this stage. This modification has two advantages: 1) the complexity with respect to the number of tracks is reduced from linear to constant; 2) the similarity measure is based on the representation from LSH, which is updated later, reflecting the user's emphasis rather than being purely algorithmic. Although the model building appears complex, it can be computed instantaneously and allows rapid adaption.

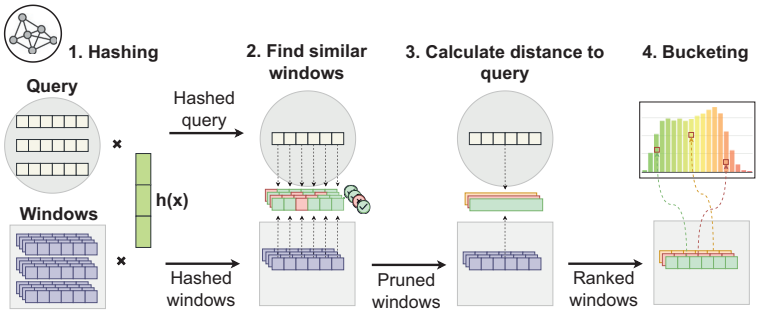


Figure 5.4: **Modeling Process (Detail).** 1) PSEUDo hashes all MTS windows including the query to univariate hash codes, 2) prunes the windows outside the hash bucket containing the query, 3) rank the similarity of the remaining windows to the query based on a similarity measure like DTW or ED.

Retrieval Invariance. Our modeling scheme allows us to account for various types of retrieval invariance (Section 2.7). While the sliding-window-based sequential scanning (Section 2.6) covers horizontal translational invariance (translation along the time axis) (Figure 5.4.1), it fails to capture horizontal scaling (pattern duration scaling) efficiently, a problem also mentioned in Section 2.6. Vertical translational invariance and scaling invariance (bias and amplitude scaling in the y-axis) are handled by normalizing all windows in the first step (also Figure 5.4.1). If necessary, the warping invariance, i.e., time shifts in the pattern, is approached with an elastic measure like DTW.

Parameters. LSH has nearly twenty parameters to set. The user needs to set three of them, and the algorithm derives the rest. We

inherit the setting of the three parameters from the experiments in [272].

- Approximation ratio: 1.3;
- False negative rate: 0.05;
- Hash bucket size: $\omega = 0.75 \times envelop\_radius$;

5.3.2 *Relevance Feedback: Injecting Domain Knowledge*

We achieve the model steering in our conceptual model (Figure 5.2) through the representative selection and relevance feedback steps. They allow users to incorporate their domain expertise and guide how the model should alter its current state. This process has two components: 1) it is inherently a visual-interactive (interface) problem and will therefore be elaborated on in Section 5.4, and 2) we need to decide which *representative* samples to show the user. In PSEUDo, users can give feedback on samples and hash tables.

Relevance Feedback on Found Patterns. We draw samples from the found patterns $\mathbb{F}$ (windows surviving filtering by LSH as explained in Section 5.3.1) and invite the user to label them. The number of candidates can be large, even after filtering with LSH. Users can not process more than possibly a few dozen without being stuck in a tedious labeling process. Following the central idea of (visual) active learning [12, 74, 79], we choose representatives based on the trade-off between exploitation and exploration. On the exploitation side, PSEUDo includes the top five found patterns with the highest similarity scores. However, if we restrict ourselves to top hits, the learning process tends to reinforce the current knowledge while refusing to learn something new. Hence, we also draw five random found patterns that may not have top similarity scores.

Relevance Feedback on Hash Tables. Besides candidate samples, we visualize the hash tables and allow the user to provide feedback on them. The details are kept in Section 5.4.2. The whole LSH model ranks patterns and sorts them into ranked buckets. Therefore, we can represent the individual hash tables as histograms. A histogram visualizes the similarity distribution perceived by a hash table. We interpret the commonly occurring histogram patterns in the following manner: a positively skewed shape  points to a hash function, which observes many windows similar to the query, while a negatively skewed shape 

means the opposite. Unitary  or bi-modal distributions  refer to undecided or decisive hash functions.

5.3.3 *Updating Model: Encoding Knowledge into Parameters*

Conventional LSH does not contain trainable parameters and cannot be optimized directly by relevance feedback. However, the randomly initialized parameters in the hash functions do not have to persist. As described in Section 5.3.1, a hash function h performs dot product $\boldsymbol{\theta} \cdot \boldsymbol{x}_i$ between its hashing vector $\boldsymbol{\theta}$ and the i-th time step $\boldsymbol{x}_i$ in a window $X = \left[\boldsymbol{x}_i \in \mathbb{R}^d\right]_{0 \le i < \phi}$. This operation can be interpreted as a weighted merge of d tracks in X with the track weights given by $\boldsymbol{\theta}$.

To capture relevance feedback, we propose to modify $\boldsymbol{\theta}$ with a learned weighting vector $\boldsymbol{w}$ and use the updated vector $\boldsymbol{\theta'} = \boldsymbol{w} \odot \boldsymbol{\theta}$ in place of $\boldsymbol{\theta}$, where $\odot$ denotes the Hadamard product. Given two matrices $A = \left[a_{ij}\right]$ and $B = \left[b_{ij}\right]$ of the same dimensions, the Hadamand product outputs a matrix $\left[a_{ij} \cdot b_{ij}\right]$ of the same shape.

To avoid vanishing or exploding parameters, we want to retain the expectation of the magnitude of $\boldsymbol{\theta}$, namely $\mathrm{E}\left(\|\boldsymbol{\theta'}\|\right) = \mathrm{E}\left(\|\boldsymbol{\theta}\|\right)$. It is achieved by normalizing the magnitude of $\boldsymbol{w}$ to $\sqrt{d}$, where d is the dimension (number of tracks) of the time series first appeared in Section 2.1.1. We normalize $\boldsymbol{w}$ instead of $\boldsymbol{\theta'}$ directly because there are multiple $\boldsymbol{\theta}'$ but only one $\boldsymbol{w}$.

Recall that $\theta_j \sim \mathcal{N}(0, 1)$, where θ_j is the j-th element of $\boldsymbol{\theta}$ corresponding to the j-th track (Section 5.3.1). The squared magnitude of $\boldsymbol{\theta}$ is

$$\begin{aligned} \mathrm{E}\left(\|\boldsymbol{\theta}\|^2\right) &= \mathrm{E}\left(\sum_{j=1}^{d} \theta_j^2\right) = \sum_{j=1}^{d} \mathrm{E}\left(\theta_j^2\right) = \sum_{j=1}^{d} \mathrm{E}\left(\left(\theta_j - 0\right)^2\right) \\ &= \sum_{j=1}^{d} \mathrm{E}\left(\left(\theta_j - \mathrm{E}\left(\theta_j\right)\right)^2\right) = \sum_{j=1}^{d} \mathrm{Var}\left(\theta_j\right) = \sum_{j=1}^{d} 1 = d \end{aligned} \tag{5.1}$$

Likewise, since $\theta'_j = w_j \cdot \theta_j \sim \mathcal{N}\left(0, w_j^2\right)$, it follows

$$\mathrm{E}\left(\|\boldsymbol{\theta'}\|^2\right) = \ldots = \sum_{j=1}^{d} \mathrm{Var}\left(\theta'_j\right) = \sum_{j=1}^{d} w_j^2 = \mathrm{E}\left(\|\boldsymbol{w}\|^2\right)$$

Aiming at $\mathrm{E}\left(\|\boldsymbol{\theta'}\|^2\right) = \mathrm{E}\left(\|\boldsymbol{\theta}\|^2\right)$, we set

$$\mathrm{E}\left(\|\boldsymbol{w}\|^2\right) = \mathrm{E}\left(\|\boldsymbol{\theta'}\|^2\right) = \mathrm{E}\left(\|\boldsymbol{\theta}\|^2\right) = d$$

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

PSEUDo allows two types of relevance feedback, namely, feedback on the found pattern samples and the hash tables. Accordingly, we maintain two weight vectors $\boldsymbol{w}_s$ and $\boldsymbol{w}_h$ for samples and hash tables, respectively.

Sample Relevance Adaption. The feedback on the positively labeled samples, $\mathbb{C}_+ = \{C_+\}$, is transformed to track importance $\boldsymbol{w}_s$. We calculate a distance function f_{dis} based on DTW or ED between the tracks of each positively labeled sample $C_+ = \left[\boldsymbol{c}_j\right]_{0\leq j<d}$ and these of the query $Q = \left[\boldsymbol{q}_j\right]_{0\leq j<d}$. We define $\boldsymbol{a} = \left[\sum_{C_+\in\mathbb{C}_+} f_{dis}\left(\boldsymbol{q}_j, \boldsymbol{c}_j\right)\right]_{0\leq j<d}$ as the aggregate distances between the corresponding tracks in C_+ and Q. Next, we normalize the entries in $\boldsymbol{a}$ to between $[0,1]$ with $\boldsymbol{a}^* = \left[a_j^* = a_j / \sum_{j=1}^{d} a_j\right]_{0\leq j<d}$. Then we convert distances to their negatively correlated weight vector yielding $\boldsymbol{w}_s^* = \left[1 - a_j^*\right]_{0\leq j<d}$, which is subsequently normalized to $\boldsymbol{w}_s = \boldsymbol{w}_s^* \frac{\sqrt{d}}{\|\boldsymbol{w}_s^*\|}$.

Hash Table Relevance Adaption. The feedback on the hash tables can be implemented likewise. Let $\Theta_+ = \{\boldsymbol{\theta}_+\}$ be the parameter vectors of all hash functions labeled as positive by the user. Then, we can define $\boldsymbol{w}_h^* = \left[\sum_{\boldsymbol{\theta}_+\in\Theta_+} \theta_{j+}^2\right]_{0\leq j<d}$. After normalizing the vector magnitude to $\sqrt{d}$, we get $\boldsymbol{w}_h = \boldsymbol{w}_h^* \frac{\sqrt{d}}{\|\boldsymbol{w}_h^*\|}$

Finally, we merge $\boldsymbol{w}_{s,b}$, $\boldsymbol{w}_{h,b}$ in b-th feedback round and $\boldsymbol{w}_{b-1}$ into $\boldsymbol{w}_b$ through a linear combination with a learning rate ρ, which can be gradually modified to enforce exploration stability [20].

$$\boldsymbol{w}_b^* = (1-\rho)\boldsymbol{w}_{b-1} + \frac{\rho}{2}(\boldsymbol{w}_{s,b} + \boldsymbol{w}_{h,b})$$

$$\boldsymbol{w}_b = \boldsymbol{w}_b^* \frac{\sqrt{d}}{\|\boldsymbol{w}_i^*\|}$$

Query Adaption. As shown by the backward arrow in Figure 5.3, we update not only the LSH hash functions but also the query because the initial query does not necessarily represent the generally desired shape of the pattern. In each training iteration, we integrate the query Q and the set of positively labeled samples $\mathbb{C}_+$. This operation is known to be nontrivial [204] and impacts future exploration direction. After the trial with the naïve element-wise average yields unsatisfactory results, because the average often resembles none of the original windows, we decided for Dynamic Time Warping Barycenter Averaging (DBA)

[205], as it takes distortion and time shifts during averaging into consideration [93, 204, 205].

5.4 USER INTERFACE

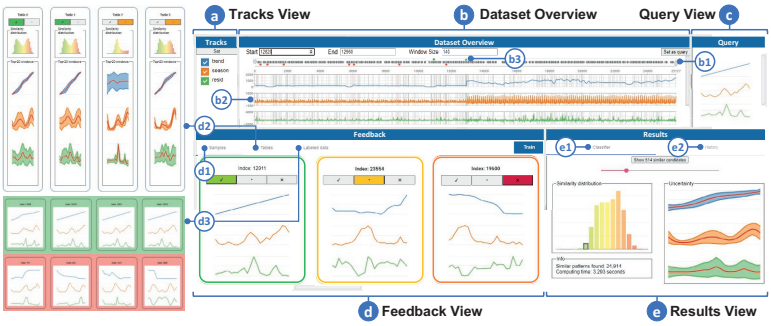


Figure 5.5: **Interface Overview.** We plot the selected tracks from the *Tracks View* (a) in the *Dataset Overview* (b) along with the user-defined query in (c). The *Feedback View* (d) depicts classification result samples and information about the hash tables and is used to receive relevant feedback. The *Results View* (e) displays the result distribution and provides search history management.

We designed PSEUDo's visual interface depicted in Figure 5.5. It comprises five interlinked views facilitating data exploration, query definition, process monitoring, result inspection, relevance feedback, and state management. A REST API connects the web-based user interface to PSEUDo's backend algorithm. The backend (in Python and C++) and frontend (with Angular) code are available under `https://git.science.uu.nl/vig/sublinear-algorithms-for-va/pseudo`. The video `https://www.youtube.com/watch?v=oJfXoDyZRPY` demonstrates the user interface as well as a concise workflow.

5.4.1 *Dataset Overview, Track View, and Query View.*

The *Dataset Overview*, *Tracks View*, and *Query View* collaborate closely. The *Dataset Overview* (Figure 5.5.b) plots tracks selected in the *Tracks View* (Figure 5.5.a) together with the window labels (Figure 5.5.b3 dots in the range slider Figure 5.5.b1 and marked intervals in the line chart Figure 5.5.b2). Besides direct panning and zooming, a range slider (Figure 5.5.b1) above the

track curves (Figure 5.5.b2) serves as a mini-map for navigation and an overview of predictions and labels.

The *Query View* (Figure 5.5.c) shows the user-defined multivariate query. The user defines the query in a query-by-example (Section 2.9) manner by selecting a region in the *Dataset Overview*. The user can change the query on the fly, but PSEUDo must repeat the hashing process whenever the query length varies. Typically, users have to wait several seconds for LSH parameter estimation. While not focused on in this project, PSEUDo's query definition interface currently lacks the option to define patterns shifted across tracks (inter-track time shifts explained in Section 2.7). We will investigate better query interfaces for MTS data following recent examples, such as [40], in Section 6.4.2.

5.4.2 *Feedback View*

The *Feedback View* (Figure 5.5.c) shows representatives of all found patterns $\mathbb{F}$, visualizes hash tables, and keeps track of labeled data. We differentiate three respective tabs for different purposes.

The *Samples Tab* (Figure 5.5.d1) lists samples of the classified windows. They are surrounded by frames color-encoded from green over yellow to red, indicating decreasing relevance or similarity to the query. Right above the line charts of the samples, PSEUDo invites the user to label the windows by clicking ✔ for acceptance, • for abstain, and ✖ for rejection, as described in Section 5.3.3 (Sample Relevance Adaption).

PSEUDo visualizes the hash tables in the *Tables Tab* (Figure 5.5.d2). Each hash table is visualized as a histogram, showing the relevance distribution perceived by the hash function. Like the frames surrounding the samples in the Samples Tab, the bars in the histograms here are also color-encoded from green to red, indicating decreasing pattern relevance or similarity to the current query. To help understand how well the hash functions work, we plot each time step's mean, minimum, and maximum values among the top-20 similar windows for each hash function. The mean value curves portray the pattern shape perceived as similar by the hash function, while the minimum and maximum values form the lower and upper bound of the pattern. The band's tightness implies the certainty or importance of the track during classification. Based on this visual encoding, the user can modify the hash tables' importance by clicking ✔ for acceptance and • for rejection, as described in Section 5.3.3 (Hash Table Rele-

vance Adaption). In the future, we plan to implement a ranking or sampling mechanism to avoid showing all hash tables.

The accepted samples $\mathbb{C}_+$ and the rejected samples $\mathbb{C}_-$ labeled in the current feedback round are kept in the *Labeled Data Tab* (Figure 5.5.d3). Users can revise decisions before clicking the Train button. PSEUDo will consider the labeled sample windows and labeled hash tables in the next training round.

5.4.3 *Results View*

The *Results View* (Figure 5.5.e) shows the outcome statistics and provides git-like version management.

In the *Classifier Tab* (Figure 5.5.e1), we use a result histogram to visualize the distribution of similarity between the query and all windows that survive LSH's pruning. Clicking a bin in the histogram shows the reconstructed visual pattern analogous to the ones in the *Feedback View*. Rather than using the top-20, this pattern result view summarizes all windows in the chosen bin. The mean curves also show the average form within this bin, bounded by each time step's minimum and maximum values to illustrate the variance within tracks. The histogram and the reconstructed shape help the user better understand the classification result and provide guidance on the strictness during the labeling process. Besides, users can set the number of top candidates and display them in the dataset overview.

5.5 EVALUATION

We evaluated PSEUDo through three distinct evaluation threads. First, we benchmarked accuracy and speed/scalability with representative techniques for time series pattern search. Second, we verified the steerability of the relevance feedback mechanism. Third, we validated the usability, including understandability, through an interview with an expert from the energy consumption domain. Please note that accuracy and speed evaluation for LSH has already been conducted in [272]. We extend the accuracy evaluation with labeled datasets and the speed evaluation with a scalability test with respect to data dimensions.

5.5.1 *Experiment Setup*

We conducted all experiments on a standard laptop running 64-bit Windows 10 Enterprise with an Intel i7-8650U CPU, 16 GB RAM, and 1 TB HDD.

We used six datasets with different characteristics, namely

- `Deep Valve`[2]: electrical current through a solenoid valve,
- `EEG Eye State` [221]: electroencephalogram with labeled eye state (close/open) of a subject,
- `Filling Prediction`: relative air filling in a petrol engine,
- `Variable Displacement`: rotational speed and acceleration of a petrol engine,
- `EEG Schizophrenia` [218]: electroencephalogram of a subject taking a button pressing test for diagnosing Schizophrenia, and
- `Intelliekon` [233]: household energy consumption.

The first four datasets have ground truths. Therefore, we could use them in the accuracy benchmark. While the speed benchmark also uses these four, we would like to evaluate `PSEUDo`'s scalability with respect to even higher data dimensions. Hence, we included the unlabeled `EEG Schizophrenia` dataset with 70 tracks because the speed/scalability test does not require labels. The last dataset is from our interviewed expert and used in the expert study.

We used the following four representative and state-of-the-art methods for benchmarking.

Correlation. It is one of the standard similarity measures. We used our own python implementation `float(numpy.dot(*normalize(np.vstack((seq_1, seq_2)) + bias)))` where `seq_1` and `seq_2` are two time series segments and `bias` is a small term to tackle constant segments and avoid numerical problems.

DTW [23, 237]. It is the most popular elastic distance measure for time series with "hard-to-beat" accuracy for time series classification [15] mentioned in Section 2.5. We used `dtw.dtw(seq_1, seq_2, dist_method="sqeuclidean", step_pattern="symmetric1", distance_only=True).distance` with the library `dtw-python` from `https://dynamictimewarping.github.io/python/`.

2 Datasets without a reference are proprietary house-internal datasets that cannot be published.

MASS [187, 188]. The so far fastest similarity search algorithm mentioned in Section 2.6. Ultimately, it uses ED. We used the version from `https://github.com/matrix-profile-foundation/mass-ts`, specifically `mass\_ts.mass2(ts, query)` to calculate the distance profile. GPU support is deactivated. We would like to clarify that though already fast, the Python version of MASS is still much slower than its MATLAB version. The speed difference of the two versions could result from the implementation or the efficiency difference between Python and MATLAB. MASS in different programming languages can be found in `https://www.cs.unm.edu/~mueen/FastestSimilaritySearch.html`.

SAX [164, 165]. A popular time series representation mentioned in Section 2.4. Together with its dedicated similarity measure, SAX can be used for similarity search. We used our own implementation. The authentic SAX assumes that the values in the time series are normally distributed. Based on this assumption, it calculates quantiles to discretize the values. Our data are not normally distributed. Therefore, we calculate the genuine value distribution then the quantiles accordingly. Furthermore, we fit the SAX (estimate distribution and calculate quantiles) with the query instead of the time series in the database, which significantly improves the accuracy, because sometimes, the query has a much smaller value range as the time series in the database. On a side note, SAX uses PAA as a necessary preprocessing step, which reduces data volume. To ensure a fair comparison, we apply PAA with the same resolution for other methods.

Table 9 lists the complexities of the similarity measures.

Similarity Measure	Complexity
Correlation	$O(dn\phi)$
ED (MASS))	$O(dnlog(n))$
DTW	$O(dn\phi^2)$
SAX	$O(dn\phi)$
LSH (PSEUDo)	$O(n\phi)$

Table 9: **Complexity of Common Similarity Measures.** d denotes the dimension or number of tracks, n the time series length or number of windows after preprocessing with sliding windows, and ϕ the query length. Note that the query length ϕ is irrelevant for MASS's complexity, and d does not influence LSH's complexity.

5.5.2 *Accuracy and Speed*

We benchmarked PSEUDo's accuracy with deactivated relevance feedback against the benchmark methods. During accuracy benchmark, we used not only the time-step-based metrics including accuracy, balanced accuracy, precision, recall, and F1 score, but also the pattern-interval-based metrics including AP30 and AP50, as explained in Chapter 4.

Figure 5.6 shows the result of the accuracy benchmark. It implies that each method performs well on some datasets while poorly on the other. This result implies that the notion of relevance/similarity may vary in different use cases, which leads to the need for adaptive similarity measures.

Next, we measured the average elapsed time over five repetitions for all methods on the same four labeled datasets and report the results in Figure 5.7. PSEUDo, in our *slow* DTW configuration, is comparable with MASS, the fastest similarity search algorithm so far, and even surpasses the latter in high-dimensional cases like the EEG eye state dataset. In low-dimensional datasets, the pruning effect of LSH plays a major role by reducing the number of candidates. Whereas in high-dimensional cases, LSH's weighted track merging provides sublinear scalability.

To further verify PSEUDo's scalability, we measured the elapsed time with an increasing number of tracks in the high-dimensional EEG Schizophrenia dataset. The result in Figure 5.8 confirms PSEUDo's good scalability for high-dimensional data. While other methods scale linearly with the data dimensions, PSEUDo scales sublinearly

5.5.3 *Steerability*

While Section 5.5.2 measures the "open-loop" performance without relevance feedback, this section verifies the effectiveness and mechanics of relevance feedback through accuracy and track importance evolution.

In general, it is challenging to evaluate active learning systems objectively. Inspired by [235], we evaluated PSEUDo's steerability with three agents simulating user behavior. The first agent simulated normal user feedback and labeled the samples according to the ground truths. If a sample had at least 50% IoU with a ground truth, the sample was accepted, otherwise rejected. The second agent accepted all samples, and the third rejected all. They formed the control groups. We used the EEG Eye State dataset to demonstrate PSEUDo's steerability because this dataset is high-

Dataset	Method	Accuracy	Balanced accuracy	Precision	Recall	F1 score	AP30	AP50
Deep valve	Correlation	0.90	0.77	0.80	0.57	0.66	0.78	0.22
- pattern contains pulses	DTW	0.92	0.81	0.92	0.63	0.75	0.59	0.26
- large horizontal scaling	ED (MASS)	0.18	0.50	0.18	1.00	0.31	0.06	0.01
	LSH (PSEUDo)	0.92	0.87	0.78	0.79	0.78	**0.90**	**0.74**
	SAX	0.96	0.94	0.89	0.91	**0.90**	0.78	0.36
EEG Eye State	Correlation	0.64	0.66	0.57	0.80	0.67	0.29	0.10
- pattern very fuzzy	**DTW**	0.78	0.79	0.72	0.84	**0.77**	**0.61**	**0.50**
- extreme horizonal scaling	ED (MASS)	0.72	0.72	0.66	0.78	0.71	0.29	0.16
	LSH (PSEUDo)	0.70	0.70	0.66	0.67	0.67	0.25	0.17
	SAX	0.45	0.50	0.45	1.00	0.62	0.10	0.09
Filling prediction	Correlation	0.97	0.79	0.95	0.59	0.73	0.79	0.79
- pattern mainly stationary	DTW	0.94	0.90	0.54	0.85	0.66	0.77	0.77
- pattern variation	ED (MASS)	0.97	0.80	0.90	0.60	0.72	0.78	0.78
concentrate on both ends	LSH (PSEUDo)	0.95	0.67	0.90	0.35	0.51	0.55	0.55
	SAX	0.98	0.87	0.93	0.74	**0.82**	**0.94**	**0.94**
Variable Displacement	**Correlation**	1.00 ± 0.00	0.96 ± 0.02	0.98 ± 0.01	0.91 ± 0.03	**0.95 ± 0.01**	**0.83 ± 0.34**	**0.83 ± 0.34**
- Vibrating pattern	DTW	1.00 ± 0.00	0.92 ± 0.07	0.88 ± 0.10	0.84 ± 0.15	0.85 ± 0.10	0.74 ± 0.31	0.67 ± 0.34
	ED (MASS)	0.94 ± 0.04	0.72 ± 0.10	0.14 ± 0.09	0.49 ± 0.23	0.19 ± 0.09	0.07 ± 0.07	0.02 ± 0.02
	LSH (PSEUDo)	0.99 ± 0.01	0.74 ± 0.12	0.69 ± 0.34	0.48 ± 0.24	0.54 ± 0.24	0.31 ± 0.21	0.30 ± 0.21
	SAX	1.00 ± 0.00	0.90 ± 0.07	0.95 ± 0.04	0.80 ± 0.14	0.86 ± 0.10	0.66 ± 0.33	0.66 ± 0.33

Figure 5.6: **Accuracy Benchmark.** The best F1-score, AP30, AP50, and method for each dataset are highlighted bold and red. Different methods work well on different datasets, indicating varying notions of relevance/similarity in different use cases. PSEUDo's relevance feedback mechanism is deactivated in this experiment.

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

Figure 5.7: **Speed Benchmark.** Search time is in seconds and averaged over five repeats (± standard deviations). The shortest execution time for each dataset are highlighted bold and red. PSEUDo is comparable in speed with the so far fastest similarity search algorithm MASS and surpasses the latter in the high-dimensional case (the EEG Eye State dataset).

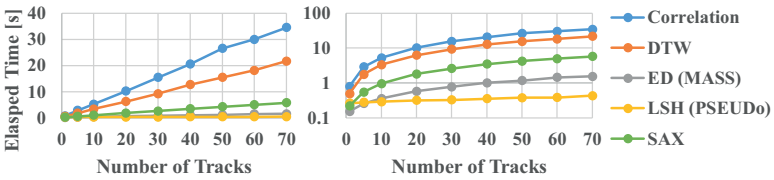


Figure 5.8: **Scalability Benchmark.** PSEUDo's speed stands out as the data dimensions increase. Left: linear scale; right: logarithmic scale.

dimensional and has ground truths. The target patterns correspond to the time periods when the subject's eyes are closed. This dataset contains 14 tracks. They are not equally informative for the pattern search. Figure 5.10 plots the 14 tracks with target patterns / ground truths in gray. We can notice that the patterns in some tracks, e.g., "F8" and "AF4", are much more prominent, while some, e.g., "T7" and "P7", are not as helpful.

We ran PSEUDo on the dataset with five feedback rounds and recorded the evolution of the accuracy as well as the track weights. Figure 5.9 shows the accuracy metrics AP30 and AP50 in five feedback rounds with the three agents. We witnessed an accuracy increase with the normal agent and no improvement in the control groups. It confirmed that sensible feedback helped improve PSEUDo's accuracy.

To further verify PSEUDo's feature-selection-based relevance feedback mechanism, we plotted the evolution of the track weights ($\boldsymbol{w}$ in Section 5.3.3) in Figure 5.10. As expected, the more instructive tracks were attached more weights, and the less informative tracks got down-weighted. In contrast, the track weights from the "all accepted" agent group evolved randomly because the feedback contained no useful information. The track weights from the "all rejected" agent group stayed the same because

PSEUDo currently cannot exploit the rejected samples, which is one of its limitations.

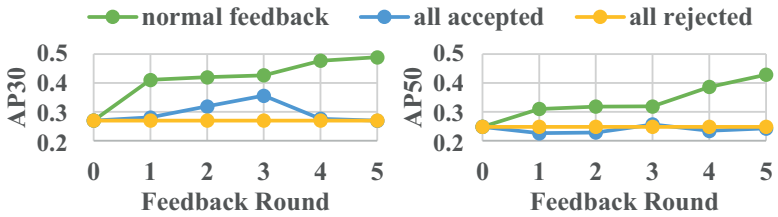


Figure 5.9: **Accuracy Evolution.** During relevance feedback, accepting or rejecting all samples yields no accuracy improvement, while faithful user feedback contributes to the accuracy. Left: AP30; right: AP50.

5.5.4 *Expert Study*

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

For our data expert's use case (depicted in Figure 5.5), we conducted experiments on a smart meter dataset collected from a field study between 2009 and 2010 within the German research project Intelliekon [233]. For two of 1720 randomly picked households, the expert decomposed the one-year hourly energy consumption into the trend, seasonal, and residual components

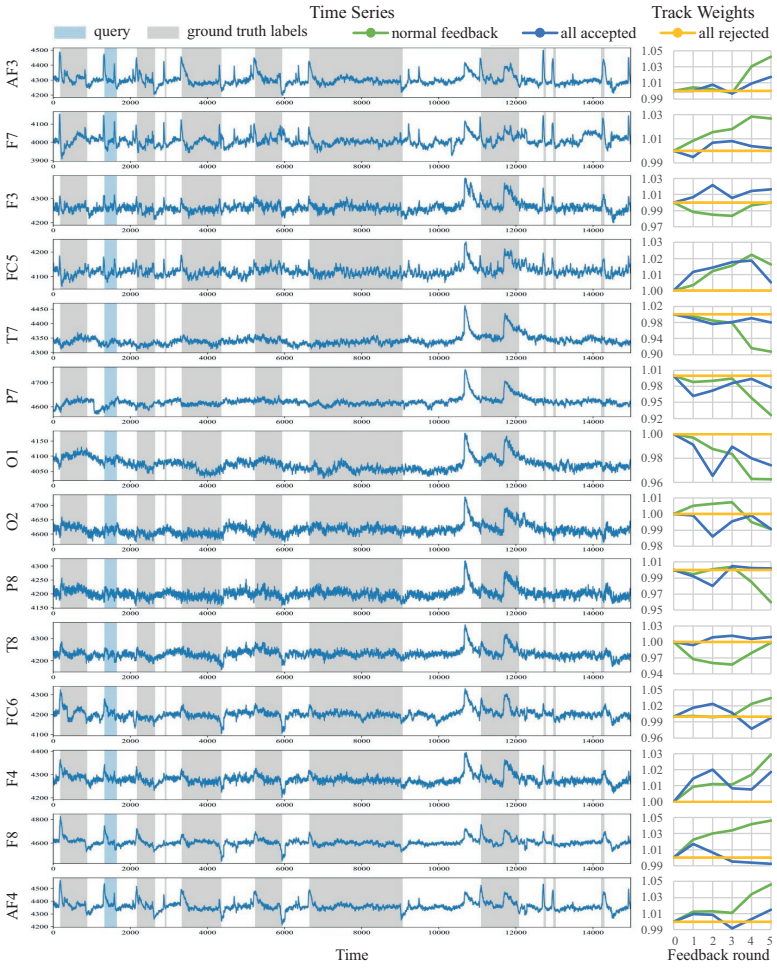


Figure 5.10: **Track Weight Evolution.** The target patterns are not equally prominent in all tracks. For instance, the patterns in the tracks "F8" and "AF4" are visually more pronounced, while less informative i "T7" and "P7". In the steerability experiment, PSEUDo attached higher weights to the former, namely, the more helpful and relevant features/tracks.

based on a Bayesian structural time series model. These three tracks are loaded as our MTS input (Figure 5.5 (a)). During the visual exploration with PSEUDo, the shifts in user behaviors can be identified by searching for characteristic patterns in various temporal scales (monthly, weekly, and daily).

During the study, the expert operated PSEUDo. It helped the expert identify shifts in user behavior and inspect the behavior differences between user groups. Overall, the expert spoke highly of PSEUDo. The expert highlighted the interaction during querying and exploration: *"It is very useful to have a view of similarities among searched results to help pick up the thresholds of similarity."*; i.e., Figure 5.5(e). In the above use case, the expert did not address any significant usability issues. Only when we imagined the broader use cases with multiple households, the expert suggested that it would be more convenient to support visualizing multiple files as track groups. From his perspective, a nice add-on for PSEUDo would be a textual explanation of the hashing algorithm and its visualizations because it can help him avoid explaining the results to customers without a deep technical background.

5.6 DISCUSSION

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

On the conceptual side, we found that a thorough task taxonomy for MTS analysis is missing. We can map PSEUDo's high-level tasks into Brehmer and Munzner's typology [33], e.g., our tool implements `browse`, `explore`, `locate`, and `lookup` tasks. However, we did not focus on MTS tasks like finding patterns with significant cross-track time shifts or tasks that assess the invariance properties of specific patterns as these tasks are conceptually on a different level of abstraction. We will deal with this problem in the subsequent chapters.

We invested much effort in the scalable and fast search backend but did not focus much on the visual interface. Our rather simplistic use of standard visualizations and the implemented query-by-example system, though work flawlessly and effortlessly, demonstrate this aspect distinctively. We admit that the visualization does not scale to very high-dimensional data. We are considering reactive switching between line charts, horizon diagrams [116], and color-encoded pixels [105]. We are also considering distorting the time series in the plot, e.g., using different heights for tracks with varied importance and distorting the time to compress unimportant time regions. In the future, we plan to extend our Visual Analytics contributions in two directions: First, we will tackle query-definition challenges, like *How can users specify a) multi-track queries or b) queries with a temporal relationship between them?*, with new query definition panels and plan to apply interactive augmentation, like Shadow Draw [236], to help with this process.

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

Another interesting challenge for Visual Analytics is tracking *biases* and *convergence* in exploration processes. Currently, we include negative and indecisive labels to promote target class separation, which inevitably adds to a potential confirmation bias in every iteration. We could, however, also regard every new positive label, which is distinct from the current set of positive labeled items, as a novel exploration thread or fork. This enables *quality metrics to quantify task change(s)*, i.e., a strong difference between positive labels could signal a transition from exploitation to exploration.

Finally, we will address the evaluation of more PSEUDo's components in the future. Specifically, we plan to run a study on the proposed relevance feedback on the hash tables (Section 5.3.2) focusing on proving the usability and effectiveness of this feedback mechanism. Second, we calculate the similarity based on the hash codes rather than the original time series windows as described in Section 5.3.1. Although it shows no problem in the accuracy evaluation, it is sensible to measure the potential accuracy sacrifice brought by this modification.

5.7 CONCLUSION

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

6

DESCRIBING TIME SERIES RETRIEVAL INVARIANCES IN SYMBOLIC SPACE

Figure 6.1: **SAXRegEx's Data Processing Pipeline.** We propose SAXRegEx for pattern search in MTS in the presence of various distortions. It undergoes five steps. 1) Stringify the time series to allow text retrieval methods; 2) zipping tracks to enable multivariate retrieval; 3) extract regex from the stringified query; 4) expand the query to tackle distortions; 5) search with regex engine.

In this chapter, we present SAXRegEx. It formulates the query as a regular expression and searches for the regex in the symbolically represented time series. Like PSEUDo, SAXRegEx also prioritizes speed over accuracy. The highlight of SAXRegEx is query expansion that enables describing multiple pattern distortions. Among them, SAXRegEx is particularly good at capturing patterns that scale heterogeneously among time axis, or exhibit time shifts between tracks. [1]

6.1 INTRODUCTION

In information retrieval, the relevant pieces seldom fit the query perfectly. Specifically in time series pattern search, the target patterns in the time series database are seldom identical to the query / the example of a target pattern. Instead, the patterns in time series are plagued with various distortions. Figure 6.2 illustrates some distortions, such as duration scaling (down-sampled or up-sampled), warped (heterogeneous duration scaling), and inter-track time shifts, also introduced in Section 2.7.

Unfortunately, all the distortions in Figure 6.2 exist in our data. Our target signals constantly vary in duration and are often dis-

1 This chapter is based on our publications PB4 (awarded Best Paper in EuroVA 2022) and PB5 with textual modifications for a coherent information flow.

tributed in several tracks with significant time shifts between tracks. The distortions call for corresponding retrieval invariance of the search algorithm. In accordance, we formulate the research question for this chapter: **How to retrieve patterns with various distortions in multivariate time series?**

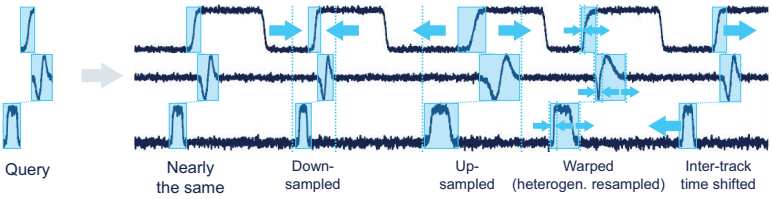


Figure 6.2: **Distorted Patterns.** The query on the left is searched for in the MTS on the right. Target patterns in the time series exhibit distortions in various forms.

On one hand, this problem is interesting and important, because distortions are a major hindrance to an accurate retrieval. Conversely, distortions are a burden for computation, especially when distortions need to be described in a brute-force way. On the other hand, this problem is challenging, because some distortions are tricky to deal with, particularly in multivariate cases. Numerous time series indexing techniques [47, 90, 109] can already find UTS patterns with minor distortions efficiently. Univariate retrieval with certain distortion types, especially warping and sometimes horizontal scaling along the time axis, is also addressed [103, 137, 284]. Horizontal length scaling and warping within a track can be regarded as heterogeneous horizontal resampling. Recently, multivariate cases have received increasing attention [153, 237], because an event can reveal itself in multiple signals. For UTS pattern search, current methods do not sufficiently address the duration scaling problem. Heterogeneous horizontal scaling is hard to capture, due to the de facto preprocessing step with a sliding window, usually assuming a fixed pattern length silently, as mentioned in Section 2.6. Multivariate distortion-invariant time series retrieval remains largely untouched, in particular, for inter-track time shifts. It is challenging above all due to the complexity when considering temporal dynamics and attribute interrelation simultaneously.

In this chapter, we present `SAXRegEx`, a method for pattern search in MTS in the presence of various distortions described in Section 2.7, such as duration variation, warping, and time shifts between signals. It is based on symbolic representation to enable text retrieval methods and regular expressions as our search en-

gine. Thereupon, we design query expansion to deal with the mentioned distortions.

SAXRegEx shows comparable accuracy, sometimes minor accuracy loss, to state-of-the-art techniques on datasets without these distortions and better performance in multivariate datasets with mentioned distortions. On the other hand, even for data without such distortions, SAXRegEx outperforms the benchmarking methods in terms of speed[1]. We designed a user interface with an emphasis on multivariate query definition with inter-track time shifts and algorithm parameter setting. A case study demonstrated its usability. While we designed the method primarily for automotive data, it is well transferable to other domains.

6.2 RELATED WORK

The method proposed in this chapter is based on the time series representation SAX and attacks the problem of pattern distortions in MTS. Please refer to Section 2.4 and Section 2.7 for more information on time series representations and pattern distortions, respectively. In this section, we will explain SAX and query expansion, the methods on which we base our proposal.

Symbolic Aggregate approXimation (SAX). Introduced in [164] and extensively explained in [165], SAX converts a UTS into a string to reveal certain patterns in the data and enables text processing methods. SAX goes through three major steps. Firstly, it applies PAA to merge temporally consecutive time steps piecewise into one by averaging them. The non-overlapping piecewise mean of PAA reduces data volume and smooths the curves. Next, it partitions the value range with quantiles as breakpoints to discretize the values. It uses quantiles so that each value range partition has roughly the same number of pieces. Finally, it assigns each value range partition a symbol, typically starting with "a", followed by "b", and onwards. In addition, SAX proposes a distance measure between symbols. It creates a lookup table, which, given two symbols, outputs their distance. Our method runs the search directly in the symbolic space with regex matching instead of a direct distance calculation, yielding a speed boost. We exploit SAX's less utilized "numerosity reduction" property (combining repetitive symbols) through regex quantifiers to add elasticity (re-

1 The MATLAB version of MASS is significantly faster than SAXRegEx. We were unaware of the issue in the previous work [274], which used a slower Python version in the experiments.

garding horizontal scaling along the time axis) to the query to deal with distortions.

Query Expansion. One highlight of our work is query expansion to address various distortions exhibited in time series patterns. Originating from information retrieval, it refers to techniques that reformulate the query given by the user to retrieve more precise or broader information [253], especially in the context of text retrieval. Many techniques strive to achieve this goal, by following linguistic rules, mining corpus-specific patterns, utilizing search logs, referring to other sources from the web, or relevance feedback [14, 41]. Notwithstanding some terms like "query by content" (Section 2.3) and "relevance feedback" (Section 2.8), time series pattern search differs from classic information retrieval in that we cannot parse time series patterns semantically, especially when approaching the problem in a domain-agnostic way. Nonetheless, our query expansion technique in some sense coerces a time series (the equivalent of the "corpus" in text retrieval) into repetitive patterns with SAX and formulate the query accordingly. It may fall into the category of corpus-based approaches, which typically involve statistical techniques to analyze correlated terms in the corpus in case of text retrieval [14, 41].

6.3 METHOD

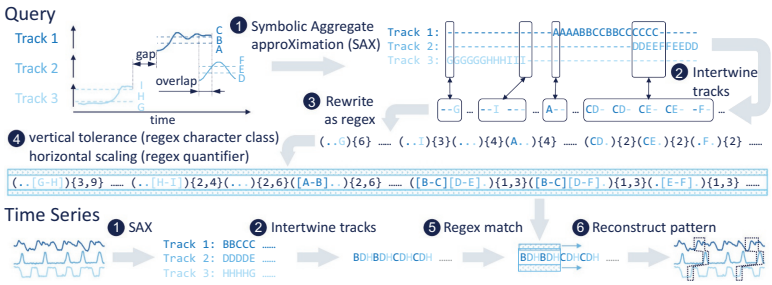


Figure 6.3: **SAXRegEx's Data Processing Pipeline.** It undergoes six stages. 1) convert with SAX each track with a different alphabet to enable text retrieval methods; 2) intertwine tracks in the query and in the time series to process multiple tracks with a single regex, while leaving wildcards "." for "gaps" to capture inter-track time shifts; 3) extract regex from the query string by combining consecutive repetitive symbols; 4) augment the regex with character classes and quantifiers to deal with distortions; 5) search for the query regex in the time series string; and 6) reconstruct the numerical data in each track.

SAXRegEx consists of six stages illustrated in Figure 6.3. First, it encodes with SAX each track in the query and in the time series in Stage ❶. It reduces data volume, smooths curves, and enables methods for text retrieval, as mentioned in Section 6.2. Next, SAXRegEx intertwines tracks in the query and in the time series in Stage ❷. This stage enables simultaneous processing of multiple tracks through a single regex. The word "intertwine" refers to the time-step-wise "zipping" of sequences. Namely, a multivariate symbolically represented time series $\underset{\sim}{S} = \left[\underset{\sim}{s}_{ij} \in \mathbb{A}\right]_{\substack{0 \leq \underset{\sim}{i} < n \\ 0 \leq j < d}}$ is transformed to

$$\underset{\sim}{\boldsymbol{s}}' = \left[\underset{\sim}{s}'_{\underset{\sim}{i}'} = \underset{\sim}{s}_{\underset{\sim}{i}\ \mathrm{div}\ d, \underset{\sim}{i}\ \mathrm{mod}\ d}\right]_{0 \leq \underset{\sim}{i}' < d \cdot i + j}$$

where $\underset{\sim}{i}$ denotes the index of the symbol step in the symbolic represented time series $\underset{\sim}{S}$, $\underset{\sim}{i}'$ the index of the symbol step in the symbolic represented time series after intertwining the tracks $\underset{\sim}{\boldsymbol{s}}'$, and $\mathbb{A}$ the alphabet of SAX. In Stage ❸, repetitive symbol groups are merged through regex quantifiers, which exploits SAX's "numerosity reduction" property [165]. Subsequently, Stage ❹ conducts query expansion to cope with distortions. As shown in Figure 6.3, it adds a tolerance band to the query by allowing character classes rather than individual characters; thereupon, it makes the query elastic along the time axis by substituting the fixed quantifier with ranges. The penultimate Stage ❺ searches for the query regex in the time series string. Finally, Stage ❻ reconstructs the pattern in each track by a fine regex matching in the predicted intervals. In the following sections, we motivate the method choices, explain our modifications to the established method in Stage ❶, and describe Stage ❸ and Stage ❹ in detail with examples.

6.3.1 *Motivations of Method Choices*

The choices for the time series representation and the search technique are based on the following motivations.

We opted for the state-of-the-art symbolic representation SAX for time series due to its two properties. On the one hand, the symbols in SAX's alphabet have an order, allowing a tolerance band needed for our subsequent search. For instance, we may define a tolerance bound "E-I" of four around the symbol "G". Symbolic representations of time series do not necessarily have an ordered alphabet. For example, SDA [7] and SSTS [219] use symbols to represent features like upward/downward/constant

trends rather than value ranges, resulting in unordered alphabets. On the other hand, SAX's numerosity reduction property (consecutively repeated symbols can be merged) [165] inspired us to the query expansion for duration-scaling-invariant search.

We decided on regex, because it naturally handles duration-scaling-invariant search, together with other possible tricks, from which distortion-invariant pattern search in time series may benefit. Furthermore, it is potentially very fast, because it stops matching immediately when the pattern partially mismatches. On the other hand, regex is a well-established technique for simple and robust usage. Lastly, `SAXRegEx` may also benefit from future development of regex.

6.3.2 *Symbolic Encoding: Representing Data for Text Retrieval*

We use the established time series representation SAX introduced in Section 6.2 to convert the numerical time series into strings. However, we modified this base representation for our needs in the task pattern search.

Query-Aware SAX. While it is common practice to conduct indexing (SAX) on the whole time series dataset, we fit SAX (parameters: bin size horizontally along the time axis and breakpoints along the value axis) with the query and then transform the time series dataset with the fitted SAX. We make such a modification because there is no clue about the appropriate bin size and breakpoints. Especially when the target patterns in the time series are quite small, either in terms of temporal length (relates to bin size) or value range (relates to breakpoints), compared to other events, or the background. This is exactly the case in our APST dataset. In this dataset, the "w"-shaped target patterns (a flag signal related to lambda control in the engine) are fairly small in terms of value range compared with the background. If applied to the whole time series dataset, SAX would have an insufficient resolution in the interesting value ranges or waste too many symbols in unimportant value ranges. In fact, [127, 272] proposed a similar practice, where the hash tables are tailored to the query, not the dataset. They call the new query-centric LSH query-aware and the database-centric one query-oblivious. Though hashing is needed every time the query changes, it significantly improves accuracy. On the other hand, the query-aware setting may raise concerns on its efficiency. However, SAX's execution time is actually negligible compared with regex's search time. In the speed benchmark, the execution time is end-to-end, including SAX's execution time.

Bin Size for Aggregation. For aggregation horizontally along the time axis, we do not set the bin size ι (how many time steps are merged to one symbol) of SAX directly. Instead, we set the final length of the SAX-encoded query $\widetilde{\phi}$, called word size [165]. Because the user may find it difficult to estimate how many time steps should correspond to one symbol. Whereas, it is relatively intuitive to say how many pieces/symbols along the time axis are needed to present the query. Furthermore, we visualize the reconstructed query from the symbolic representation to help the user set $\widetilde{\phi}$, as will be explained in Figure 6.4.2. By default, we set $\widetilde{\phi} = 20$, which is enough for all our datasets. ι is inferred with $\iota = \phi \operatorname{div} \tilde{\phi}$. This arrangement aims to have some control over the regex length because we find empirically that our regex does not scale well with its length (this drawback also limits the number of tracks). We leave a scalability test as future work.

Breakpoints for Quantification. We do not work on the assumption of the original SAX that the values within each track follow a normal distribution. Based on this assumption, the original SAX calculates quantiles to quantify the values and categorize them into symbols, so that each symbol has approximately the same number of values. This helps to zoom in on the value ranges where the values concentrate. The values in the tracks in many of our datasets are clearly not normally distributed. Therefore, we calculate the quantiles based on the true value distribution in each track, exactly as in Section 5.5.2. Again, we do not calculate the value distribution based on the whole time series dataset but on the query. Because we want to focus on the value range of interest, which is given by the query.

6.3.3 *Regex Extraction: Formulating Query for Pattern Description*

The first two stages of our processing pipeline typically leave repetitive symbols in the text sequences. They can be bundled with quantifiers. For instance, the string `ACCCCGGGBAAA` can be rewritten as `AC{4}G{3}BA{3}`. As we will see later, these quantifiers also enable horizontal-scaling-invariant search.

Generally, regex is meant for exact search instead of "fuzzy" search like pattern search in time series, which resembles more information retrieval, as mentioned in Section 6.2. It can only match patterns strictly satisfying the restrictions imposed by the regex. To allow fuzzy search, we add a tolerance band to the query with character classes analogous to the L^∞ norm. For instance, the previously encoded query `AC{4}G{3}BA{3}` can be further aug-

mented as `[A-B][B-D]{4}[F-H]{3}[A-C][A-B]{3}`. In this example, we allow a tolerance band with a two-symbol width.

6.3.4 *Query Expansion: Capturing Pattern Distortions for Retrieval Invariances*

We conduct query expansion to address the necessary retrieval invariance required in our use cases, particularly heterogeneous horizontal scaling and inter-track time shifts.

For heterogeneous horizontal scaling, we use value ranges rather than fixed regex quantifiers. Pertaining to our running example, we can further augment the query to `[A-B]{0,2}[B-D]{2,8}[F-H]{1,6}[A-C]{0,2}[A-B]{1,6}`. This modification captures similar patterns with a half to double duration. Moreover, it allows different fragments in the pattern to have different scaling factors, thus capturing heterogeneous horizontal scaling or time warping.

For inter-track time shifts, we use a different alphabet per track and intertwine the tracks to a single string. As shown in Figure 6.3 Stage ❷, we intertwine the query while taking time shifts between the shapes in different tracks into consideration. We can add this flexibility by introducing the wildcard character "." for "gaps". This way, the query ignores the potentially interfering context in each track and focuses only on the given shape in the track. As a result, the shape in a track within the query does not have to span the whole query length. Note in Figure 6.3, the term `(...){2,6}`, which captures the elastic gap between the shapes in Track 3 and in Track 1.

6.4 USER INTERFACE

We have designed a user interface to assist domain users, especially during query definition and parameter setting. It also enters the untouched area in VQS design, namely the definition of multivariate time-shifted query.

We derived our design from the requirements of our domain users, who request a tool with a streamlined workflow, little configuration overhead, and quick response. In line with the findings in [240], our domain users disapprove of visualizations for understanding the algorithm's internal mechanism. They are reluctant to learn the algorithm itself but appreciate intuitive visual assistance, especially during query definition and parameter setting.

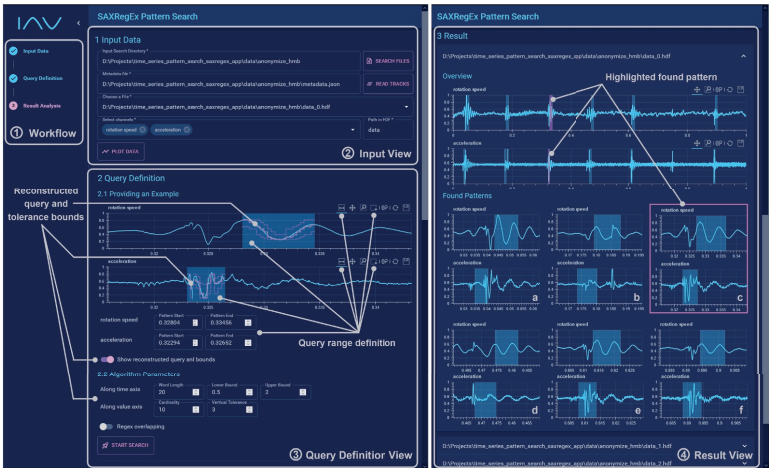


Figure 6.4: **User Interface.** assists the user with a streamlined workflow. It highlights the specification of track-wise time-shifted query and the on-the-fly visualization of the expanded pattern perceived by the algorithm with the current parameter setting.

We used React as the frontend library, BokehJS to render charts, and Django to access the algorithm presented in Section 6.3. In the following, we will describe individual features of the user interface and present a coherent workflow in a case study in Section 6.5.4.

6.4.1 *Sidebar and Input View*

To satisfy our workflow requirements, we chose a layout allowing the user to proceed sequentially through the workflow steps "Input Data", "Query Definition", and "Result Analysis" shown in the sidebar. The accomplished steps are in blue, in-processing steps in the highlighting color pink, as well as coming steps in gray, as shown in Figure 6.4 ①. In our proprietary system, more workflow steps are listed in the sidebar and `SAXRegEx`'s algorithm acts as the first quick search. We have removed preprocessing steps like dataset description, resampling, and steps after `SAXRegEx` like the accuracy-centered feedback-driven pattern search based on the user's feedback because they are outside the scope of this chapter. In the depicted user interface, the user needs to provide the dataset and relevant tracks in the *Input View* in Figure 6.4 ②.

6.4.2 *Query Definition View*

As the primary visual component of the user interface, the *Query Definition View* in Figure 6.4 ③ assists the user in specifying the desired query and setting the algorithm parameters.

Providing An Example

As explained in Section 2.9, we opted for the query-by-example approach. Based on this premise, we found the most efficient way to define an inter-track time-shifted query is to approximately mark one interval for all tracks simultaneously first, then fine-tune the interval in each track individually. The process is depicted in Figure 6.5. To define an approximate query without time shifts, the user activates the box select tool . It is configured to allow marking a box along the time axis, and the selection will be synchronized across all tracks. A box is technically a Bokeh range tool , which is automatically activated in all tracks after marking the first query without time shift. This allows the user to move the box and fine-tune its left and right time stamps in each track, i.e., to modify the spread of a pattern. This design emphasizes the relationship between the patterns in different tracks. Compared with drawing boxes in each track individually, the user gets a sense of the positions of the patterns in different tracks relative to each other, as the user moves the box, that was marked for the pattern in another track, to align its left edge with the start time of the pattern in the current track. Likewise, the user feels the difference between pattern durations when adjusting the box size from the length of the first pattern to that of the next. If there are no inter-track time shifts between the patterns in several tracks, the user does not need to mark the same time interval multiple times. The edge positions or the time intervals are listed additionally as numbers in the input fields below.

Algorithm Parameters

The user can tune six parameters shown in Figure 6.6. They control the resolutions, the query expansion, and thus the fuzziness or strictness of the search. They have default values explained in the next paragraphs, which fit our use cases. The user may begin with the default and adjust them if necessary.

The parameters introduced by SAX, namely word size ϕ (called word length in the UI) and cardinality, regulate the horizontal

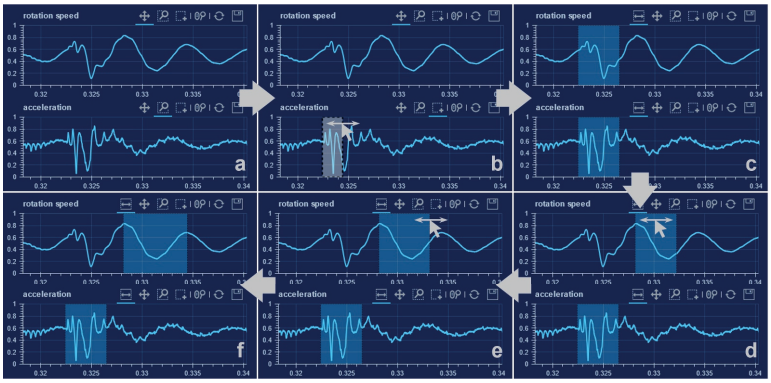


Figure 6.5: **Labeling an Example as the Query.** The user draws a box in any track to define a query without inter-track time shifts, then fine-tune the boxes in different tracks individually by moving it or adjusting its left and right edges.

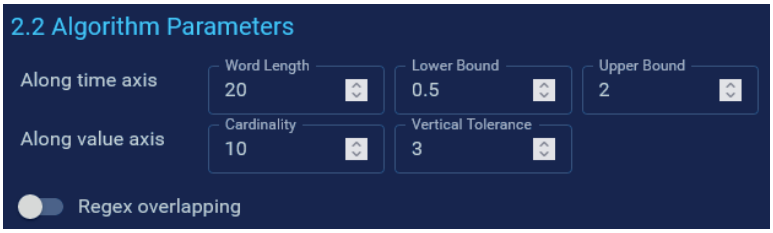


Figure 6.6: **Algorithm Parameters.** They influence the strictness of the search by controlling the resolutions and query expansion.

(along the time axis) and vertical (along the value axis) resolutions, respectively. For instance, the default $\phi = 20$ explained in Section 6.3.2 encodes the pattern in the first track to a string of length 20. The patterns in other tracks are converted to strings with proportional lengths. We find a word size 20 usually sufficient. The default cardinality 10 means discretizing the values in each track with 10 symbols. According to the experiments in [165], cardinality 10 is enough.

The lower and upper bounds along the time axis determine the horizontal elasticity of the query. The default lower bound 0.5 means that the lengths of the found patterns are not allowed to be shorter than half of the query length, while the default upper bound 2 means that the lengths of the found patterns cannot exceed the doubled length of the query. Namely, $\forall F \in \mathbb{F}, 0.5\phi \leq |F| \leq 2\phi$. These two parameters are task-dependent, but users can often set them before starting the search. Note that the scaling

can be heterogeneous, i.e., some fragments in the query can be down-scaled while others up-scaled.

Vertical tolerance refers to the allowed deviation of symbols. For example, the deviation between A and D is 3 in the symbolic space. It resembles the Chebyshev distance, or L^∞ norm, and plays a filtering role analogous to a relevance/similarity/-confidence threshold or the desired number of found patterns. It is difficult to estimate this parameter a priori. The best setting is affected by how similar the target patterns are compared to the query (homogeneity within the positive class) and how dissimilar the rest of the data are with the query (difference between positive and negative classes). Users may need to adjust this parameter to find fewer or more patterns.

The last parameter regex overlapping is boolean and controlled by a switch. It handles cases when the patterns can overlap, like Pattern 3 and Pattern 4 in Figure 6.10 (last row for `SAXRegEx`). The same pattern will likely be found multiple times if this feature is activated. Nonetheless, this feature is less used and deactivated by default.

To assist users with parameter tuning, we overlay the query with its reconstruction from the symbolic space and a pair of bounds visualizing the most dynamic parameter, namely vertical tolerance if the user activates the switch "Show reconstructed query and bounds", as shown in Figure 6.7. The reconstruction and the bounds will update automatically when the user adjusts the query boxes, the word size, the cardinality, or the vertical tolerance. Figure 6.8 shows the resulting reconstruction and tolerance bounds with different parameter settings.

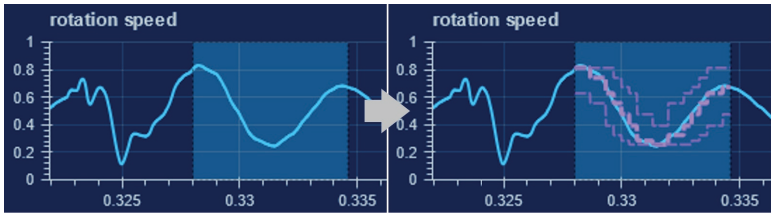


Figure 6.7: **Query Reconstruction and Tolerance Bounds.** The original query is overlaid with its reconstruction from the symbolic space and the tolerance bounds, if the user activates the switch "Show reconstructed query and bounds". They assist the user with parameter setting.

The reconstruction and tolerance bounds are generated by converting a symbol with the mean of its lower and upper breakpoints. A breakpoint is the borderline between two symbols. If

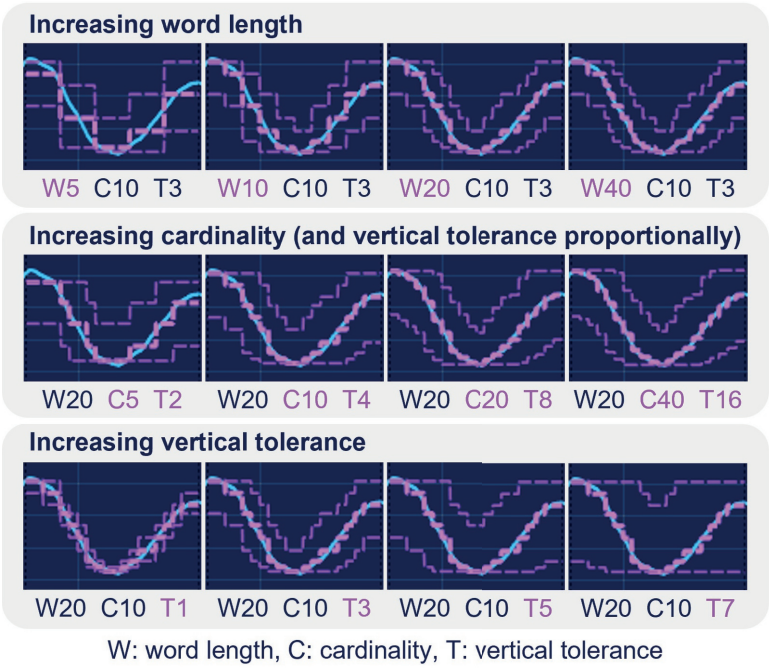


Figure 6.8: **Query Reconstruction and Tolerance Bounds With Different Parameter Settings.** The query reconstruction and the tolerance bounds are updated whenever the original query or the parameters change. They help the user gain an intuition of the effects of the parameter setting before starting the search.

a symbol does not have a lower or upper breakpoint, we use the minimum or maximal value in the track of the query instead. For instance, a track in the query has a minimum value of 0 and a maximal value of 10. Supposing that we use A, B, and C to discretize the tracks, the breakpoint between A and B is 4, and between B and C 6. Then, the reconstructed value for A is $(0+4) \div 2 = 2$, for B is $(4+6) \div 2 = 5$, and for C is $(6+10) \div 2 = 8$. Because we use the mean of the breakpoints rather than the breakpoints themselves, the tolerance bounds appear visually tighter than the actual bounds. Nonetheless, the imprecise visualization of the tolerance bounds helps the user obtain an intuition of the vertical tolerance before starting the search.

6.4.3 *Result View*

The *Result View* in Figure 6.4 ④ shows all data files and the found patterns in each data file. For each input data file, we show a large overview plot and small plots of each individual found pattern beneath the overview plot. An overview plot shows the entire data in a data file and the found patterns in the data file. The overview plot and the found patterns can communicate with each other. Clicking a plot for an individual found pattern will highlight the plot with a pink frame. The corresponding pattern in the overview plot will also be highlighted in pink. Likewise, clicking a found pattern in the overview plot will trigger the same highlighting in the overview plot and the plot for the individual found pattern.

6.5 EVALUATION

We evaluated the accuracy and speed of `SAXRegEx`'s algorithm as well as the usability of its VQS.

6.5.1 *Experiment Setup*

For algorithm evaluation, we used eight labeled datasets, including the labeled four datasets in Section 5.5.1. The rest four datasets were

- `APST`: the relative air-fuel ratio in a petrol engine measured by the engine control unit.
- `Cable cutter`: the simulated power consumption of a cable cutting machine.
- `CAN 1 and CAN 2`: synthesized CAN bus data.

These four datasets are proprietary house-internal datasets that cannot be published. The `CAN 1` and `CAN 2` datasets contain patterns with heterogeneous horizontal scaling or inter-track time shifts. They are mainly automotive measurement data provided by different engineers in our industrial collaborator IAV. While the datasets show different characteristics like the number of tracks, the pattern shape, and the extent of duration variation, by no means do they cover all possible cases, like high-frequency audio patterns.

For benchmarking `SAXRegEx`, we used the same four representative and state-of-the-art methods in Section 5.5.1, namely,

`correlation`, `DTW`, `MASS`, `SAX`. Finally, we conducted a case study to demonstrate, how `SAXRegEx`'s VQS assists users in applying the algorithm.

We conducted all experiments with the same hardware and system setting described in Section 5.5.1.

For regex, we used `regex.compile(query).finditer(time_series)` from `https://github.com/mrabarnett/mrab-regex`. We used this library because the parameter `overlapped` can be set to `True` to find overlapping patterns, though this feature is less used. Whereas, the native Python library `re` does not support this feature.

6.5.2 *Accuracy Benchmark*

We benchmarked the accuracy of all methods quantitatively and attempted to explain the pitfalls of the baseline methods with a visual inspection. Like in Section 5.5.2, we used not only time-step-based metrics introduced in Section 4.2 including accuracy, balanced accuracy, precision, recall, F1-score, but also pattern-interval-based metrics including AP30 and AP50 elaborated in Section 4.3.

`SAXRegEx` enjoys retrieval invariance for inter-track time shifts and can locate varied intervals in different tracks within a found pattern. Namely, rather than starting and ending at the same time, the shape in each track can have different start and end time. For a consistent accuracy benchmark with other methods, we use the earliest start time and the latest end time within a pattern as the found pattern interval for `SAXRegEx`. To find horizontally scaled patterns, we used a set of eight sliding windows of exponentially increasing window lengths to deal with the scaling problem for the benchmark methods.

`SAXRegEx` does not calculate relevance/confidence/similarity. However, the pattern-interval-based metric AP requires a relevance score for each found pattern. To solve this problem, we calculated a pseudo-relevance score for the found patterns. The idea was that we ran `SAXRegEx` with a range of tolerance bands of increasing widths (the allowed number of symbol deviations explained in Section 6.3.3). Each pattern was found starting with a tolerance band. The narrower the tightest band for finding a pattern, the higher the relevance score. We inverted and normalized the band width as the pseudo-relevance score. As mentioned Section 2.5, the absolute similarity value is not important for the metrics, as long as the similarity ranking between the found patterns remains.

Dataset	Method	Accuracy	Balanced accuracy	Precision	Recall	F1 score	AP30	AP50
APST	Correlation	0.99 ± 0.01	0.98 ± 0.03	0.80 ± 0.16	0.97 ± 0.07	0.88 ± 0.12	0.89 ± 0.25	0.89 ± 0.25
	DTW	0.98 ± 0.02	0.98 ± 0.03	0.65 ± 0.22	0.98 ± 0.04	0.76 ± 0.19	0.87 ± 0.30	0.87 ± 0.30
	ED (MASS)	1.00 ± 0.01	0.98 ± 0.03	0.87 ± 0.07	0.97 ± 0.07	**0.92 ± 0.04**	**1.00 ± 0.00**	**1.00 ± 0.00**
	SAX	0.99 ± 0.02	0.97 ± 0.03	0.57 ± 0.28	0.96 ± 0.07	0.69 ± 0.23	0.57 ± 0.33	0.57 ± 0.33
	SAXRegEx	0.88 ± 0.27	0.92 ± 0.14	0.47 ± 0.37	0.95 ± 0.10	0.55 ± 0.36	0.64 ± 0.39	0.64 ± 0.39
Cable Cutter	**Correlation**	0.98	0.93	0.79	0.86	**0.82**	**0.89**	**0.56**
	DTW	0.94	0.96	0.48	0.98	0.64	0.84	0.19
	ED (MASS)	0.93	0.74	0.36	0.52	0.43	0.27	0.18
	SAX	0.97	0.85	0.74	0.71	0.72	0.69	0.31
	SAXRegEx	0.96	0.92	0.55	0.87	0.67	**0.89**	0.55
Deep Valve	Correlation	0.89	0.73	0.88	0.47	0.61	0.82	0.29
	DTW	0.79	0.85	0.46	0.94	0.62	0.46	0.19
	ED (MASS)	0.90	0.76	0.88	0.53	0.67	**0.85**	0.28
	SAX	0.96	0.95	0.87	0.93	**0.90**	0.73	**0.66**
	SAXRegEx	0.88	0.86	0.62	0.84	0.71	0.83	0.50
EEG Eye State	Correlation	0.79	0.78	0.76	0.77	0.76	0.57	0.40
	DTW	0.80	0.80	0.73	0.87	**0.79**	0.57	0.43
	ED (MASS)	0.69	0.70	0.63	0.77	0.69	0.58	0.36
	SAX	0.65	0.66	0.58	0.80	0.67	0.40	0.37
	SAXRegEx	0.68	0.70	0.59	0.92	0.72	**0.62**	**0.62**
Filling Prediction	Correlation	0.97	0.80	0.91	0.61	0.73	0.88	0.88
	DTW	0.95	0.89	0.57	0.82	0.67	0.75	0.75
	ED (MASS)	0.97	0.80	0.91	0.61	0.73	0.84	0.84
	SAX	0.97	0.92	0.69	0.87	**0.77**	**0.95**	**0.95**
	SAXRegEx	0.90	0.81	0.37	0.72	0.49	0.58	0.47
Variable Displacement	**Correlation**	1.00 ± 0.00	0.96 ± 0.02	0.98 ± 0.01	0.92 ± 0.03	**0.95 ± 0.01**	**0.88 ± 0.33**	**0.88 ± 0.33**
	DTW	1.00 ± 0.00	0.91 ± 0.05	0.95 ± 0.05	0.82 ± 0.10	0.88 ± 0.07	0.83 ± 0.32	0.81 ± 0.33
	ED (MASS)	1.00 ± 0.00	0.96 ± 0.01	0.97 ± 0.04	0.92 ± 0.03	0.94 ± 0.02	0.87 ± 0.33	0.87 ± 0.33
	SAX	1.00 ± 0.00	0.95 ± 0.03	0.96 ± 0.03	0.90 ± 0.06	0.93 ± 0.04	0.84 ± 0.33	0.84 ± 0.33
	SAXRegEx	1.00 ± 0.00	0.95 ± 0.03	0.96 ± 0.06	0.89 ± 0.07	0.92 ± 0.05	0.83 ± 0.32	0.83 ± 0.32
CAN 1 (with heterogeneous horizontal scaling)	Correlation	0.73	0.70	0.68	0.93	0.79	0.79	0.75
	DTW	0.95	0.95	0.93	1.00	0.96	**1.00**	**1.00**
	ED (MASS)	0.73	0.70	0.68	0.93	0.79	0.90	0.78
	SAX	0.76	0.75	0.74	0.86	0.80	0.90	0.81
	SAXRegEx	0.99	0.99	0.99	0.99	**0.99**	**1.00**	**1.00**
CAN 2 (with inter-track time shifts)	Correlation	0.88	0.88	0.82	0.99	0.90	0.97	0.97
	DTW	0.77	0.77	0.80	0.75	0.77	0.89	0.70
	ED (MASS)	0.88	0.89	1.00	0.78	0.87	0.94	0.94
	SAX	0.81	0.81	0.76	0.93	0.84	0.84	0.84
	SAXRegEx	1.00	1.00	1.00	1.00	**1.00**	**1.00**	**1.00**

Figure 6.9: **Accuracy Benchmark.** Best performance of all methods. We highlighted the best F1-score, AP30, AP50, and method for each dataset in bold and red. Each method performed well in some datasets, while poorly in the other.

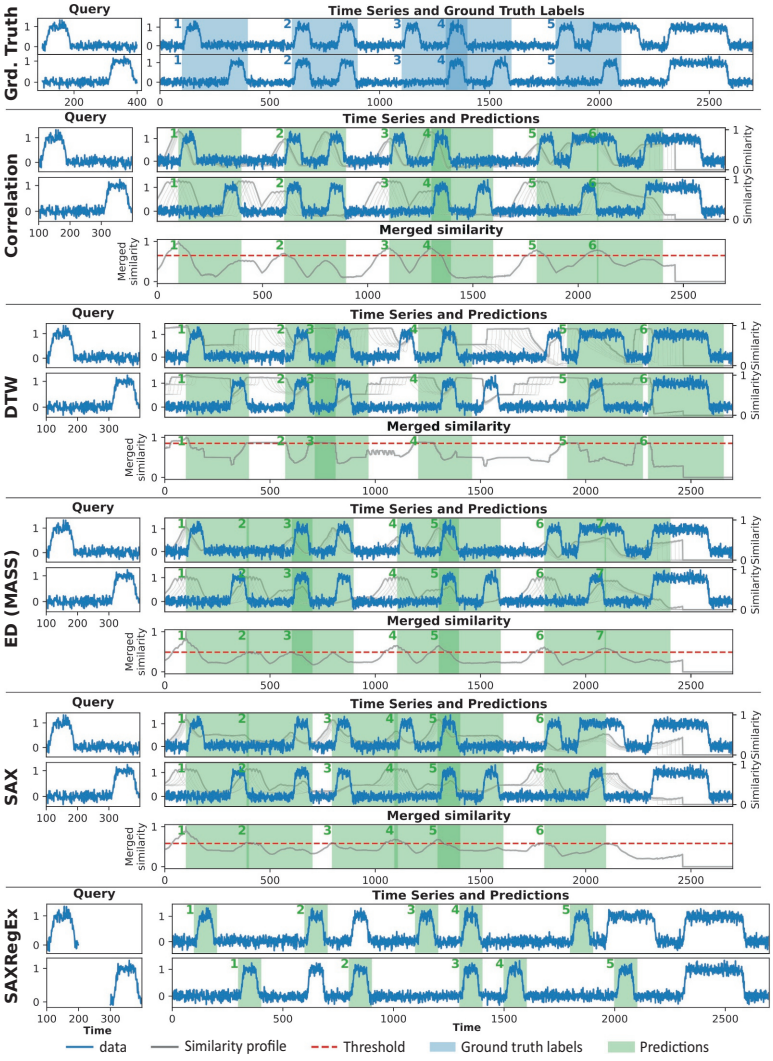


Figure 6.10: **Visual Inspection.** Result on the dataset CAN 2. SAXRegEx perfectly finds inter-track time-shifted patterns, while the benchmark methods suffer from false negatives and false positives.

The table in Figure 6.9 depicts our evaluation metrics (Chapter 4) for all methods on various datasets. The result suggests that different data favor different methods and no method constantly outperforms the other, confirming the finding in [65]. However, SAXRegEx can better capture heterogeneous horizontal scaling and inter-track time shifts. We could observe this specifically on the last two datasets in Figure 6.9.

We plotted the found patterns and ground truths with all methods on all datasets for a visual inspection. Due to space limit, we show an example in Figure 6.10. From the visual inspection, we could infer the pitfalls of the methods.

Figure 6.10 shows the query on the left side and the time series database on the right. The blue curves stand for the original data. The benchmark methods calculate similarity profiles, which are delineated with the gray curves. The similarity profile at a time step for a track records the similarity of a potential predicted interval starting at the time step. The length of the interval corresponds to the sliding window size. As explained in this subsection, we used a range of sliding windows with increasing window sizes. The similarity profile keeps records of the window size with the highest similarity for a time step. The similarity profiles are averaged over all tracks and the merged similarity is shown in a separate sub-figure (as an additional track for each method) under all time series tracks. The user can tune a threshold marked as red lines in the plots. If a similarity peak reaches the threshold, a predicted interval denoted as a green area is found. We conducted NMS to remove intervals with large overlapping. SAXRegEx works differently as the benchmark methods in that 1) it does not have similarity profiles as explained in this subsection, thus missing the additional track with the merged similarity profile and the dashed red line for the threshold; 2) it accepts inter-track time-shifted patterns as the query and can find such patterns in the time series (please note the difference between the query for SAXRegEx and the benchmark methods).

We found that the benchmark methods spread focus evenly over all time steps rather than critical transitions (usually large steps/ramps). This led to, e.g., the false positive Pattern 7 with ED. Pattern 7 did resemble the query in terms of the general shape, but missed an upward ramp at the beginning in the first track and a downward ramp at the end in the second track. Next, the benchmark methods took the context of the pattern shape in each track into account. Consequently, the search could be misled by a changing context, as implied by the missing / ill-detected ground truth 2 for all benchmark methods. In this case, the target pattern

contained a distracting plateau in each track. Finally, while DTW captured warping or time shifts within a track well, as proven by the performance on the CAN 1 dataset in Figure 6.9 (and in the unshown visual inspection for the dataset), it had a hard time when it came to time shifts between tracks. In fact, it tended to regard inter-track time-shifted patterns as not shifted, as indicated by Prediction 2 and 3. In the presence of such distortions, SAXRegEx outperformed the benchmark methods and fulfilled our needs. On the flip side, SAXRegEx may find excessively long patterns when they do not have unclear boundaries, as in the EEG Eye State dataset. Because the quantifiers in regex are greedy by default. Thus, the regex tries to match the longest possible patterns.

6.5.3 *Speed Benchmark*

We calculated the execution time of all methods on all datasets. Every experiment was repeated five times.

SAX's first step PAA aggregates several time steps, calculates their average, and merges them into one, which will be transformed into one symbol in subsequent steps. As mentioned in Section 6.3.2, SAXRegEx down-scales the query of a certain length to 20 symbols because regex does not scale well with the pattern length. Accordingly, the time series is down-scaled with the same bin size horizontally along the time axis. This reduces the data volume and accelerates the subsequent processing. To ensure a fair speed comparison, we conducted PAA with the same resolution for other benchmark methods to accelerate them. This only affected the speed benchmark. In the accuracy comparison, we did not conduct PAA for the benchmark methods. Rather, we used the finest possible resolution to achieve the highest possible accuracy for the benchmark methods.

Figure 6.11 shows the result of the speed benchmark. In Figure 6.12, we report on the relative performance gain compared to DTW.

As shown in Figure 6.12, SAXRegEx is nearly 50 times faster than DTW and 4.6 times faster than MASS (please note the clarification in Section 5.5.1 that the Python version of MASS may not reflect its true potential), the so far fastest similarity search tool for time series retrieval. We attribute this speed boost to two reasons. First, SAXRegEx naturally captures horizontally scaled patterns. In contrast, the benchmark methods use eight sliding windows with varying window lengths, costing roughly 8 time the execution time compared with one sliding window with the same

Method	APST	Cable Cutter	Deep Valve	EEG Eye State	Filling Prediction	Variable Displ.	CAN 1	CAN 2
DTW	24.54±2.29	43.56±3.39	9.98±0.37	2.00±0.57	1.36±0.07	94.66±5.25	0.320 ± 0.031	0.410 ± 0.028
Correlation	27.90±3.25	55.09±4.11	12.35±1.24	3.61±0.38	1.41±0.03	107.44±7.57	0.644 ± 0.636	0.542 ± 0.288
SAX	2.59±0.21	7.89±1.83	1.26±0.04	0.20±0.01	0.23±0.01	8.28±0.83	0.222 ± 0.016	0.282 ± 0.006
ED (MASS)	15.05±0.87	27.18±1.48	6.19±0.29	1.23±0.30	0.83±0.06	48.36±2.36	0.062 ± 0.003	0.074 ± 0.005
SAXRegEx	0.65±0.03	4.39±0.55	0.22±0.01	0.05±0.00	0.04±0.01	1.36±0.16	0.022 ± 0.003	0.022 ± 0.007

Figure 6.11: **Speed Benchmark.** Unit: seconds. The methods/rows are arranged in descending order of elapsed time. We repeated the same experiment five times and recorded their mean as well as standard deviation. `SAXRegEx` outperformed the other methods for all datasets.

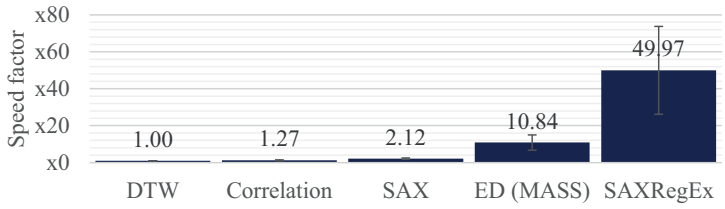


Figure 6.12: **Speed Benchmark in Factor.** The mean execution time of each method was divided from that of `DTW`. `SAXRegEx` was on average 50 times faster than `DTW` in our experiments. This figure was derived from Figure 6.11.

method. However, even reducing the sliding window numbers to four, which is already too coarse, `SAXRegEx` still outperformed all benchmark methods. Second, regex terminates a matching as soon as the search engine notices a partial mismatch.

The speed ranking of `correlation` and `DTW` is inconsistent among all datasets. Because the synthesized datasets CAN 1 and CAN 2 are fairly small. Therefore, the searching time was not dominant during the processing. The used `DTW` library seems to have a better implementation and outperforms `correlation` implemented by ourselves. However, as Table 9 shows, when the query length grows, its relatively inferior scalability ($O(dn\phi^2 m)$) compared with that of `correlation` ($O(dn\phi m)$) starts to take its toll, where d stands for the number of tracks, ϕ the query length, n the length of the time series, and m the number of sliding windows of different sizes for horizontally scaled patterns. Due to this reason, we excluded the datasets CAN 1 and CAN 2 when making Figure 6.12, where we averaged the results on all other datasets.

6.5.4 *Case Study*

In this section, we will demonstrate a use case with a real-world dataset but without real users. Please also refer to the video https://www.youtube.com/watch?v=pb8ZUCjl_8Q for details of interactions.

We use the `Variable Displacement` dataset for demonstration. Unfortunately, it is proprietary and cannot be published. The dataset contains measurements of a car switching between the four-cylinder and two-cylinder drive modes. The used tracks are domain-specific extracted features but roughly correspond to the engine rotational speed and acceleration. The time spans and value ranges of all tracks were normalized to the range $[0, 1]$ for (partial) anonymization. This is required when using the screenshots, but not a prerequisite of `SAXRegEx`.

When the user starts the user interface, only the *Input View* is visible. After providing input data and clicking the button "Plot data", the *Query Definition View* appears. It has a line chart plotting the chosen tracks in the given data file and input fields populated with default algorithm parameters.

The user interacted with the line chart by panning and zooming to take a quick look at the data and spotted a pattern that was characteristic of the traditional event between the two drive modes. As shown in Figure 6.5.a, there was a period with an abrupt increase of amplitude in the second track (acceleration). This turbulence could be the cause of the uncomfortable jerks that the driver sensed, and that our engineers were investigating. After the acceleration turbulence, the engine rotational speed track shows a damped oscillation. The user clicked and dragged a box with the Bokeh box select tool to label it (Figure 6.5.b). There appeared boxes with the same labeled range in both tracks (Figure 6.5.c). Just for this use case, the user could use this label as the query. Let us assume that the user was interested in the pattern in the first track shown in Figure 6.5.f. The user moved the box in the first track so that the left edge of the box matched the left side of the desired pattern (Figure 6.5.d). Next, the user adjusted the right edge of the box (Figure 6.5.e). The corresponding Bokeh range tool is activated automatically after the first marking to allow these interactions. When moving the box in the first track, the user sensed that the pattern in the first track lagged behind for a certain time; when adjusting the right edge of the box, the user noticed that the pattern in the first track lasted longer to some extent.

Then, the user moved on to the algorithm parameters shown in Figure 6.6. The user turned on the switch "Show reconstructed query and bounds" and noticed the reconstructed query together with the tolerance bounds, as in Figure 6.7. Without a deep understanding of the algorithm, the user could interpret the thicker pink line in the middle, right on the blue query curve, as a reflection of how the algorithm perceived the pattern and the thinner pink curve pairs as a tolerance band. The user found the resolution sufficient and decided not to change the word size or the cardinality. The user also kept the lower bound 0.5 and upper bound 2 for horizontal scaling. So the query or any part of it could be compressed or stretched maximal with the factor 2. The user was unsure about the vertical tolerance and tried multiple values (last row in Figure 6.8). With the help of the visualized tolerance bound, the user found the preset reasonable. Finally, the user clicked the "Start search" button. The *Result View* appeared and eight panes corresponding to eight data files appeared in a second. The user opened one pane, skimmed the six found patterns, clicked interesting ones, and inspected the clicked ones highlighted in the overview chart. We show a mediocre result in Figure 6.4.4 as a common first outcome. Please refer to the video `https://www.youtube.com/watch?v=pb8ZUCjl_8Q` for a better result.

In Figure 6.4.4, the user noticed that some patterns indeed resembled the query, like Pattern c, e, and f. Pattern b and d passed for the desired pattern to some extent, but might indicate a different event because the acceleration shows different dynamics. Pattern a indicates a potentially desired pattern, but the found interval in the acceleration track does not align well with the pattern. If only in search of some samples of a target event, the user might check several files and collect the desired patterns. If aiming at all similar patterns in all data files with high recall and precision, the user might take further steps. Specifically, the user might add more relevant tracks (but possibly without visually similar patterns), provide feedback on the found patterns, run a supervised-learning-based search with the labeled findings, and finally conduct domain-specific analysis with the refined result, which will be the topic in the next two chapters.

6.6 DISCUSSION

Our industrial collaborator IAV applies `SAXRegEx` to data from engine control units, transmission control units, and CAN bus, where heterogeneous horizontal scaling of the pattern and inter-

track time shifts occur regularly. It satisfactorily meets automotive engineers' flexible needs for prompt search.

One shortcoming of SAXRegEx is the neglect of vertical translation (bias) and vertical scaling (amplitude scaling) of the pattern (please refer to Section 2.7 for an introduction to various retrieval invariance). The user should ensure that they do not affect the patterns in the database. Ideally, query expansion should be able to cover them.

Notwithstanding the limitation above regarding query expansion, there is still potential for further augmentation of the regex. For instance, groups in regex can be used to denote a pattern with multiple phases; positive/negative lookarounds allow retrieving patterns with/without certain preceding or following patterns; conditionals in regex allow a part of the pattern to vary according to another part. However, their usefulness and effectiveness remain to be examined.

SAXRegEx does not calculate similarity profiles. On the one hand, it accelerates the algorithm because the regex search stops immediately when a part of the pattern does not match the query. On the other hand, it is not possible to tune a similarity threshold to improve the found patterns. Therefore, it functions better as the first quick-and-dirty search. The user can improve the search results with fine-grained methods, especially model-based pattern search methods, e.g, techniques from Chapter 7 and Chapter 8, with user-labeled found patterns from SAXRegEx.

We plan to investigate the time complexity of SAXRegEx and carry out experiments to study its scalability with increasing query sizes (regex length) and the increasing number of tracks in a contour plot. We would also like to examine the degree of distortions that SAXRegEx can bear.

We use one vertical tolerance for all tracks, providing that the characteristics of the tracks do not differ significantly. However, this assumption does not always hold. We could allow an individual vertical tolerance for each track but still need a way to avoid overwhelming the user with too many parameters to tune.

SAXRegEx lacks the steerability to meet C9 Subjective and Task-Dependent Similarity and Relevance in Section 1.2. We imagine that there is room for adaptiveness. For instance, we may refine the SAX parameters like bin size and breakpoints in each track based on track importance inferred from user feedback, exactly the same as the feedback mechanism used in Section 5.3.3 for PSEUDo. Moreover, the quantifiers and character classes are also potential trainable parameters. A heuristic is to tighten the tolerance band described by the character classes for pattern frag-

ments with low variance in positively labeled patterns, again, similar to PSEUDo's technique.

Overall, our evaluation of the user interface for SAXRegEx is incomplete. We will extend the user interface in Chapter 7 and Chapter 8, and evaluate it more rigorously with case studies and expert studies.

6.7 CONCLUSION

In this chapter, we have proposed SAXRegEx, an algorithm for MTS pattern search based on SAX, regex, and query expansion. It excels in capturing distorted patterns of various types. In particular, it is ideal for retrieving patterns scaled horizontally along the time axis heterogeneously or patterns showing inter-track time shifts, while remaining efficient. We have also presented a user interface featuring multivariate query definition with inter-track time shifts and parameter setting, assisting users in applying the algorithm. SAXRegEx helps automotive engineers quickly find patterns related to various events in the measurements. Nonetheless, the method itself is not limited to any domain-specific prerequisites and can be used in other domains as well. In the future, we plan to extend the evaluation with a scalability test for the algorithm and expert studies for the user interface as well as the overall VQS.

7

A MODEL-AGNOSTIC FRAMEWORK FOR MACHINE-LEARNING-BASED TIME SERIES PATTERN SEARCH

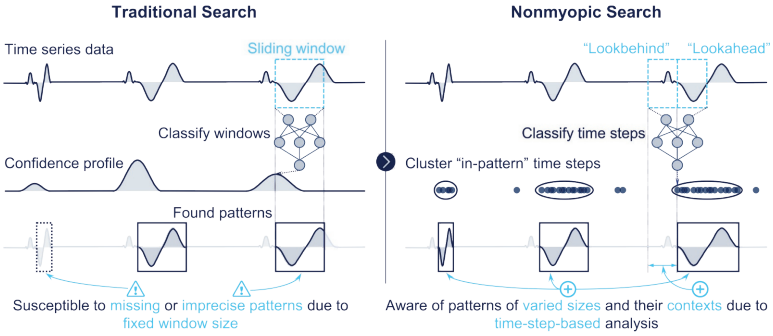


Figure 7.1: **Traditional Search vs. Nonmyopic Search.** We propose nonmyopic search for model-based time series pattern search. The traditional pattern search in time series scans the time series data with a sliding window to calculate a relevance profile, where top local maxima correspond to the found patterns. The traditional method has problems with patterns of variable duration and is oblivious to the pattern context. We propose nonmyopic search. It classifies whether each time step is in a target pattern or not and clusters the "in-target-pattern" time steps to retrieve pattern intervals. Nonmyopic search finds patterns of variable lengths in one scan and is aware of the pattern context.

In this chapter, we present nonmyopic [1] search for model-based pattern search. Instead of classifying time series patterns/windows directly, it advocates the classification of time steps first. This change brings multiple benefits, especially duration-

1 According to Oxford Learner's Dictionary, the word "myopic" means "unable to see things clearly when they are far away" (https://www.oxfordlearnersdictionaries.com/definition/english/myopic?q=myopic). It is synonymous with "short-sighted". We add the prefix "non-", which is seen in many works [118, 133, 151], though the meaning of "nonmyopic" differs in each use case. We call our algorithm "nonmyopic" because it sees patterns of various sizes in one scan and even peeks into the pattern context.

scaling-invariant search and context-awareness. Ultimately, nonmyopic search improves accuracy and speed simultaneously. [2]

7.1 INTRODUCTION

The previous chapters Chapter 5 and Chapter 6 have primarily dealt with rule-based pattern search. While they are fast and interpretable, they have limited capability in capturing the subjective and task-dependent notion of pattern relevance, which is the strength of (machine-learning-)model-based pattern search. However, model-based pattern search suffers from two weaknesses, namely, fixed retrieved pattern lengths and context-obliviousness, as explained in Section 2.6. In accordance, we formulate the research question for this chapter: **How to retrieve patterns with variable durations in MTS in model-based pattern search while taking the pattern context into consideration?**

On one hand, this problem is interesting and important, because as explained in Section 2.6, fixed retrieval pattern length requires scanning the data with the model in different scales, costing multiple times of execution time; and the context-obliviousness hinders retrieval accuracy. On the other hand, this problem is challenging, because most machine learning models except sequence models like LSTM have a fixed input vector size, e.g., the number of input neurons in a neural network. While there are exceptions for rule-based pattern search, existing methods for model-based pattern search invariably scan time series data with a sliding window and classify the pattern currently in the window, making the weaknesses nearly inevitable.

We propose a novel data processing pipeline for model-based pattern search in MTS. Instead of classifying the patterns / time series windows as relevant or not, which is referred to as sequential scanning in Section 2.6, nonmyopic search classifies whether each time step in the time series is located in a target pattern or not. Then, it merges the "in-target-pattern" time steps into intervals of found patterns with density-based clustering. We call this approach "nonmyopic search", because, as we will show, during the search it can detect patterns of different lengths and even peek into their context. Namely, this improvement overcomes the two weaknesses associated with sequential scanning.

Nonmyopic search works theoretically with all classification models and does not require specific transformations of the

2 This chapter is mainly based on our ongoing publication PB6 with textual modifications for a coherent information flow.

time series. Experiments showed that nonmyopic search outperformed traditional search in retrieval accuracy and speed. To be precise, it increased retrieval accuracy by 0.19 and decreased execution time by 68% according to our experiments (with the best model in traditional search, averaged over all use cases). We extended the user interface in Section 6.4 and named the overall VQS `NOOPS`, Nonmyopic Multivariate Time Series Pattern Search.

7.2 RELATED WORK

The nonmyopic search put forward in this chapter highlights the combination of time-step-based classification and density-based clustering. It can be regarded as a novel traversing approach for time series, which breaks the curse that model-based pattern search relies on sequential scanning. Please refer to Section 2.6 for an introduction to traversing approaches for time series. Next, we will review methods for time-step-based classification and clustering.

Time-Step-Based Classification. Time-step-based classification is a relatively established task in time series analysis. Differing from time series classification mentioned in Section 2.2, which classifies whole time series, time-step-based classification classifies each time step in a time series. However, we have not nominated it in Section 2.2, because until now, we have only seen this task for a single dataset, i.e., the famous `EEG Eye State` dataset [221, 222], where algorithms classify whether the subject's eyes are open or closed at each time step. Researchers have proposed numerous classification methods and reached nearly perfect results [144, 195, 280]. This success inspired us to transfer the technique to pattern search in time series.

Time-Step-Based Clustering. Time series clustering is also a well addressed task in time series analysis, as Section 2.2 describes. Classic time series clustering takes a set of time series as input and divide them into several groups, each is a subset containing multiple time series [5, 121]. In our time-step-based clustering, the input is a set of (relevant) time steps, which are clustered based on their index/time along the unidimensional time axis. Methodologically, this time-step-based clustering resembles time series segmentation, where a time series is split into consecutive segments, as mentioned in Section 2.2.1. Time series segmentation can be formulated as clustering **all** time steps in a time series **based on the value(s)** at the moment of each time stamp. From this clustering-based perspective, time series segmentation may

begin with the whole time series and split it (the top-down approach akin to partition-based clustering) [84, 108], or begin with each time step as a cluster and merge them (the bottom-up approach akin to hierarchical clustering) [16]. Otherwise, time series segmentation may scan the time series for change points with a sliding window, which is not from the clustering-based perspective [57, 168]. Our time-step-based clustering clusters only **some (generally a small portion of relevant)** time steps in a time series; and it is **based on the position/index/time** of each time step along the time axis, rather than the value(s) at each time step. From this perspective, density-based clustering is the most suitable clustering method in our use case.

7.3 METHOD

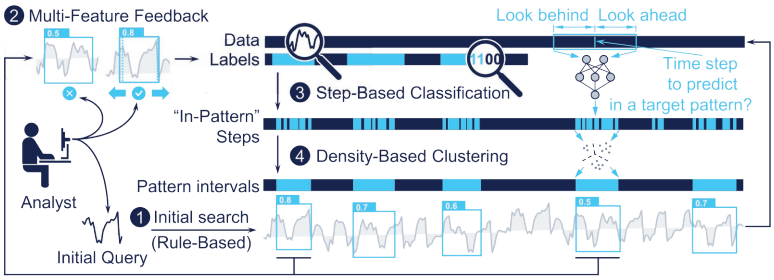


Figure 7.2: **NOOPS's Data Processing Pipeline.** NOOPS combines time-step-based classification and density-based clustering to retrieve patterns of variable duration and be aware of their context. Moreover, NOOPS extends traditional binary relevance feedback to positions and sizes of the found patterns, improving retrieval accuracy even further, which is the topic of Chapter 8.

We propose a four-stage data processing pipeline for feedback-driven pattern search in MTS, illustrated in Figure 7.2.

Stage ❶ searches for the query Q based on an established fast rule-based method MASS to get initial results. Although our work focuses on feedback-driven pattern search, we still need this stage because there are usually insufficient labels at the beginning to train a model. If the found patterns in this stage fail to reach the user's desired retrieval accuracy, they serve as label candidates for the subsequent feedback-driven pattern search.

Stage ❷ gathers user feedback. During this stage, domain knowledge flows into the pipeline to steer the model to converge to the user's notion of relevance. Compared to traditional feed-

back on the relevance of the found patterns alone, we allow the user to correct their positions and sizes and call this type of augmented user feedback "multi-feature feedback". This will be the topic of Chapter 8. But in a nutshell, with multi-feature feedback, the user can confidently accept a found pattern F that partially matches a target pattern T after correcting its position and size, so that $F_{new} := T$.

Stage ❸ and Stage ❹ form the proposed "nonmyopic search".

Stage ❸ uses the user labels obtained in Stage ❷ to train a model. Traditional search uses the model to classify patterns directly as relevant or not. The input of the model is a pattern P, and the direct output of the model is the probability that P is relevant $f_{rel}(P)$. In contrast, nonmyopic search trains a model to classify whether each time step in the time series is in a target pattern or not. The input of the model is the local proximity $S(i-\zeta, i+\zeta+1)$ around the time step $\boldsymbol{s}_i$ to classify. The direct output of the model is the probability that $\boldsymbol{s}_i$ is in a target pattern $\tilde{f}_{rel}(S(i-\zeta, i+\zeta+1))$, where we use the relevance function for each time step

$$\tilde{f}_{rel}\colon \mathbb{R}^{(2\zeta+1)\times d} \to [0,1]$$

instead of f_{rel} defined in Equation 2.3 to indicate that the model is used to estimate the relevance of time steps, rather than the relevance of patterns. This alteration brings many benefits, which we will elaborate in Section 7.3.2.

Because the user expects relevant patterns in whole pieces, rather than time steps in relevant patterns, Stage ❹ performs density-based clustering to merge (the indexes of) the time steps classified as "in-target-pattern" $\{i \mid \alpha \le i < \beta, S(\alpha,\beta) \in \mathbb{F}\}$ into intervals of found patterns $\{(\alpha,\beta) \mid S(\alpha,\beta) \in \mathbb{F}\}$, where $\mathbb{F}$ denotes all found patterns.

7.3.1 *Initial Search: Suggesting Label Candidates*

NOOPS's pipeline begins with a rule-based initial search (Stage ❶ in Figure 7.2). We need this stage because the user often can only provide one target pattern example at the beginning. It is insufficient for model training, and the rule-based search is the only viable option. However, if the user can provide multiple target pattern examples, the user can start with Stage ❸ directly. Conversely, if the rule-based initial search has already , it is unnecessary to proceed with the subsequent feedback-driven pattern search.

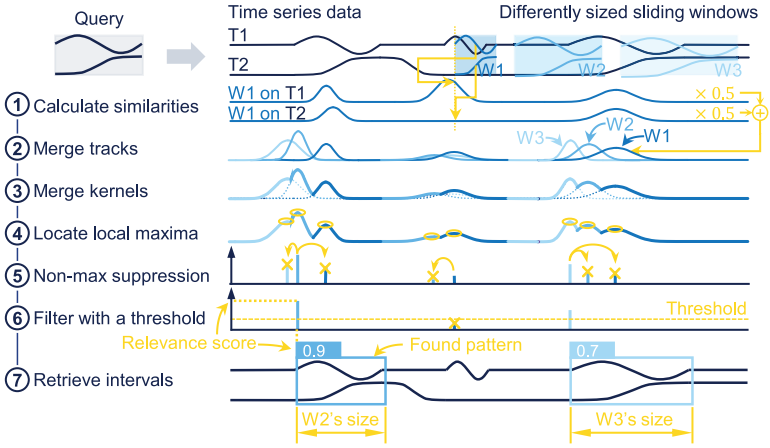


Figure 7.3: **Rule-Based Initial Search.** In a nutshell, it scans the time series data to find the most probable temporal positions of the target patterns; at each temporal position, it finds the most plausible size of the target pattern.

Our work aims to contribute to feedback-driven pattern search. Stage ❶ is rule-based and does not contain academic contributions. Nonetheless, we explain it in detail in this section for three reasons. First, we have used this technique, and it is not self-explanatory. Second, nowhere else have we found a complete and detailed description of how it works. Third, the rule-based initial search illustrated in Figure 7.3 is basically the same as the traditional model-based feedback-driven pattern search after model training, which classifies time series patterns directly and is the baseline for the proposed nonmyopic search.

Figure 7.3 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 ϕ, it undergoes seven steps and outputs found pattern intervals together with their relevance $\{((\alpha, \beta), f_{rel}(S(\alpha, \beta))) \mid S(\alpha, \beta) \in \mathbb{F}\}$.

Step ① calculates the similarity between Q and every pattern $[S(i, i+\phi)]_{0 \le i \le n-\phi}$ in S. We adopt the fastest similarity measure algorithm MASS [188] to carry out the task, which computes the similarity profile with ED in high speed, the primary concern for the initial search. Like many other methods, MASS does not support MTS directly. So, we run MASS for each track and merge the results over all tracks in Step ②. In practice, we use m resampled (temporally scaled) versions of Q of lengths $\boldsymbol{\phi} = [\phi_u]_{0 \le u < m}$ to find patterns of variable sizes. MASS calculates distance rather than similarity. We inverse the distances and normalize them to

range $[0, 1]$ to get raw relevance for each time step i in each track j with each resampled version u of the query

$$R_{raw} = \left[r_{iju} = \begin{cases} f_{sim}\left(\boldsymbol{s}_j\left(i,\, i+\phi_u\right), \boldsymbol{q}_{ju}\right) & \text{if } 0 \le i \le n-\phi_u \\ 0, & \text{if } n-\phi_u < i < n \end{cases} \right]_{\substack{0\le i<n \\ 0\le j<d \\ 0\le u<m}}$$

where $\boldsymbol{s}_j$ is the j-th track in S, and $\boldsymbol{q}_{ju}$ is the j-th track in u-th resampled version of Q. We pad R_{raw} with 0s, such that $\forall i \in (n-\phi_u, n)$, $r_{iju} = 0$. Padding ensures that all m resampled versions of Q produce a relevance profile of the same size n along the time axis indexed by i and simplifies subsequent calculations.

Step ② merges the raw relevance R_{raw} by averaging over tracks to get the interim relevance for each time step i with each resampled version u of the query

$$R_{interim} = \left[r_{iu} = \frac{1}{d} \sum_{j=0}^{d} r_{iju} \right]_{\substack{0\le i<n \\ 0\le u<m}}$$

Step ③ merges $R_{interim}$ by choosing the highest relevance over u for each time step i to get the merged relevance

$$\boldsymbol{r}_{merged} = \left[r_i = \max_{0\le u<m} \left(r_{iu}\right) \right]_{0\le i<n}$$

During Step ③, the lengths of the chosen u-th resampled version of Q for each time step index i are kept

$$\tilde{\boldsymbol{\phi}} = \left[\tilde{\phi}_i = \phi_{\substack{\text{argmax}(r_{iu}) \\ 0\le u<m}} \right]_{0\le i<n}$$

so that we can retrieve the found pattern lengths later in Step ⑦.

Step ④ retrieves all strict local maximum points in $\boldsymbol{r}_{merged}$ to get potential start positions of the found patterns

$$\boldsymbol{\alpha}_{raw} = \{ i \mid r_{i-1} < r_i,\, r_i > r_{i+1} \}$$

To avoid finding the same pattern as several separate overlapping ones, Step ⑤ conducts NMS,

$$\boldsymbol{\alpha}_{nms} = \left\{ i \in \boldsymbol{\alpha}_{raw} \mid r_i = \max_{i-\varphi\le t\le i+\varphi} r_t \right\}$$

where φ is the radius within which the highest local maximum suppresses the nearby lower local maxima.

Step ⑥ filters $\boldsymbol{\alpha}_{nms}$ with the relevance threshold η

$$\boldsymbol{\alpha}_{filtered} = \{i \in \boldsymbol{\alpha}_{nms} \mid r_i \geq \eta\}$$

The user usually cannot set an optimal η a priori but needs to adjust it after inspecting the results and repeat Step ⑥ and Step ⑦.

Finally, Step ⑦ retrieves the intervals of the found patterns as well as their relevance

$$\left\{\left((i,\, i+\tilde{\phi}_i)\,, r_i\right) \mid i \in \boldsymbol{\alpha}_{filtered}\right\}$$

Because this chapter is dedicated to nonmyopic search (the combination of Stage ❸ and Stage ❹), we will explain them in subsequent subsections and defer multi-feature feedback (Stage ❷) to Chapter 8.

7.3.2 *Time-Step-Based Classification: The Adaptive Core*

In the proposed data processing pipeline in Figure 7.2, the time-step-based classification (Stage ❸) serves as the adaptive core.

Traditional model-based pattern search classifies time series patterns directly. To find patterns of different sizes, the model has to go through multiple resampled versions of the time series S. Otherwise, the input size of the model would dictate the length of the found patterns. The input of the model is straightforwardly the pattern to classify.

We propose to use the model to estimate the probability that a time step s_i in S is in a target pattern or not. Let us call the "in-target-pattern" time steps positive time steps and the "not-in-target-pattern" time steps negative time steps. The relevance of all time steps are

$$\tilde{\boldsymbol{r}} = \left[\tilde{r}_i = \begin{cases} \tilde{f}_{rel}\left(S\left(i-\zeta_\alpha,\, i+\zeta_\beta+1\right)\right), & \text{if } \zeta_\alpha \leq i < n-\zeta_\beta \\ 0, & \text{otherwise} \end{cases}\right]_{0\leq i<n}$$

where we pad the relevance with 0 (especially for $0 \leq i < \zeta_\alpha$) so that the indexes of time steps do not change. The indexes of the positive time step are

$$\boldsymbol{\tau} = \left\{i \mid \tilde{f}_{rel}\left(S\left(i-\zeta_\alpha,\, i+\zeta_\beta+1\right)\right) \geq \eta\right\}$$

Recall that $\tilde{f}_{rel}$ introduced at the beginning of Section 7.3 estimates the probability that a time step is located in a target pattern or not, based on its proximity. ζ_α and ζ_β are hyperparameters that we call "lookbehind" and "lookahead", respectively, because they resemble the positive lookbehind and positive lookahead in regular expression. In practice, we set $\zeta_\alpha = \zeta_\beta = \zeta$

and call ζ "lookaround". It is set to the average length of the positively labeled found patterns $\mathbb{C}_+$ from Stage ❷, namely, $\zeta = \frac{1}{|\mathbb{C}_+|} \sum_{C_+ \in \mathbb{C}_+} |C_+|$.

For each model, the concrete implementation of $\tilde{f}_{rel}$ differs slightly. For instance, some models, like `CNN` and `LSTM`, support multiple channels/tracks out of the box and can take $S(i - \zeta_\alpha, i + \zeta_\beta + 1)$ as their input directly. Other models, like `random forest` and `XGBoost`, require flattening $S(i - \zeta_\alpha, i + \zeta_\beta + 1)$ before inputting it into the model. How to properly use each model to calculate the relevance of a pattern is out of the scope of our work, which proposes a model-agnostic data processing pipeline. The configuration of the models can be found in Section 7.5.1.

The proposed change from pattern-interval-based classification to time-step-based classification brings three benefits.

Firstly, it can retrieve patterns of variable sizes in one scan. The traditional search uses a sliding window of size ϕ to retrieve patterns of the same size ϕ. If target patterns in the data vary in size considerably, they need a series of sliding windows of different sizes. The time-step-based classification breaks the connection between the model input size and the sizes of the found patterns. It still uses a sliding window of size $\zeta_\alpha + \zeta_\beta + 1$ to scan the data. However, this size does not dictate the sizes of the found patterns. And the sliding window scans the time series only once.

Secondly, it is aware of the pattern context. As illustrated in Figure 7.1 (please note the arrow for "context"), when the sliding window is at the start time step of a target pattern, the lookbehind juts out to the historical time steps on the left side of the pattern. Likewise, the lookahead will gradually capture the future time steps after the end time step of the pattern, as the sliding window approaches the end of the pattern. Therefore, when retrieving a found pattern $F = S(\alpha, \beta)$, the search takes the subsequence $S(\alpha - \zeta_\alpha, \beta + \zeta_\beta)$ into consideration, including the context $S(\alpha - \zeta_\alpha, \alpha)$ and $S(\beta, \beta + \zeta_\beta)$. This context-awareness can be helpful when there is characteristic information preceding or succeeding the target patterns. This is for example the case where the target patterns represent a phase in a process, e.g. in our `Deep Valve Linear Phase` use case illustrated in Figure 7.10. Whereas, if the context is not informative, the model learns to ignore it during training anyway, as we will see in the experiments with our other use cases.

Thirdly, model training becomes more robust. In traditional search, one labeled pattern $S(\alpha, \beta)$ is also used as a single train-

ing instance. $S(\alpha, \beta)$ is the input of the model and the label (a relevant pattern or not) is the output of the model. In contrast, the time-step-based classification uses a labeled pattern $S(\alpha, \beta)$ as $\beta - \alpha$ positive training instances $\left[S\left(i - \zeta_\alpha,\, i + \zeta_\beta + 1\right)\right]_{\alpha \le i < \beta}$. These training instances depict the pattern from different temporal perspectives. It greatly increases the robustness of the training, reflected in the much smaller standard deviation of the retrieval accuracy over multiple runs in our experiments in Section 7.5.2. Moreover, it enables us to perform certain data augmentation to further increase the number of labeled training instances. If $S(\alpha, \beta)$ is a positively labeled pattern and its start time step index α and its end time step index β are precise, its context should have negative time steps. Consequently, we can use $\left[S\left(i - \zeta_\alpha,\, i + \zeta_\beta + 1\right)\right]_{\alpha - \tilde{\zeta} \le i < \alpha \text{ or } \beta \le i < \beta + \tilde{\zeta}}$ as negative training instances during training, where $\tilde{\zeta}$ is a hyperparameter which we set as ζ. Not only does this data augmentation further increase training instances, but it also makes it optional for the user to provide negative labels. Whereas, the traditional feedback-driven pattern search forces the user to provide negatively labeled patterns because the model cannot be trained only with positive training instances. On the debit side, training costs more time due to the increase of training instances. However, training time is less of a concern compared to inference time, as we will see in Section 7.5.2, because training time depends on the number of labels, which is typically small; while the inference time scales linearly with the length of the time series, which is generally large.

Regarding the first two benefits, because the proposed search method sees patterns of different sizes in one scan and peeks into the context beyond the patterns during retrieval, we name it "nonmyopic search".

7.3.3 *Density-Based Clustering: From Steps to Intervals*

Identifying all time steps in relevant patterns $\boldsymbol{\tau}$ from Stage ❸ is one step closer to the desired output. However, the user expects found patterns in whole pieces, not the time steps in them. Accordingly, Stage ❹ converts the positive time steps $\boldsymbol{\tau}$ into found pattern intervals $\{(\alpha, \beta) \mid S(\alpha, \beta) \in \mathbb{F}\}$, where $\mathbb{F}$ denotes all found patterns.

We cannot blindly merge consecutive positive time steps, since misclassification is inevitable. For instance, in Figure 7.4, it is likely that Stage ❸ misclassified Time Step 2 and 3 as positive,

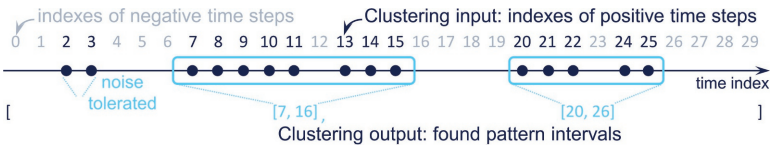


Figure 7.4: **A Canonical Example of Clustering Time Steps.** We merge the positive time steps to found pattern intervals because the user expects found patterns in whole pieces rather than individual time steps in target patterns. Because the positive time steps from the same pattern concentrate along the time axis, we can achieve the merging using density-based clustering.

and misclassified Time Steps 12 and 23 as negative. False positive time steps would result in incorrect found patterns, and false negative time steps would split a target pattern into several smaller found patterns. The merging process must tolerate such errors.

We formulate this problem as a one-dimensional density-based clustering problem. Density-based clustering [24], especially Density-Based Spatial Clustering of Applications with Noise (DBSCAN) introduced by [88], the algorithm that we use, is well established. Its data assumptions and properties suit our case very well. On the one hand, density-based clustering assumes that the data items inside a cluster are densely located, and the clusters are separated by "vacuums" only sparsely dotted with data items. In our case, the number of clusters corresponds to the number of found patterns. Along the time axis, a target pattern reveals itself as a group of densely located "in-target-pattern" time steps. While between target patterns, the density of "in-target-pattern" time steps (resulting from misclassification in Stage ❸) is much lower in comparison. In our case, the number of target patterns is usually unknown before the search. On the other hand, density-based clustering supports outlier (occasional false positives in low-density regions) removal off the shelf. In our case, we would like to ignore time steps misclassified as "in-target-pattern".

The input of DBSCAN contains the indexes of the positive time steps $\boldsymbol{\tau}$, and the output gives the found pattern intervals $\{(\alpha, \beta) \mid S(\alpha, \beta) \in \{F\}\}$, where $\{F\}$ denotes all found patterns. Finally, the relevance of a found pattern $F = S(\alpha, \beta)$ is the average relevance of all time steps in F, i.e., $f_{rel}(F) = \frac{1}{\beta - \alpha} \sum_{i=\alpha}^{\beta-1} \tilde{r}_i$, where $\tilde{r}_i \in \tilde{\boldsymbol{r}}$ is the relevance of i-th time step, explained in Section 7.3.2.

Unlike partition-based clustering [18] (typically k-means introduced by [169]) and hierarchical clustering [213], density-based

clustering does not require a priori knowledge of the number of clusters or the number of levels. However, it is admittedly not an advantage of density-based clusteirng in our case due to other parameter settings indirectly related to the number of clusters / found patterns. Specifically, DBSCAN has two standard parameters, *eps* and *min_samples*. We set them as follows. The `scikit-learn` documentation explains *eps* as "the maximum distance between two samples for one to be considered as in the neighborhood of the other." 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 pattern search. Empirically, we set it $\frac{2}{3} \cdot \zeta$. For *min_samples*, the `scikit-learn` documentation explains that 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." Conceptually, the higher the value, the denser the positive time steps 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 $min\_propotion = 0.1$. Other parameters use defaults from `sklearn.cluster.DBSCAN`.

7.4 USER INTERFACE

We extended the user interface in Section 6.4 to assist users in application of the developed data processing pipeline highlighting nonmyopic search. Because Section 6.4 has already explained some details like data input and query specification, we explain only the extensions and the unique visualizations as well as interactions designed for nonmyopic search.

7.4.1 *Result Inspection*

To allow the user both to explore the data freely and to inspect the found patterns readily, we opt for two views to display the results. First, an *Overview* shown in Figure 7.5 provides a broad context of the events for (further) analysis. We plot the data in line charts and events as boxes, including blue found patterns, green positive labels, red negative labels, and pink highlighted patterns. The colors are consistently used throughout our user interface. Second, a *Found Patterns View* shown in Figure 7.6 lists the extracted patterns in descending order of their relevance so that

the user does not need to zoom in on them individually in the *Overview* and can compare them with each other easily. Double-clicking a found pattern in either view will highlight the pattern, i.e., by filling the pattern pink in the *Overview*, adding a pink frame to the pattern in the *Found Patterns View*, and highlighting it in other views that will be described soon.

Figure 7.5: **Overview.** In this view, the user can explore the data file freely and examine the context of the events.

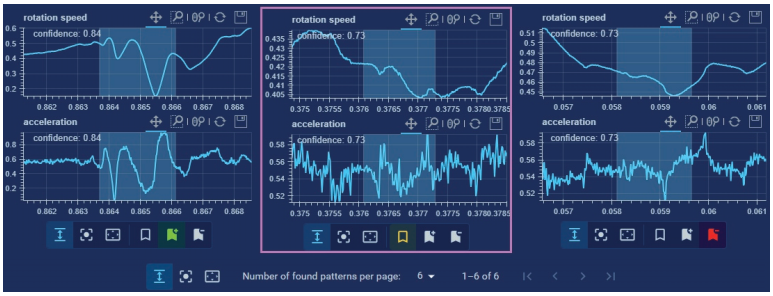


Figure 7.6: **Found Patterns View.** In this view, the user can easily inspect and compare the extracted found patterns in their own charts.

Below the original data tracks, we depict the relevance scores of all pattern candidates. As explained in Section 7.3.1, the rule-based initial search uses a relevance threshold η to filter out found patterns from pattern candidates. Correspondingly, the user can drag a horizontal line representing η to adjust it. This is useful when the user intends to include or exclude certain pattern candidates.

The *Found Patterns View* displays the found patterns in their own value range by default to expose their details to the maxi-

mal extent. The user can change the y-axis range by scrolling the mouse wheel, or clicking a button in the button group under the chart to zoom to the range of the pattern itself (default), of the query, or of the whole time series data. The same button group is also available globally so that the user can change the displayed value ranges of all found patterns with one click, e.g., for a better comparison among found patterns with the same value range.

7.4.2 *Result Analysis*

To facilitate result analysis, e.g., discovering false positives / negatives and diagnosing the problematic stage, we propose multiple visualizations in the *Analysis View* shown in Figure 7.7.

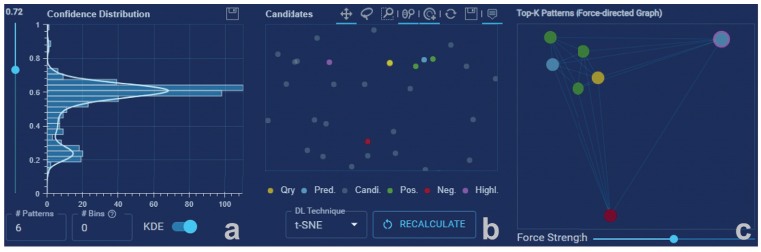


Figure 7.7: **Analysis View.** This view helps with relevance threshold tuning, suggests false negatives / positives for examination, and hints at the erroneous stage.

The histogram in Figure 7.7.a shows the relevance distribution of all candidates. It aids the user in tuning the relevance threshold η, a nontrivial subproblem in pattern search. The user can move the slider to the left of the histogram to adjust the threshold. It is equivalent to tuning the threshold in the last track in the *Overview*, as explained in Section 7.4.1. Empirically, we find that a small cluster of top relevance scores after the initial search indicates a good result, and η fits right below the cluster. Alternatively, the user can adjust the number of found patterns in the input field below the histogram, and η will fit accordingly. This feature is convenient when the user knows roughly the number of occurrences of the target event or wants to remove the found patterns with the lowest relevance.

The dimensionality reduction scatter plot in Figure 7.7.b visualizes the (relative) similarities of the candidates, including the found patterns, the positive / negative labels, and the highlighted pattern. The user can switch between UMAP, t-SNE, and

PCA to exclude artifacts caused by specific dimensionality reduction techniques. This visualization intends to help find false negatives (missing patterns). The false negatives manifest themselves as gray points (candidates) that are close to the yellow query or the green positive labels. We would have calculated the numerical distances and made suggestions without this visualization. However, there are potentially interesting phenomena that such a visualization can reveal. For instance, the user may find false negatives next to a found pattern far away from the query, suggesting that the query is not representative, or the false negatives form a distinctive strain. The user can double-click a candidate to highlight it in all other views for inspection. If there are many such potential false negatives, indicating a potentially too high η, the user can use the Bokeh lasso select tool. η can be lowered accordingly to include all of them as found patterns.

Similar to the scatter plot for false negatives, the force-directed graph in Figure 7.7.c aims to reveal false positives (irrelevant found patterns). We use the library `react-force-graph`. Besides the default forces, we have added a new force related to the pairwise pattern/node similarities, i.e., `d3.forceLink().distance((link) => 10 ** forceStrength * (1 - link.similarity))`. In this line of code, `link` represents an edge between two nodes/patterns A and B. `link.similarity` is the similarity between the two patterns, i.e., $f_{sim}(A, B)$. *forceStrength* is given by the slider beneath the force-graph so that the user can regulate the strength and thus the distances between the pattern nodes. Ideally, false positives are distant from the cluster containing true positives. The user can highlight potential false positives and inspect them in other coordinated views. Although the scatter plot provides similar functions, we find empirically that the force graph complements it. When there are few candidates, the former's performance deteriorates, while the latter works well. Moreover, the user can drag nodes in the force graph to sense the "robustness" of the result. Because target events usually occur sparsely in the data, we do not observe scalability issues with the force layout in all our real-world use cases.

The feedback-driven pattern search outputs found patterns directly. There are no candidates and no need to filter candidates with η. Therefore, we have removed the dimensionality reduction scatter plot due to the small number of found patterns and introduced new visualizations to diagnose the problematic stage. Figure 7.8.a shows a found pattern in the *Overview* (cut short drastically). We can see not only the elements explained in Sec-

tion 7.4.1, but also the relevance of the time steps as a dark yellow area chart in the lowest / third track. Based on the relevance of the time steps, clustering successfully detects a single found pattern, tolerating incorrectly classified time steps in the narrow relevance valley between the two plateaus. Had the clustering found two separate patterns corresponding to the two plateaus or only the larger one on the right, the user could adjust the clustering parameters accordingly. By contrast, if the user encounters the problem in Figure 7.8.b, where only a concentrated small portion of the time steps in the target pattern are correctly classified, the problem lies in the classification stage, not in clustering. In this case, the user may label the found pattern as a positive label, adjust its boundaries (we did this exactly in Figure 7.8 to help the reader recognize the target pattern, and the user interface showed a green box around the label), and update the model with a new feedback round, as will be explained in Section 8.4. Nonetheless, user feedback is the topic of Chapter 8.

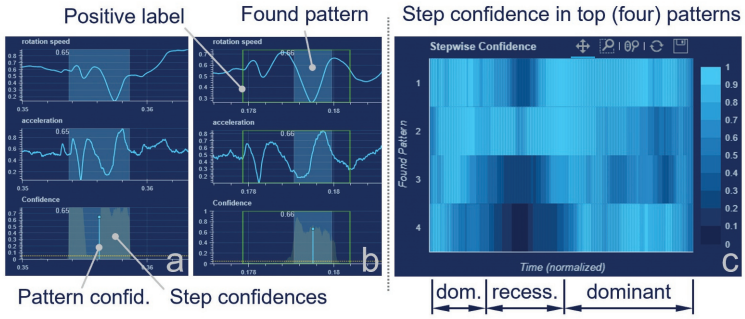


Figure 7.8: **Time-Step-Based Analysis:** The time-step-based pipeline opens new possibilities for visual analysis. It helps to find the problematic stage and dominant temporal regions.

Finally, a heatmap shown in Figure 7.8.c depicts the relevance of each time step in the found patterns. Here, each row represents one found pattern, and each column is one time step. The patterns are resampled to have the same number of time steps so that the relative positions of the regions inside the patterns are aligned. The found patterns in Figure 7.8.c show two dominant regions with high time step relevance. They are more significant features of the event because they contribute more to the detection of the patterns than the recessive region between them, where the patterns usually show more variance and inconsistency. One limitation of this visualization is that the boundaries of the found patterns should be relatively precise or at least

similarly imprecise; otherwise, the regions in different patterns may not align well.

7.5 EVALUATION

We evaluated the proposed nonmyopic search against the baseline, traditional (myopic) search, with a quantitative benchmark on retrieval accuracy and speed. We defer the evaluation of the VQS to Section 8.5.3 and Section 8.5.4 because the multi-feature feedback to be presented in Chapter 8 extends nonmyopic search in this chapter, and the two chapters share the same user interface. Nonetheless, we evaluated the distinctive algorithmic proposals separately.

7.5.1 *Experiment Setup*

We used the same three labeled datasets, namely, `Deep Valve`, `Variable Displacement`, and `EEG Eye State`, as in Section 5.5.1 and Section 6.5.1. Because this experiment was much more complex and time-consuming than those in previous chapters, we used fewer but the most challenging datasets that we have. The `Deep Valve` dataset has two use cases, each with unique target patterns. In previous chapters (Section 5.5 and Section 6.5), we used only the simpler use case because neither our proposed method (`PSEUDo` and `SAXRegEx`) nor the state-of-the-art benchmark methods performed reasonably in the harder use case requiring context-awareness. In this chapter, we will use both use cases of this dataset to showcase `NOOPS`' advantageous context-awareness.

We included four models in this experiment: 1) `Random forest`: the winner of retrieval accuracy in traditional search according to [156]; 2) `XGBoost`: the winner both in terms of retrieval accuracy and speed in our quick-and-dirty model selection; 3) `LSTM`: one of the most popular models for time series analysis; 4) `1D-CNN`: a common alternative to RNNs for time series analysis [55]. Nonetheless, we did not intend to find the best model. Rather, we included multiple models to avoid judgments biased by the model choice.

The algorithm involves many (hyper-)parameters, e.g., lookaround of time-step-based classification, the hyperparameters of the classification models, and the parameters of DBSCAN. It is not our intention to analyze the influence of all parameters and optimize them (WG3). Instead, we would like to verify the

advantages of nonmyopic search over traditional search. We set the (hyper-)parameters with reasonable values (mostly defaults provided by the library) and leave them unchanged throughout the entire evaluation.

We conducted all experiments with the same hardware and system setting described in Section 5.5.1.

For rule-based initial search, we set the number of sliding windows $m = 8$, as mentioned in Section 2.6. The lengths of these sliding windows $\boldsymbol{\phi} = [\phi_u]_{0 \le u < m}$ scaled logarithmically between a lower bound ϕ_0 and an upper bound ϕ_{m-1}. The sliding window sizes were

$$\boldsymbol{\phi} = \left[\phi_i = \left\lfloor \phi \times \left(\frac{\lambda_{upper}}{\lambda_{lower}} \right)^{\frac{u}{m-1}} \right\rfloor \right]_{0 \le u < m}$$

where the operator $\lfloor . \rfloor$ denotes the floor function, λ_{lower} and λ_{upper} are the scaling factors for ϕ_0 and ϕ_{m-1}, respectively. For instance, $\lambda_{lower} = 0.5$ and $\lambda_{upper} = 2.0$ mean that the lengths of the found patterns should be at least half and at most double of the query length ϕ.

The lower and upper scaling factors λ_{lower} and λ_{upper} are determined by how much the durations of the target patterns in the time series scales. Specifically for each use case, we set

- `Deep Valve Whole Signal`: $\lambda_{lower} = 0.5$, $\lambda_{upper} = 3.0$;
- `Deep Valve Linear Phase`: $\lambda_{lower} = 0.5$, $\lambda_{upper} = 3.0$;
- `Variable Displacement`: $\lambda_{lower} = 0.5$, $\lambda_{upper} = 2.0$; and
- `EEG Eye State`: $\lambda_{lower} = 0.5$, $\lambda_{upper} = 12.0$ (the durations of the target patterns vary substantially in this use case).

We set the NMS radius φ to the same as the query length ϕ. We configured the models as follows.

- `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.selu`, 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 was set to 10 epochs, except for random forest, which does not support early stopping.

Model training requires labels. For positive labels, we used the first (sorted in ascending order of start time) several ground truths as positive labels for both nonmyopic search and traditional search. For negative labels, due to their different mechanisms of label usage, we provided different negative labels for nonmyopic search and traditional search. For nonmyopic search, we simply provide two or three relatively long negative labels. This is the desired way of providing negative labels for nonmyopic search, basically to show what the background looks like. Otherwise, negative labels are optional anyway due to the data augmentation specific to nonmyopic search explained in Section 7.3.2. For traditional search, negative labels are mandatory. Moreover, meaningful negative labels should have similar sizes as the positive labels / target patterns. Therefore, we provided at least five and up to twenty negative labels chosen randomly, depending on the total number of ground truths.

7.5.2 *Performance Comparison Between Nonmyopic Search and Traditional Search*

Figure 7.9 shows the experiment results, one diagram for a use case. We omitted `LSTM` because it was much slower than the other models, causing a skewed visual representation of results from other faster models. The abscissae represent wall-clock time, and the ordinates retrieval accuracy measured by AP30. Accordingly, the overall performance increases from bottom right to top left in a diagram. We chose AP30 because it spread the metric scores in the diagrams relatively well between 0 and 1. The light blue points represent the models in nonmyopic search, while the dark blue (nearly black) linked points the same models in traditional search. Similar to the rule-based initial search in Section 7.3.1, the performance of traditional (model-based) search also depends on a number of sliding windows to retrieve patterns of different

sizes. Hence, we measured the performance of a model in traditional search with different numbers of sliding windows and used linked points or lines to represent them. Each line represents one model. We repeated every accuracy measurement five times, averaged the scores, and drew vertical error indicator bars in the diagrams to show the standard deviations over the five runs. Each speed measurement was averaged over ten repetitions. We did not conduct accuracy and speed measurements simultaneously because the former contained more tasks like generating plots and saving results, which were not interesting to the latter.

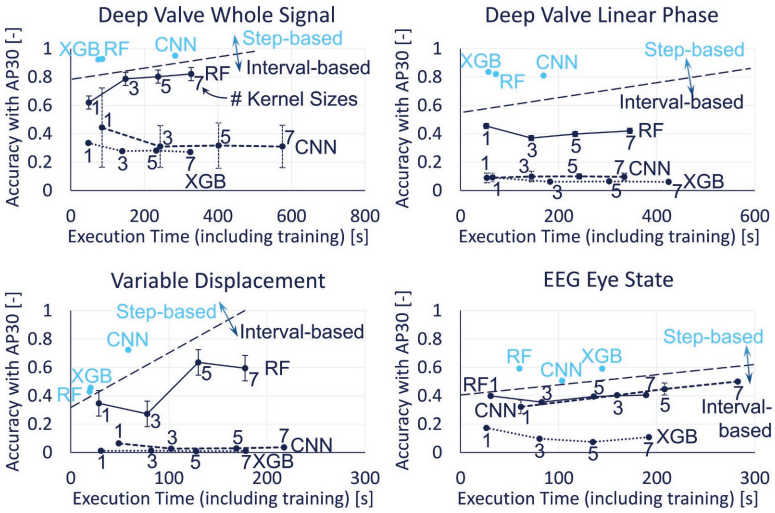


Figure 7.9: **Benchmark on Retrieval Accuracy and Speed.** Compared with the linked dark blue points denoting traditional search with different sliding windows, the light blue points denoting nonmyopic search are located more to the top left in each plot, indicating better retrieval accuracy and speed overall. (RF: `Random Forest`, XGB: `XGBoost`)

With the same model, the light blue points denoting nonmyopic search are generally higher than the dark blue lines denoting traditional search, indicating a consistently higher retrieval accuracy with nonmyopic search. In fact, many dark blue lines are fairly low, suggesting that traditional search does not work with certain models in certain use cases.

Furthermore, nonmyopic search is more robust (in terms of retrieval accuracy) because 1) we can hardly discern the error indicator bars around the light blue points, while the error indicator bars are prominent around some dark blue points, implying that traditional search may produce more inconsistent results in dif-

ferent runs; 2) the light blue points of different models in the same diagram are relatively of the same height while the dark blue lines of different models in the same diagram are more distributed vertically, hinting at the robustness of nonmyopic search with regard to the model choice. We attribute the huge improvement in robustness primarily to the increase in the number of training samples elaborated in the third benefit of nonmyopic search in Section 7.3.2.

In the `Deep Valve Linear Phase` use case, all models in traditional search performed poorly. Figure 7.10 illustrates a typical pattern in the dataset. The `Deep Valve` dataset records the electric current through a solenoid valve used in a cooling system test bench. The illustrated pattern captures a complete valve operating cycle, which consists of a maximum of five phases (sometimes, an operating cycle may skip some phases). 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 surge. 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, which is under the hood the response of an underdamped second-order linear system excited by a step function signal in control engineering. In the third phase, the linear phase, the movable part moves from one end to another. The valve changes from completely closed to completely open. In the fourth phase, after reaching the stop, the current may deviate a bit from the linear phase. In the fifth phase, the current drops 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 (heterogeneous scaling or warping), even disappear. The patterns in the `Deep Valve Whole Signal` use case include a complete operating cycle. Whereas, the `Deep Valve Linear Phase` use case requires retrieving the linear phase titled "Move". The internal characteristics of this pattern (simply a linear phase) are not distinctive enough, resulting in poor performance of all traditional search methods. However, together with its preceding and succeeding phases, the pattern becomes distinctive and easier to detect, which explains the high performance of nonmyopic search that is aware of the pattern context.

Not only is nonmyopic search more accurate and robust, but it is also faster because it needs to scan the data only once, while traditional search multiple times to find patterns of different sizes.

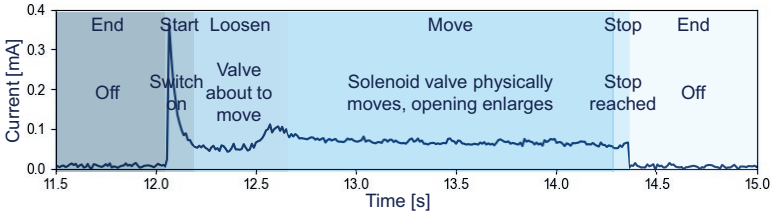


Figure 7.10: **An Operation Cycle of The `Deep Valve Linear Phase` use case.** It undergoes five phases. The target pattern is the linear phase titled "Move" in the plot. The linear phase per se is not distinctive enough during retrieval. However, the context (phases before and after the target phase) has rich features and helps identify the target pattern.

As shown in Figure 7.9, the light blue points are generally more left than the dark blue lines.

7.5.3 *Further Insights From Analysis of Training and Inference Time*

Figure 7.11 shows the training and inference time with `random forest` in nonmyopic search and traditional search with various numbers of sliding windows, which reveals more insights. Due to space limits, we show only the result with `random forest`. Nonetheless, it exposes the typical qualitative relationships between different parts of execution time that experiments with other models also exhibit. 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.

In Figure 7.11, the inference time of nonmyopic search (39.92 ± 0.73 seconds in `Deep Valve Whole Signal` use case) is comparable with traditional search with one sliding window (48.74 ± 1.47 seconds in the same use case), which can only find patterns of a fixed size. They both scan the data only once. Compared with traditional search with five sliding windows (237.15 ± 4.31 seconds in the same use case), which is coarse enough (we use eight sliding windows in production in rule-based initial search), nonmyopic search saves much time. On the other hand, nonmyopic search requires considerable training time (45.73 ± 0.91 in the same use case) seconds, due to the massive increase of training instances, while training time is negligible for traditional search (only 0.16 ± 0.01 seconds in the same use case). However, increased training time in nonmyopic search is not a serious prob-

lem for larger data. Because training time is mainly determined by the number of labels, which is usually small, while inference time scales linearly with the size of the time series data, which is generally large. Therefore, for larger datasets, the inference time will dominate and should be the major concern in terms of execution time. For traditional search, the low training time implies insufficient training, leading to its low retrieval accuracy.

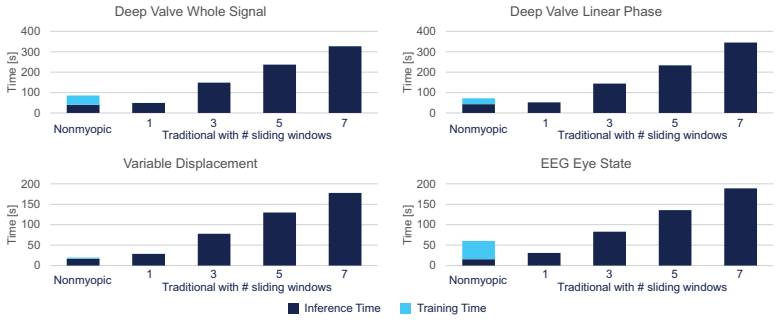


Figure 7.11: **Training and Inference Time with `Random Forest`.** This series of diagrams show the relationships between training and inference time in nonmyopic and traditional search. For one thing, the inference time of nonmyopic search is comparable with traditional search with one sliding window, both scanning the data only once; whereas, the execution time in traditional search scales roughly proportional to the number of sliding windows. For another, while the training time in traditional search is negligible compared to inference time, the training time in nonmyopic search is considerable due to the massive increase of training instances.

From a simplified perspective, if restricting the experiment results to `random forest` (the best model in terms of retrieval accuracy and speed for traditional search) and the number of sliding windows to five, nonmyopic search brought 0.19 retrieval accuracy boost (absolute difference) in 68% less execution time, as shown in Table 10 derived from the first sheet in "evaluation.xlsx" under `https://osf.io/36kf2`.

7.5.4 *Secondary Findings*

In addition to the main findings, we present secondary observations. Firstly, no model consistently outperforms the other. Nonetheless, `random forest` proves both highly accurate and fast (GPU acceleration deactivated for all models), both in nonmyopic search and in traditional search, showing its general suit-

Use Case	Retrieval Acc. with AP30			Execution Time in Sec.		
	Nonm.	Trad.	Acc. Gain	Nonm.	Trad.	Reduc.
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 10: **(Oversimplified) Performance Gain with Nonmyopic Search.** If restricting to `random forest` (the most accurate and fastest model in traditional search) and to five sliding windows (which should be coarse enough), nonmyopic search increased retrieval accuracy by 0.19 (absolute difference) 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`)

ability for time series pattern search. `XGBoost`, particularly with GPUs enabled, is likely to offer the fastest performance. Conversely, neural networks like `CNN` have significantly lower memory consumption since they support training in batches. Secondly, adding the number of sliding windows does not necessarily improve retrieval accuracy, but execution time increases proportionally. It is very likely that some models in some use cases in traditional search fail to learn sufficient useful information during training anyway, regardless of the resolution (number of sliding windows) during inference.

In summary, nonmyopic search improves retrieval accuracy and speed for feedback-driven pattern search.

Nonmyopic search can best unleash its power when used together with multi-feature feedback to be present in Chapter 8. Thus, we designed a single VQS incorporating both techniques. However, we have to defer the evaluation of the user interface including a case study and an expert study to Section 8.5.3 and Section 8.5.4, respectively, after we present the UI components for multi-feature feedback.

7.6 DISCUSSION

`NOOPS` helps automotive calibration engineers trace events of interest in sensor measurements, especially when the fast rule-based

pattern search techniques presented in previous chapters fail. While we focus on the general data processing pipeline highlighting nonmyopic search, further research, especially into the components of the pipeline, will help consolidate the methodology and optimize its effectiveness.

Nonmyopic search does not work for overlapping patterns. If several target patterns overlap, it will find a long pattern and cannot separate them, similar to semantic segmentation in computer vision. In practice, the target patterns are manifestations of a target event. It is often meaningless for our target event, e.g., the periods when components in a valve physically move as in our `Deep Valve Linear Phase` use case, or cylinder deactivation in a gasoline engine as in our `Variable Displacement` use case, to have overlapping occurrences.

There are several (hyper-)parameters involved in the proposed data processing pipeline, e.g., hyperparameters of the machine learning models and parameters in DBSCAN. We set them once with reasonable values, but defer a comprehensive analysis of their influence on the search performance to future work. Notwithstanding (hyper-)parameter optimization, it is evident through our experiments that the proposed nonmyopic search improves the performance of the traditional search.

The rule-based initial search is based on a similarity measure. The provided tracks should reveal the event of interest as visually distinguishable patterns. In contrast, a model-based search can capture subtler features. Therefore, it makes sense to allow the user to add tracks after the rule-based initial search for the subsequent model-based feedback-driven pattern search. Because some tracks, informed by domain knowledge, may be relevant to the event of interest. However, the event of interest may not manifest itself as recognizable visual patterns in those tracks. Still, they may help the model identify task-specific or system-interval dynamics.

7.7 CONCLUSION

In this chapter, we have proposed a data processing pipeline and a tool called `NOOPS` for model-based pattern search in MTS. It features nonmyopic search that combines time-step-based classification and density-based clustering, enabling finding patterns of variable sizes, taking context of the patterns into consideration during retrieval, and makes model training more robust. Interestingly and disappointingly, we found that traditional (model-based pattern) search that features sliding windows scanning the

data with a machine learning model actually does not work very well and fails to satisfy our industrial needs. With nonmyopic search, we can raise retrieval accuracy and speed significantly and simultaneously. Our method is in production and helps automotive calibration engineers trace events of interest. In prospect, we can expect nonmyopic search to be the standard approach in model-based pattern search. In the future, we plan to study the effects of various (hyper-)parameters in the individual stages in the nonmyopic search data processing pipeline on retrieval accuracy and speed.

8

MULTI-FEATURE FEEDBACK FOR EFFECTIVE FEEDBACK-DRIVEN PATTERN SEARCH

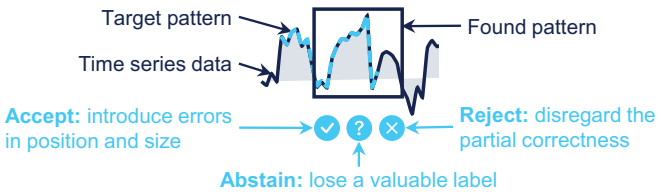


Figure 8.1: **Pitfall of Traditional Relevance Feedback.** Traditional relevance feedback is confined to acceptance, rejection, and abstention, which may place the user in a dilemma, where a found pattern is partially correct, and all options are controversial. We propose additional feedback on the positions and sizes of the found patterns for feedback-driven pattern search.

In this chapter, we present multi-feature feedback for feedback-driven pattern search. Besides feedback on relevance alone, it advocates feedback also on the positions and sizes of the found patterns. The additional feedback significantly raises the converged accuracy of the retrieved patterns. [1]

8.1 INTRODUCTION

Chapter 7 has looked into model-based pattern search while assuming that sufficient labels required during model training are already available. In reality, they are seldom at the user's disposal. Feedback-driven pattern search with relevance feedback / active learning aims to solve this problem. However, when we thought deeply and worked practically on this issue, the following questions struck us.

1. Is relevance the only form of feedback that the user can provide in the feedback-driven pattern search?
2. Is relevance alone sufficient for the retrieval results to improve to the user's satisfaction?

1 This chapter is mainly based on our ongoing publication PB6 with textual modifications for a coherent information flow.

This chapter tries to answer these questions.

This problem is interesting and important because, in our industrial practice, we find that existing research may have an overoptimistic view of performance improvement brought by relevance feedback. On the other hand, we believe that existing solutions have not exploited the specialties of the use cases. Existing methods confine user feedback on the found patterns to choices between acceptance, rejection, and abstention, which is ineffective if a found pattern is relevant but imprecisely bounded. Namely, when a found pattern partially overlaps a target pattern, for example, in Figure 8.1, none of the options is satisfactory. The idea of relevance feedback originates from information retrieval, especially text retrieval. However, time series have more structure to utilize than text. The relevance feedback originally designed for text retrieval may benefit from such structures.

We propose "multi-feature feedback", a technique that is based on nonmyopic search proposed in Chapter 7 and takes it to the next level. To overcome the limitation of binary feedback on the relevance of found patterns alone, we propose to solicit feedback not only on the relevance of the found patterns but also on more features, specifically, the positions and sizes of the found patterns. Then, we examined the effectiveness of this augmented feedback. In our experiments, multi-feature feedback raised retrieval accuracy by 0.35 according to the least strict metric (even more with stricter ones), averaged over all use cases in our experiments. Because multi-feature feedback works seamlessly with nonmyopic search in Chapter 7, we extended the user interface for `NOOPS`, as a necessary auxiliary work. We note that the focus here was not to drive the research in the direction of visual contributions, but rather to explore how to make effective use of active learning in multivariate time-series exploration scenarios. A case study and an expert study validated the usefulness of the VQS and revealed problems for future work.

8.2 RELATED WORK

This chapter extends traditional active learning or relevance feedback with feedback on multiple features of the retrieved result in addition to pattern relevance. Like query expansion reviewed in Section 6.2, relevance feedback is yet another essential technique in information retrieval. Please refer to Section 2.8 for an introduction to active learning / relevance feedback and its application in time series pattern search. Next, we will review works

that transgress conventional binary relevance feedback in information retrieval.

We have not discovered works on similar notions of multi-feature feedback yet. Instead, after reviewing the advancements of relevance feedback especially in Content-Based Image Retrieval (CBIR), we found some tangentially related works. Wang and Zhu concluded that considering multiple features like colors, textures, and shapes significantly improves retrieval accuracy compared with relying solely on one feature in CBIR [256]. Not only is the consideration of multiple features beneficial but also proper treatment of features on different levels. Lu et al. proposed creating a semantic network and taking into account the semantic contexts besides low-level features for CBIR [173]. Albeit multiple and multi-level features as the algorithm input, user feedback was still a choice between yes, no, and unsure for a sampled outcome. Ishikawa et al. broke this convention by requiring a score in $[0, 1]$ to drive their relevance feedback algorithm [131]; while Rui and Huang actually designed a slider in the user interface for this kind of analog 0-1 relevance feedback [223]. Peng criticized such continuous scale for relevance feedback because the user may not rate the samples of the retrieval results so precisely [203]. Instead, they put forward multi-class relevance feedback extending the classic binary relevance feedback (called "two-class relevance feedback" in their work), where the user assigns a class to each sampled retrieval result.

While existing methods in information retrieval integrate multiple features and learn feature importance based on information from relevance feedback, we advocate direct relevance feedback on multiple features in time series pattern search. Features like TF-IDF for text retrieval, or color/texture/shape for CBIR are primitive, and it is formidable to formulate feedback to them. However, the positions and sizes of the found patterns in time series pattern search are both user- and machine-friendly features. The user can easily make sense of and correct them, while the active learning algorithm can readily understand and adapt to such feedback. According to our experiments, our multi-feature feedback is feasible and favorable.

8.3 METHOD

The chapter proposes the distinctive user feedback type in Stage ❷ in Figure 7.2, which closes the data processing pipeline for feedback-driven pattern search. It provides training labels for

the classification model while infusing the user's domain knowledge into the pipeline.

Conventional active learning allows user feedback on relevance alone. For each found pattern, it asks the user to accept it, reject it, or abstain. Then, it uses the accepted found patterns as positive labels and the rejected ones as negative labels during model training. It ignores the indecisive ones. This can lead to a dilemma. As illustrated in Figure 8.1, if accepted, the found pattern as a positive label may baffle the model with its imperfection; conversely, if rejected, its partial correctness could be confusing; if ignored, the partial correctness of the found pattern is wasted.

Specifically for time-step-based classification as the essential stage in the nonmyopic search presented in Chapter 7, this imprecision in user feedback introduces incorrectly labeled training instances directly. As explained in Section 7.3.2, the labeled found patterns are not directly training instances fed into the model in time-step-based classification. Suppose that the user accepts a found pattern $F = S(\alpha, \beta)$. This single user label provides $\beta - \alpha$ positive training instances $\left[S\left(i - \zeta_\alpha,\, i + \zeta_\beta + 1\right)\right]_{\alpha \le i < \beta}$. However, by accepting F, the user may actually want to credit the target pattern $T = S(\gamma, \delta)$ that overlaps F. Suppose $\alpha < \gamma < \beta < \delta$, i.e., F starts and ends too early. To be precise, F overlaps with T on $S(\gamma, \beta)$, but F incorrectly detects the fragment $S(\alpha, \gamma)$ and misses $S(\beta, \delta)$. This user feedback correctly labels all time step in $S(\gamma, \beta)$ (corresponding to training instances $\left[S\left(i - \zeta_\alpha,\, i + \zeta_\beta + 1\right)\right]_{\gamma \le i < \beta}$) as positive, but also incorrectly labels all time step of $S(\alpha, \gamma)$ (corresponding to training instances $\left[S\left(i - \zeta_\alpha,\, i + \zeta_\beta + 1\right)\right]_{\alpha \le i < \gamma}$) positive.

To solve this problem, we propose multi-feature feedback, not only on the relevance of the found patterns but also on their positions and sizes. Pertaining to the same example above. If the algorithm suggests F, the user should be able to notice T in the user interface as well, otherwise, the user would not accept F. The user can adjust the position and size of F to align with T, so that $\alpha_{new} := \gamma$, $\beta_{new} := \delta$, and $F_{new} := T$. In this way, we resolve the predicament caused by the imprecision of the traditional binary relevance feedback.

Now that such adjustments are possible, the user may even accept a found pattern of low quality that overlaps only a little with a target pattern and that should be rejected by conventional binary relevance feedback. As long as the user notices the target pattern, the user can basically use the target pattern as a positive label, instead of the imprecise found pattern.

8.4 USER INTERFACE

Although multi-feature feedback is also applicable to traditional feedback-driven pattern search in time series, it works best together with nonmyopic search presented in Chapter 7. Therefore, we reused and extended NOOPS's VQS. In this section, we explain the feature closely related to user feedback and particularly to multi-feature feedback.

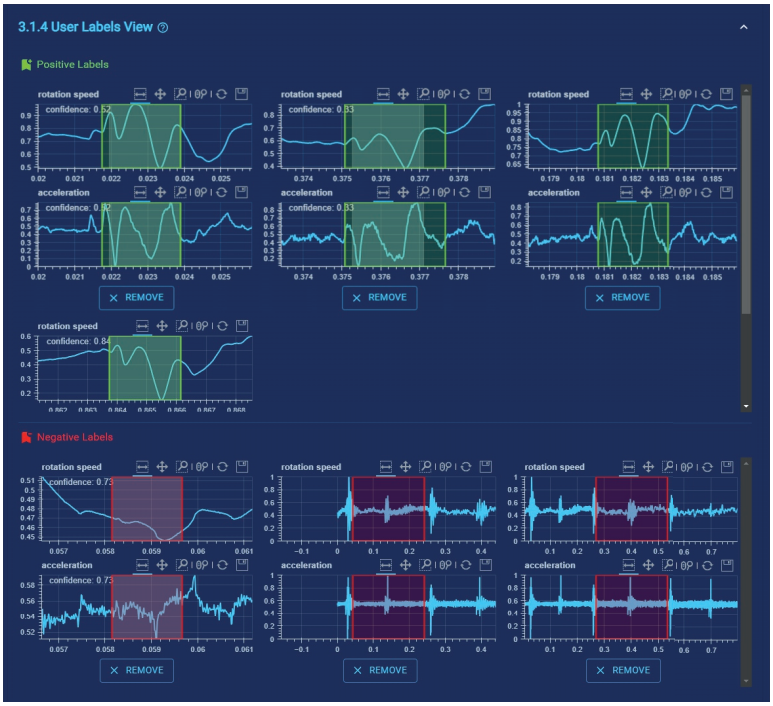


Figure 8.2: **User Labels View.** This view lists the positive and negative labels from the user for inspection and fine-tuning, before they are used in the next round of feedback-driven pattern search.

We have outlined various ways to adjust the relevance threshold η, e.g., dragging the threshold line in the additional track in the *Overview* explained in Section 7.4.1, moving a slide in a relevance score distribution histogram, and circling the desired patterns in the candidate dimensionality reduction scatter plot, both in the *Analysis View* explained in Section 7.4.2. Unfortunately, tuning η can potentially introduce false positives and false negatives. In such cases, the initial search cannot perfectly separate the tar-

get patterns from the background, calling for user feedback and the feedback-driven pattern search.

By clicking the corresponding button in the button group  shown in Figure 7.6, the user can mark an irrelevant found pattern as a negative label, or a relevant one (preferably with low relevance to help the model adjust the relevance perception) as a positive label. The labels are listed in the *User Labels View* shown in Figure 8.2. The user can fine-tune the green / red box representing the positive / negative label by moving it or dragging its left and right edges. A fixed blue box indicates the original found pattern as a reference, as the first two patterns in Figure 8.2 show.

Sometimes, the user may want to add labels spontaneously. When encountering a missing pattern in the *Overview*, the user can add it as a positive label. In particular, the user can also add one or two long negative labels to inform the model of the background, e.g., the right two negative labels in Figure 8.2. Because each user label already contains many training labels, as explained in the third benefit of time-step-based classification in Section 7.3.2. This was previously not possible because one user label used to correspond to only one training sample. Accordingly, it does not make sense to add a long negative label traditionally, because it, as a whole, cannot be a relevant pattern anyway. To add a label spontaneously, the user can draw a box with the box selection tool  , fine-tune it, mark it as positive or negative, and commit it, as shown at the bottom of Figure 7.5. In the *User Labels View*, the spontaneously added labels do not have the blue box representing the originating found pattern, as the third positively labeled pattern and the last two negatively labeled patterns in Figure 8.2 show.

8.5 EVALUATION

We evaluated the proposed multi-feature feedback on the positions and sizes of the found patterns besides their relevance against the baseline, binary relevance feedback, with a quantitative benchmark on retrieval accuracy. Then, we evaluated the usability of the VQS with a case study and an expert study.

8.5.1 *Experiment Setup*

In this experiment, we have used the same datasets / use cases as specified in Section 7.5.1. Moreover, we used nonmyopic search with `random forest` due to its strong performance in Section 7.5.

Designing experiments involving user feedback is challenging due to the subjectivity of user feedback. We propose to simulate user behavior, similar to works facing the same problem [235, 275]. Specifically, we created two agents, the BR-Agent and MF-Agent, to emulate Binary Relevance and Multi-Feature feedback, respectively.

The BR-Agent accepts a found pattern $F = S(\alpha, \beta)$ as a positive label if it satisfies the following two criteria. 1) F overlaps a ground truth $G = S(\gamma, \delta)$ no less than an IoU threshold (set to 30%), i.e., $\exists G \in \mathbb{G}$, such that $IoU(F, G) \geq 0.3$, where $\mathbb{G}$ denotes all ground truths; 2) To avoid duplicate found patterns, F should not overlap an already added positive label F' no less than the same IoU threshold, i.e., $\forall F' \in \mathbb{F}$, it follows $IoU(F, F') < 0.3$, where $\mathbb{F}$ denotes all found patterns before adding F as a new positive label. If F fails to fulfill the first criterion, the BR-Agent labels F negative. If F meets the first criterion but not the second, the BR-Agent ignores it.

The MF-Agent adds a positive label as long as a found pattern F intersects with a ground truth G, regardless of their IoU, i.e., $\exists G \in \mathbb{G}$, such that $F \cap G \neq \emptyset$. We designed this behavior because we assume that the user can notice the target pattern upon seeing the found pattern. However, instead of the found pattern F, MF-Agent adds the ground truth G as a positive label if G is not already added, which simulates the user's corrections on the position and size of the found pattern. The MF-Agent only label a found pattern negative when it does not overlap any ground truth at all.

The experiment began with the imperfect found patterns from the rule-based initial search in Section 7.3.1. Each agent assessed the found patterns based on the ground truths and returned five labels. Because `random forest` and `XGBoost` do not support iterative training only with new labels, we combined the new labels with the old to retrain all models. We conducted ten rounds of feedback-driven pattern search, which is sufficient for the results to converge to a stable retrieval accuracy in all our use cases.

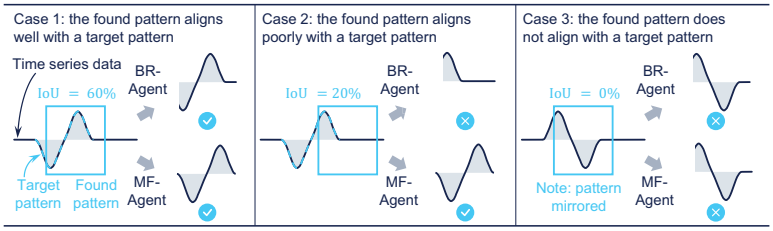


Figure 8.3: **BR-Agent vs. MF-Agent.** In a nutshell, the BR-agent labels a found pattern positive, when it sufficiently overlaps a ground truth; otherwise, it labels the found pattern negative. The MF-agent labels a found pattern positive, as long as it overlaps a ground truth (no matter how small the overlap is), and it adjusts the position and size of the found pattern to match the target pattern (using the ground truth as the positive label instead of the found pattern); the MF-agent labels a found pattern negative when it does not overlap any ground truth at all.

8.5.2 *Performance Comparison Between Multi-Feature and Binary Relevance Feedback*

Accuracy Evolution

Figure 8.4 shows the retrieval accuracy according to AP30 and AP50 after the rule-based initial search and after each of the ten feedback-driven pattern search rounds 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 light blue curves denoting multi-feature feedback above the dark blue (nearly black) curves denoting binary relevance feedback.

As indicated by the rising AP30 in the `Deep Valve Whole Signal` use case, both feedback types may improve the search result from rule-based initial search (Feedback Round 0 in Figure 8.4). The improvement occurred mainly in the first feedback round, but this was not always the case. For instance, in the `Variable Displacement` use case shown in Figure 8.4, retrieval accuracy only smoothed after six feedback rounds.

The light blue curves denoting multi-feature feedback are consistently above the dark blue curves denoting binary relevance feedback, verifying that multi-feature feedback outperforms binary relevance feedback. In fact, binary relevance feedback does not always work, as, for instance, the low dark blue lines in the

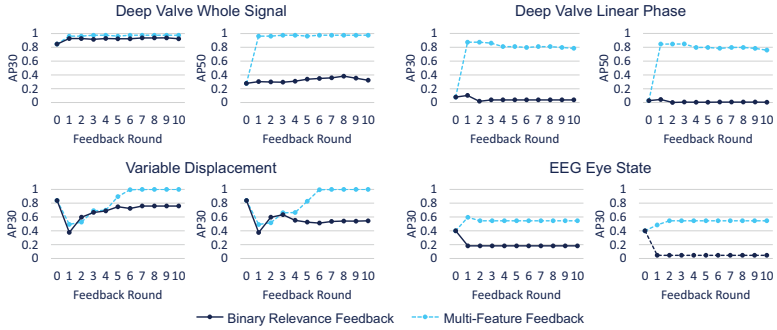


Figure 8.4: **Evolution of Retrieval Accuracy.** Both feedback types can improve retrieval accuracy, as indicated, for instance, by the rising AP30 in the `Deep Valve Whole Signal` use case. The improvement mainly happens in the first feedback-driven pattern search round. Multi-feature feedback outperforms binary relevance feedback, both in terms of the correct hits according to AP30 and the quality of the found patterns according to AP50. In fact, binary relevance feedback fails to improve retrieval accuracy if measured by the stricter metric AP50.

`Deep Valve Linear Phase` use case show. In this use case, the temporal context of the target patterns needs to be taken into consideration during pattern search. Hence, correction of positions and sizes can be especially helpful in this case. The `EEG Eye State` use case provides another example, where multi-feature feedback also significantly outperforms the binary relevance feedback. In this use case, boundaries between the target patterns and the background are relatively difficult to locate, requiring the user's help with delineation.

Not only does multi-feature feedback catch more target patterns correctly, as indicated by the higher AP30 in all use cases, but the true positives are also of higher quality (more precise locations and sizes), as indicated by the higher AP50 in all use cases. 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 was observed after the first feedback round in the `Variable Displacement` use case. This was probably due to

the limit we set for the feedback. Specifically, we allowed only five new labels in each feedback round. The rule-based initial search worked already 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.

Converged Accuracy

According to Figure 8.4, the retrieval accuracy converged to a stable metric score in all our use cases after ten feedback-driven pattern search rounds. To better compare the destination or stable state to which the feedback-driven pattern search led, we plotted this converged retrieval accuracy with a series of APs in Figure 8.5. In all use cases, the light blue bars representing multi-feature feedback are taller than the dark blue bars representing binary relevance feedback, showing the general performance gain with multi-feature feedback. When 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 led to higher retrieval accuracy, implying that the multi-feature feedback catches the target patterns better. Furthermore, the stricter the metric, the larger the performance gain appears. To be precise, the metric scores with binary relevance feedback dwindle rapidly with increasing strictness of the metric but slide only mildly with multi-feature feedback, suggesting better quality of the found patterns with multi-feature feedback, indicating the higher quality of the found patterns, albeit potential missing patterns as in the `EEG Eye State` use case. In fact, binary relevance feedback does not work at all in the more challenging `Deep Valve Linear Phase` use case and `EEG Eye State` use case. The target patterns in the `Deep Valve Linear Phase` use case, as illustrated in Figure 7.10, 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 steps 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 steps. Moreover, the special data augmentation that labels the time steps surrounding a positive label, introduced as the third benefit in Section 7.3.2, requires that the labeled found patterns have correct start and end time steps. Otherwise, it may cause incorrect negative training instances. All in all, we can witness that the adjustment of the pattern positions and sizes contributes greatly to the detection of the target patterns.

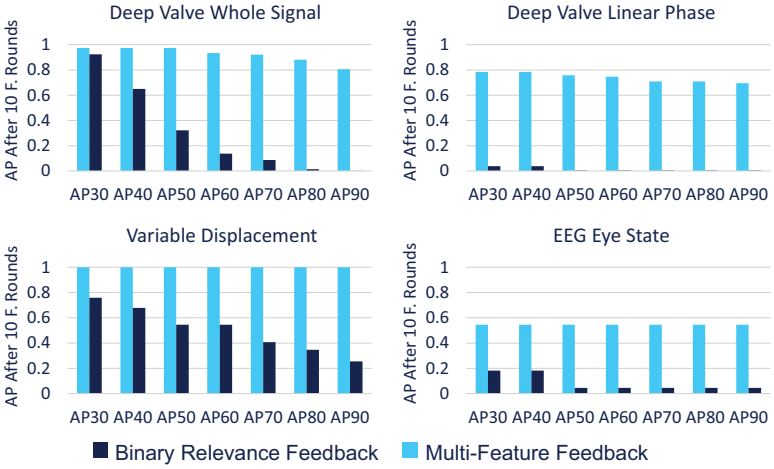


Figure 8.5: **Converged Retrieval Accuracy.** After ten feedback-driven pattern search rounds, multi-feature feedback lifted the retrieval accuracy to a higher level than binary relevance feedback. The stricter the metric, the larger the performance gain appears, implying that the found patterns are of high quality, notwithstanding possible missing patterns (as in the `EEG Eye State` use case).

As shown in the simplified Table 11 derived from the second sheet in "evaluation.xlsx" under `https://osf.io/36kf2`, multi-feature feedback raised retrieval accuracy by 0.35 according to AP30 and by 0.59 according to AP50 on average in our experiment.

In conclusion, multi-feature feedback outperforms binary relevance feedback and should be the preferred choice for feedback-driven pattern search in time series, especially in combination with the nonmyopic search.

Use Case	AP30			AP50		
	MF	BR	Acc. Gain	MF	BR	Acc. 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 11: **(Oversimplified) Performance Gain with Multi-Feature Feedback.** Multi-feature feedback raised retrieval accuracy by 0.35 according to AP30 and by 0.59 according to AP50 on average in our experiment. (DV-Whole: `Deep Valve Whole Signal`, DV-Linear: `Deep Valve Linear Phase`, VD: `Variable Displacement`, EEG: `EEG Eye State`)

8.5.3 *Case Study*

Lastly, we conducted a case study to demonstrate the features of the user interface and an expert study with two calibration engineers to validate its usability. Please note that they are not limited to multi-feature feedback presented in this chapter, but also for the previous Chapter 7 because they share the same VQS.

This section outlines a cohesive workflow. We used the `Variable Displacement` use case for demonstration. Its two tracks roughly represent engine rotational speed and acceleration. The target patterns signify the reactivation of deactivated cylinders, harboring NVH issues of interest to our engineers. We omit screenshots in this subsection because there are too many interactions and large areas of visualizations with small details. However, we deliberately captured screenshots pertaining to this use case in Section 7.4 as well as Section 8.4 and will refer to them in this subsection. Please refer to the video under `https://www.youtube.com/watch?v=7-MPiC7asBg` for the complete workflow.

We loaded and explored files, designated a target pattern as the query, and launched the initial search. After obtaining results, we checked the sanity of the found patterns in the *Found Patterns View* in Figure 7.6. Notably, the force graph in Figure 7.7.c displayed two nodes distant from the cluster around the yellow query node. Upon perturbation, we observed swift restoration of the stable layout, indicating that these nodes were likely false negatives. To highlight them in all views, we clicked on them

in the force graph. In the *Found Patterns View* in Figure 7.6 (the rightmost two patterns), we observed their partial resemblance to the query in terms of shape. After clicking  to scale all found patterns to the query's value range, we discovered that the shapes fluctuated in a much smaller value range and could not relate to NVH issues. To remove these two false positives, we reduced the number of found patterns by two in the first input field below the relevance score distribution histogram in Figure 7.7.a. The relevance threshold slider left to the histogram was adjusted automatically. We went on to examine the candidates near the yellow query in the scatter plot Figure 7.7.b for potential false negatives. Fortunately, there were no false negatives in this case; however, if any existed, we could mark them as positive labels. To enhance recognition of the background, we added a long red negative label in the *Overview* in Figure 7.5. In the *User Labels View* in Figure 8.2, we adjusted the label positions and sizes and began the feedback-driven pattern search. In the improved result set, we found two patterns shown in Figure 7.8. The relevance profiles indicated problems during classification. We chose them as positive labels and modified the position and size of the one in Figure 7.8.b. Following another feedback-driven pattern search, we obtained a nearly perfect result. Finally, we systematically examined the found patterns in the *Found Patterns View* and analyzed them with more context in the *Overview*.

8.5.4 *Expert Study*

We conducted expert studies with two experts to validate the usability of NOOPS. Due to space limits, please refer to Appendix I under `https://osf.io/f6phs` for more details like the procedure, pre-defined tasks, questionnaire, protocol, screenshots, and our analysis as well as reflection.

Expert A is a senior technical consultant with 15-year experience in function development and calibration for gasoline engines. Expert B is an on-board diagnostics engineer with one-year experience in air, fuel, and thermal-related features of gasoline engines. Each study lasted one hour, began with a five-minute introduction, continued with a 15-minute live demonstration iterating the use case in Section 8.5.3, followed by a 30-minute hands-on test with predefined tasks, and concluded with a 10-minute interview. During the demonstration and the hands-on test, we encouraged the experts to provide feedback.

While we cannot report on all the findings, we will highlight here expert feedback centered around our three core analysis

goals: the handling of 1) false negatives (missing target patterns), 2) false positives (irrelevant found patterns), and 3) validating true positives (relevant found patterns).

Both experts were initially unfamiliar with our dimensionality reduction scatter plot and its applicability to searching for false negatives. However, they quickly adapted during the hands-on test. Expert A stated: *"While it [dimensionality reduction] is a tough and sophisticated concept; I think this is helping me to understand the similarity between events"*. They expressed both concerns about the learning curve. Expert A continued: *"[...] It is critical to embed your explanation during and after the live demo somehow in the app itself."*. Overall, the feedback on our visual mapping of pattern candidate positions (in the scatter plot) was positive. Before NOOPS, they had no way to find false negatives. The experts understood that similar patterns / points are closer, and the judgment based on the relative distance to the query helped them to spot false negatives more efficiently.

As for the false positives, the feedback was less conclusive. The tasks were primarily designed to be accomplished with the assistance of the force graph. Expert B states: *"While I get the general idea, this is still a gimmick for me. I can faster gauge the similarity by looking at the line charts of the found patterns in the Found Patterns View directly."*.

NOOPS's user interface excelled in the validation of true positives. Both experts successfully deduced that showing only true positives by tuning the relevance threshold η is impossible if there were false positives with higher relevance (the model cannot separate the relevant and irrelevant patterns perfectly). Expert B stated: "*Finding automatically a perfect separation [between the two classes] might be impossible, but the draggable horizontal line [for adjusting η] is an effective solution.*" Combining this interactive filtering with our coordinated view system and, more importantly, with the feedback-driven pattern search, was perceived by both experts as a core solution to the problem.

Our experts validated that NOOPS effectively addresses their use cases and the user interface supports our feedback-driven pattern search effectively.

8.6 DISCUSSION

Multi-feature feedback is primarily an add-on for nonmyopic search presented in Chapter 7, though also compatible with the traditional search techniques. It aims to improve the effectiveness of user feedback in feedback-driven pattern search.

We are curious if feedback on even more features or more fine-granular feedback is possible and beneficial. For instance, we can model a found pattern as a coarse spline and invite users to modify individual support points. We can also invite the user to rate the quality of a found pattern in a Likert scale, or even continuous by moving a slider. It remains to be examined whether such features contributes to the improvement of retrieval rather than causing confusion and overhead.

Currently, we show found patterns with high relevance to the user for feedback. This is suboptimal because the notion of relevance may narrow down to a few subclasses instead of subsuming a mix of subclasses in the target class, i.e., confirmation bias. We would like to examine the idea of suggesting moderately confident pattern candidates that are dissimilar to other very confident found patterns for feedback.

8.7 CONCLUSION

In this chapter, we have proposed a new type of user feedback for feedback-driven pattern search in MTS called multi-feature feedback. It advocates feedback not only on the binary relevance of the found patterns alone, as existing methods suggest, but also on their positions and sizes. To our surprise and dismay, we found the traditional binary relevance insufficient for the retrieval results to improve significantly. On the other hand, multi-feature feedback worked in all our real-world industrial use cases and achieved 0.35 higher accuracy. In prospect, we expect the establishment of multi-feature feedback in feedback-driven pattern search. In the future, we will optimize individual stages in NOOPS's pipeline, study visualizations especially for the time-step-based relevance, and examine possibilities of intelligent recommendations for user feedback.

9 CONCLUSION

In this thesis, we have studied the research question **How to query patterns in MTS accurately, efficiently, and understandably?** Centering around this question, we have proposed three tools/VQSs (PSEUDo, SAXRegEx, and NOOPS) incorporating four techniques (steerable LSH, query expansion in SAX, nonmyopic search, and multi-feature feedback) and established a taxonomy for time series representations.

We have achieved all four requirements collected in Section 1.3. All our proposed methods work for multivariate time series, and some (PSEUDo and SAXRegEx) even prefer multivariate cases, complying with R1 Multivariate Suitability. We improve accuracy through retrieval invariance of the algorithm (SAXRegEx), through a traversing approach that utilizes the structure of the data (nonmyopic search), and through active learning that better respects the task (multi-feature feedback), trying to satisfy R2 Sufficient Accuracy. We accelerate the search through an efficiency-oriented data representation (PSEUDo) for rule-based pattern search and through a more efficient data traversing other than sequential scanning (SAXRegEx and nonmyopic search) for model-based pattern search, striving to meet R3 High Efficiency. We achieve interpretability of the algorithms through a feature-importance-based model update mechanism (PSEUDo) and an explicit description of desired pattern distortions (SAXRegEx), endeavoring to fulfill R4 Explainability.

We addressed all challenges in Section 1.2, attained all goals except the last two could-have/nice-to-have goals in Section 1.4, made the last four contributions, and are working on the first two contributions listed in Section 1.5.

We summarize the challenges, requirements, goals, and contributions addressed by each major chapter in Table 12.

Table 12: **Challenges, Requirements, Goals, and Contributions Addressed in Each Chapter.** empty: no meaningful overlap or function not supported, "+": established techniques used or function (trivially) supported, "++": tangential or minor contributions, "+++": major contributions.

Chapter	3 Survey	4 Metrics	5 PSEUDo	6 SAXRegEx	7 NOOPS 1	8 NOOPS 2
Challenges						
C1 Temporal Length				+	+	
C2 High Dimensions			+++	+++		
C3 Duration Scaling				+++	+++	++
C4 Inter-Track T. Shifts				+++		
C5 Context Oblivious.					+++	
C6 Unexplainability			+++	+	+	
C7 Inefficient Training			+++			
C8 Meager Labels					++	++
C9 Subj. Relevance			++		++	+++
Requirements						
R1 Multivar. Suitability		+	+++	+++	+	+
R2 Sufficient Accuracy			+		+++	+++
R3 High Efficiency			+++	+++	++	
R4 Explainability			++	+	+	
Goals						
MG1 STAR Analysis	+++	+	+	+	+	+
MG2 Baseline Alg.			+	+	+	+
MG3 Basic Algorithm			+	+	+	
MG4 Alg. Benchmark			+	+	+	+
MG5 Basic App			+	+	+	+
MG6 Basic Vis.			+	+	+	+
MG7 Query-by-Exam.			+	+	+	
MG8 Case Study				+		+
SG1 Multiv. Suitability			+++	+++	+	+
SG2 Single Example			+	+	+	
SG3 Performance Gain			+++	+++	+++	+++
SG4 Adaptive Alg.			+++		+++	+
SG5 Streaml. Workflow			+	+	+	+
SG6 Click & Run				+	+	
⋮						

Table 12: Challenges, Requirements, Goals, and Contributions Addressed in Each Chapter – Continued

Chapter	3 Survey	4 Metrics	5 PSEUDo	6 SAXRegEx	7 NOOPS 1	8 NOOPS 2
SG7 Expert Studies			+			+
CG1 Transf. Survey	+++					
CG2 Track Importance			+++			
CG3 Eval. Framework		+++				
CG4 Query-by-Sketch						
CG5 Intelligent Assist.						
Contributions						
CO1 Taxonomy	+++					
CO2 Eval. Framework		+++				
CO3 Adap. Scal. Repr.			+++			
CO4 Search Invariance				+++		
CO5 Nonmyopic Sear.					+++	
CO6 Multi-Feat. Feed.						+++

The theoretical findings of our research are published in two major works [275, 276], submitted and to be submitted in another two publications. The developed codes are deployed in our industrial collaborator IAV and its customers, actively helping automotive engineers trace events of interest in various measurements.

In prospect, we expect 1) a surge of extensions of existing methods for univariate time series analysis to multivariate cases because of the solid practical need and the potential of many methods; 2) closer collaboration of hashing algorithms and data science methods due to the exploding data volume to process; 3) standardization of nonmyopic search and multi-feature feedback for model-based and feedback-driven pattern search, as they bring sheer benefits and cause no harm.

Due to limited resources, our work forgoes exploration in the following interesting directions, which opens avenues for future research.

Comprehensive Search Invariance. We have endeavored to capture as many time series pattern distortions and enable as much retrieval invariance as possible while remaining efficient and understandable. At present, no algorithm captures all distortions, especially in multivariate cases. Articulating the idea in Chapter 5 (scalable representation) with Chapter 6 (query expansion), we would like to extend multivariate DWT [45, 207, 217] because

it is a scalable representation (like LSH in PSEUDo) exploiting an integral transform and supports capturing horizontal scaling (duration variation) out of the box (like regex quantifiers in SAXRegEx). Nonetheless, we believe that it still has a long way to go before the invention of a method that can accurately and efficiently address all common pattern distortions. Currently, a method ensemble could be a practical workaround.

Versatile Method. Apropos of a method ensemble, we have addressed the challenges, requirements, and goals with different algorithms and tools. Accordingly, we provide our users with a toolbox of methods including the state-of-the-art approaches in a single platform web application, with guidelines on the typical use cases for each method. It would be ideal if there were a method fitting most use cases. Alternatively, it would be rosy to have a set of criteria for choosing methods systematically and automatically based on the data and the query. For instance, is something like "large time series models" for time series retrieval possible, just like large language models for natural language processing that have witnessed huge success recently?

Unified Evaluation. As mentioned in Section 4.1, the research field MTS pattern search lacks a standard evaluation framework. We consider three gaps to fill: 1) a repository of labeled datasets for time series pattern search, like the dataset repository [69] for time series classification, which fuels an explosion of research for this task; 2) unified performance metrics, which we have addressed in Chapter 4 and are going to extend with our ongoing visual analytics approach; 3) a leaderboard site for collecting methods and ranking them according to their performance on various datasets measured by diverse metrics, like `https://www.kaggle.com` and `https://huggingface.co/`, which our ongoing project contemplates.

Scalable Visualization. While all our algorithms support multivariate time series, our visualizations do not scale to large time series. We can see three avenues. 1) In terms of temporal length (C1 Temporal Length), we can scale the time series along the time axis heterogeneously, e.g., enlarging the patterns of interest and shrinking the space between them. 2) In terms of the dimensions (C2 High Dimensions), we can utilize visualizations that demand less vertical screen space, like horizon diagrams [116], and color-coded pixels [105]. 3) Since scalable visualizations may distort the data or weaken the expressiveness of details, we can examine coordinated views showing overviews and undistorted zoomed-in details simultaneously.

Intelligent Interactions. Our work focuses more on data mining than visual analytics. We envision three research directions in terms of intelligent visualizations and interactions. 1) Intelligent assistance during query definition: for instance, during query-by-sketch, the VQS could provide typical patterns in the time series database that match the user's ongoing drawing, as mentioned in Section 5.6. 2) Intelligent result analysis: instead of showing domain users scatter plots or other data-science-savvy visualizations, which our users criticized, the VQS could analyze such diagrams algorithmically and only expose simple interactions for the user, e.g., alerting the user that a pattern is a potential false negative, as it appears close to the query even more than some found patterns in a latent space, as the case study in Section 8.5.4 inspired us. 3) Unbiased found patterns selected to be shown to the user for feedback: the VQS should not only draw samples for user feedback from highly confident found patterns, but also, e.g., from the ones that the algorithm is unsure about, or from the found patterns that differ much from the rest, as Section 8.6 has pointed out.

Knowledge Externalization. Finally, while we strive to infuse domain knowledge into the data processing pipeline seamlessly, knowledge-assisted visual analytics inspires us to externalize tacit expert knowledge into explicit knowledge [91], e.g., through semantic interaction analysis. We can imagine that certain data processing operations, such as the necessity of data normalization, can be inferred from user feedback. The VQS can extract, derive, and manage them as explicit knowledge/rules.

BIBLIOGRAPHY

[1] C. B. Aditya Satrio, W. Darmawan, B. U. Nadia, and N. Hanafiah. Time series analysis and forecasting of coronavirus disease in indonesia using arima model and prophet. *Procedia Computer Science*, 179:524–532, 2021. DOI 10.1016/j.procs.2021.01.036.

[2] R. Agrawal, K. I. Lin, H. S. Sawhney, and K. Shim. Fast similarity search in the presence of noise, scaling, and translation in time-series databases. In *Proceeding of the 21th International Conference on Very Large Data Bases*, pages 490–501, 1995.

[3] S. Ahmed, I. E. Nielsen, A. Tripathi, S. Siddiqui, R. P. Ramachandran, and G. Rasool. Transformers in time-series analysis: A tutorial. *Circuits, Systems, and Signal Processing*, 2023. DOI 10.1007/s00034-023-02454-8.

[4] S. Alaee, R. Mercer, K. Kamgar, and E. Keogh. Time series motifs discovery under dtw allows more robust discovery of conserved structure. *Data Mining and Knowledge Discovery*, 35(3):863–910, 2021. DOI 10.1007/s10618-021-00740-0.

[5] A. Alqahtani, M. Ali, X. Xie, and M. W. Jones. Deep time-series clustering: A review. *Electronics*, 10(23):3001, 2021. DOI 10.3390/electronics10233001.

[6] H. André-Jönsson. *Indexing Strategies for Time Series Data*. Department of Computer and Information Science, Linköpings universitet, 2002.

[7] H. André-Jönsson and D. Z. Badal. Using signature files for querying time-series data. In J. Komorowski and J. Zytkow, editors, *Principles of Data Mining and Knowledge Discovery*, pages 211–220, Berlin, Heidelberg, 1997. Springer Berlin Heidelberg. ISBN 978-3-540-69236-2.

[8] N. Andrienko, G. Andrienko, G. Fuchs, A. Slingsby, C. Turkay, and S. Wrobel. *Visual Analytics for Data Scientists*. Springer, [S.l.], 2020. ISBN 978-3-030-56146-8.

[9] M. Angelaccio, T. Catarci, and G. Santucci. Query by diagram: A fully visual query system. *Journal of Visual*

Languages & Computing, 1(3):255–273, 1990. DOI 10.1016/S1045-926X(05)80009-6.

[10] A. P. Ansuj, M. E. Camargo, R. Radharamanan, and D. G. Petry. Sales forecasting using time series and neural networks. *Computers & Industrial Engineering*, 31(1-2):421–424, 1996. DOI 10.1016/0360-8352(96)00166-0.

[11] M. Aoki. *State Space Modeling of Time Series*. Springer Berlin Heidelberg, Berlin, Heidelberg, 1990. ISBN 978-3-540-52870-8. DOI 10.1007/978-3-642-75883-6.

[12] D. Arendt, E. Saldanha, R. Wesslen, S. Volkova, and W. Dou. Towards rapid interactive machine learning: evaluating tradeoffs of classification without representation. In W. Fu, S. Pan, O. Brdiczka, P. Chau, and G. Calvary, editors, *Proc. 24th Int. Conf. on Intelligent User Interfaces, IUI 2019*, pages 591–602. ACM, 2019. DOI 10.1145/3301275.3302280.

[13] C. K. Assaad, E. Devijver, and E. Gaussier. Survey and evaluation of causal discovery methods for time series. *Journal of Artificial Intelligence Research*, 73:767–819, 2022. DOI 10.1613/jair.1.13428.

[14] H. K. Azad and A. Deepak. Query expansion techniques for information retrieval: A survey. *Information Processing & Management*, 56(5):1698–1735, 2019. DOI 10.1016/j.ipm.2019.05.009.

[15] A. Bagnall, J. Lines, A. Bostrom, J. Large, and E. Keogh. The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances // 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. DOI 10.1007/s10618-016-0483-9.

[16] Z. Banko, L. Dobos, and J. Abonyi. Dynamic principal component analysis in multivariate time-series segmentation. *Conservation, Information, Evolution-towards a sustainable engineering and economy*, 1(1):11–24, 2011.

[17] Z. Bar-Joseph. Analyzing time series gene expression data. *Bioinformatics (Oxford, England)*, 20(16):2493–2503, 2004. DOI 10.1093/bioinformatics/bth283.

[18] M. C. N. Barioni, H. Razente, A. M. R. Marcelino, A. J. M. Traina, and C. Traina. Open issues for partitioning clustering methods: An overview. *WIREs Data Mining and Knowledge Discovery*, 4(3):161–177, 2014. DOI 10.1002/widm.1127.

[19] O. E. Barndorff-Nielsen and N. Shephard. Econometric analysis of realized volatility and its use in estimating stochastic volatility models. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 64(2):253–280, 2002. DOI 10.1111/1467-9868.00336.

[20] M. Behrisch, F. Korkmaz, L. Shao, and T. Schreck. Feedback-driven interactive exploration of large multidimensional data supported by visual classifier. In *2014 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pages 43–52, 2014. DOI 10.1109/VAST.2014.7042480.

[21] K. Benidis, S. S. Rangapuram, V. Flunkert, Y. Wang, D. Maddix, C. Turkmen, J. Gasthaus, M. Bohlke-Schneider, D. Salinas, L. Stella, F. X. Aubet, L. Callot, and T. Januschowski. Deep learning for time series forecasting: Tutorial and literature survey. *ACM Comput. Surv.*, 55(6):1–36, 2023. DOI 10.1145/3533382.

[22] A. Berg, T. McMurry, and D. N. Politis. Testing time series linearity. In *Time Series Analysis: Methods and Applications*, volume 30 of *Handbook of Statistics*, pages 27–42. Elsevier, 2012. ISBN 9780444538581. DOI 10.1016/B978-0-444-53858-1.00002-8.

[23] D. J. Berndt and J. Clifford. Using dynamic time warping to find patterns in time series. In *KDD workshop*, volume 10, pages 359–370, 1994.

[24] P. Bhattacharjee and P. Mitra. A survey of density based clustering algorithms. *Frontiers of Computer Science*, 15(1), 2021. DOI 10.1007/s11704-019-9059-3.

[25] K. Bittner and I. Spence. *Use Case Modeling*. The Addison-Wesley object technology series. Addison-Wesley, Boston, Mass., 2003. ISBN 978-0-201-70913-1.

[26] A. Blázquez-García, A. Conde, U. Mori, and J. A. Lozano. A review on outlier/anomaly detection in time series data. *ACM Comput. Surv.*, 54(3):1–33, 2022. DOI 10.1145/3444690.

[27] T. Bollerslev. Generalized autoregressive conditional heteroskedasticity. *Journal of Econometrics*, 31(3):307–327, 1986. DOI 10.1016/0304-4076(86)90063-1.

[28] P. Bonizzi, J. M. H. Karel, O. Meste, and R. L. M. Peeters. Singular spectrum decomposition: A new method for time series decomposition. *Advances in Adaptive Data Analysis*, 06 (04):1450011, 2014. DOI 10.1142/S1793536914500113.

[29] G. E. P. Box and D. R. Cox. An analysis of transformations. *Journal of the Royal Statistical Society. Series B (Methodological)*, 26(2):211–252, 1964. ISSN 00359246. URL http://www.jstor.org.proxy.library.uu.nl/stable/2984418.

[30] G. E. P. Box and G. M. Jenkins. *Time Series Analysis; Forecasting and Control*. Holden-Day series in time series analysis. Holden-Day, San Francisco, 1970. ISBN 0816210942.

[31] G. E. P. Box, G. M. Jenkins, and G. C. Reinsel. *Time Series Analysis*. Wiley, 2008. ISBN 9780470272848. DOI 10.1002/9781118619193.

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

[33] M. Brehmer and T. Munzner. A multi-level typology of abstract visualization tasks. *IEEE Trans. Vis. Comput. Graph.*, 19(12):2376–2385, 2013. DOI 10.1109/TVCG.2013.124.

[34] P. M. T. Broersen and S. de Waele. Some benefits of aliasing in time series analysis. In *2000 10th European Signal Processing Conference*, pages 1–4, 2000.

[35] E. Brophy, Z. Wang, Q. She, and T. Ward. Generative adversarial networks in time series: A systematic literature review. *ACM Comput. Surv.*, 55(10):1–31, 2023. DOI 10.1145/3559540.

[36] P. Buono, A. Aris, C. Plaisant, A. Khella, and B. Shneiderman. Interactive pattern search in time series. *Visualization and Data Analysis 2005*, 5669:175–186, 2005. DOI 10.1117/12.587537.

[37] J. A. Cadzow, B. Baseghi, and T. Hsu. Singular-value decomposition approach to time series modelling. *IEE Proceedings F Communications, Radar and Signal Processing*, 130 (3):202, 1983. DOI 10.1049/ip-f-1.1983.0034.

[38] B. Cai, S. Yang, L. Gao, and Y. Xiang. Hybrid variational autoencoder for time series forecasting. *Knowledge-Based Systems*, 281:111079, 2023. DOI 10.1016/j.knosys.2023.111079.

[39] V. Capasso and D. Bakstein. *An Introduction to Continuous-Time Stochastic Processes*. Springer International Publishing, Cham, 2021. ISBN 978-3-030-69652-8. DOI 10.1007/978-3-030-69653-5.

[40] B. C. M. Cappers, P. N. Meessen, S. Etalle, and J. J. van Wijk. Eventpad: Rapid malware analysis and reverse engineering using visual analytics. In D. Staheli, C. L. Paul, J. Kohlhammer, D. M. Best, S. Trent, N. Prigent, R. Gove, and G. Sauer, editors, *15th IEEE Symposium on Visualization for Cyber Security, VizSec 2018, Berlin, Germany, October 22, 2018*, pages 1–8. IEEE, 2018. DOI 10.1109/VIZSEC.2018.8709230.

[41] C. Carpineto and G. Romano. A survey of automatic query expansion in information retrieval. *ACM Comput. Surv.*, 44 (1), 2012. DOI 10.1145/2071389.2071390.

[42] T. Catarci, M. F. Costabile, S. Levialdi, and C. Batini. Visual query systems for databases: A survey. *Journal of Visual Languages & Computing*, 8(2):215–260, 1997. DOI 10.1006/jvlc.1997.0037.

[43] S. Ceri, A. Bozzon, M. Brambilla, E. Della Valle, P. Fraternali, and S. Quarteroni. *Web Information Retrieval*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013. ISBN 978-3-642-39313-6. DOI 10.1007/978-3-642-39314-3.

[44] K. Chakrabarti, E. Keogh, S. Mehrotra, and M. Pazzani. Locally adaptive dimensionality reduction for indexing large time series databases. *ACM Trans. Database Syst.*, 27(2):188–228, 2002. DOI 10.1145/568518.568520.

[45] K. P. Chan and A. W. C. Fu. Efficient time series matching by wavelets. In *Proceedings 15th International Conference on Data Engineering (Cat. No.99CB36337)*, pages 126–133, 1999. DOI 10.1109/ICDE.1999.754915.

[46] G. Chatzigeorgakidis, S. Dimitrios, K. Patroumpas, T. Palpanas, S. Athanasiou, and S. Skiadopoulos. Twin subsequence search in time series. *CoRR*, abs/2104.06874, 2021.

[47] G. Chatzigeorgakidis, S. Dimitris, K. Patroumpas, T. Palpanas, S. Athanasiou, and S. Skiadopoulos. Efficient range

and knn twin subsequence search in time series. *IEEE Transactions on Knowledge and Data Engineering*, 2022. DOI 10.1109/TKDE.2022.3167257.

[48] L. Chen and R. Ng. On the marriage of lp-norms and edit distance. In *Proceedings of the 30th International Conference on Very Large Data Bases*, volume 30, pages 792–803, 2004.

[49] N. Chen, H. Tu, X. Duan, L. Hu, and C. Guo. Semisupervised anomaly detection of multivariate time series based on a variational autoencoder. *Applied Intelligence*, 2022. DOI 10.1007/s10489-022-03829-1.

[50] Z. Chen, M. Ma, T. Li, H. Wang, and C. Li. Long sequence time-series forecasting with deep learning: A survey. *Information Fusion*, 97:101819, 2023. DOI 10.1016/j.inffus.2023.101819.

[51] S. H. Cheng, C. Augustin, A. Bethel, D. Gill, S. Anzaroot, J. Brun, B. DeWilde, R. C. Minnich, R. Garside, Y. J. Masuda, D. C. Miller, D. Wilkie, S. Wongbusarakum, and M. C. McKinnon. Using machine learning to advance synthesis and use of conservation and environmental evidence. *Conservation Biology*, 32(4):762–764, 2018. ISSN 08888892. URL http://www.jstor.org/stable/44973913.

[52] L. Chi and X. Zhu. Hashing techniques: A survey and taxonomy. *ACM Comput. Surv.*, 50(1):1–36, 2018. DOI 10.1145/3047307.

[53] G. Chiarot and C. Silvestri. Time series compression survey. *ACM Comput. Surv.*, 55(10):1–32, 2023. DOI 10.1145/3560814.

[54] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation, 2014.

[55] F. Chollet. *Deep Learning mit Python und Keras: Das Praxis-Handbuch vom Entwickler der Keras-Bibliothek*. mitp, Frechen, 1. auflage edition, 2018. ISBN 978-3-95845-839-0.

[56] G. G. Chowdhury. *Introduction to Modern Information Retrieval*. Facet Publ, London, 3. ed. edition, 2010. ISBN 185604694X.

[57] C. S. J. Chu. Time series segmentation: A sliding window approach. *Information Sciences*, 85(1-3):147–173, 1995. DOI 10.1016/0020-0255(95)00021-G.

[58] F. L. K. Chung, T. C. Fu, W. P. R. Luk, and V. T. Y. Ng. Flexible time series pattern matching based on perceptually important points. In *Workshop on Learning from Temporal and Spatial Data in International Joint Conference on Artificial Intelligence*, pages 1–7. Conference presentation (not published in journal/proceeding/book), 2001.

[59] D. Clegg and R. Barker. *CASE Method Fast Track: A RAD approach*. Computer aided systems engineering. Addison-Wesley Pub. Co, Wokingham, England and Reading, Mass, 1. pr edition, 1994. ISBN 978-0-201-62432-8.

[60] W. S. Cleveland. The inverse autocorrelations of a time series and their applications. *Technometrics*, 14(2):277–293, 1972. DOI 10.1080/00401706.1972.10488914.

[61] M. Cochez and H. Mou. Twister tries: Approximate hierarchical agglomerative clustering for average distance in linear time. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, pages 505–517, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450327589. DOI 10.1145/2723372.2751521.

[62] A. Collins Jackson and S. Lacey. The discrete fourier transformation for seasonality and anomaly detection of an application to rare data. *Data Technologies and Applications*, 54 (2):121–132, 2020. DOI 10.1108/DTA-12-2019-0243.

[63] J. J. F. Commandeur. *An Introduction to State Space Time Series Analysis*. Practical econometrics series. Oxford University Press, Oxford, 2007. ISBN 0199228876. URL http://site.ebrary.com/lib/alltitles/docDetail.action?docID=10199686.

[64] A. A. Cook, G. Misirli, and Z. Fan. Anomaly detection for iot time-series data: A survey. *IEEE Internet of Things Journal*, 7(7):6481–6494, 2020. DOI 10.1109/JIOT.2019.2958185.

[65] M. Correll and M. Gleicher. The semantics of sketch: Flexibility in visual query systems for time series data. In *2016 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pages 131–140, 2016. DOI 10.1109/VAST.2016.7883519.

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

[67] H. Dadkhahi, M. F. Duarte, and B. Marlin. Isomap out-of-sample extension for noisy time series data. In *2015 IEEE 25th International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6. IEEE, 2015. ISBN 978-1-4673-7454-5. DOI 10.1109/MLSP.2015.7324314.

[68] A. S. Das, M. Datar, A. Garg, and S. Rajaram. Google news personalization: Scalable online collaborative filtering. In *Proceedings of the 16th International Conference on World Wide Web*, pages 271–280, New York, NY, USA, 2007. Association for Computing Machinery. ISBN 9781595936547. DOI 10.1145/1242572.1242610.

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

[70] C. Deb, F. Zhang, J. Yang, S. E. Lee, and K. W. Shah. A review on time series forecasting techniques for building energy consumption. *Renewable and Sustainable Energy Reviews*, 74:902–924, 2017. DOI 10.1016/j.rser.2017.02.085.

[71] L. Debnath and D. Bhatta. *Integral Transforms and Their Applications*. Chapman and Hall/CRC, 2016. ISBN 9780429139550. DOI 10.1201/9781420010916.

[72] M. Deistler and W. Scherrer. *Time Series Models*, volume 224. Springer International Publishing, Cham, 2022. ISBN 978-3-031-13212-4. DOI 10.1007/978-3-031-13213-1.

[73] L. A. Demidova and M. A. Stepanov. Approach to the analysis of the multidimensional time series based on the umap algorithm in the problems of the complex systems proactive maintenance. In *2020 International Conference on Information Technologies (InfoTech)*, pages 1–4. IEEE, 2020.

ISBN 978-1-7281-6914-9. DOI 10.1109/InfoTech49733.2020.9211008.

[74] F. L. Dennig, T. Polk, Z. Lin, T. Schreck, H. Pfister, and M. Behrisch. Fdive: Learning relevance models using pattern-based similarity measures. In *2019 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pages 69–80. IEEE, 2019.

[75] A. Desai, C. Freeman, Z. Wang, and I. Beaver. Timevae: A variational auto-encoder for multivariate time series generation, 2021. URL http://arxiv.org/pdf/2111.08095.

[76] H. Ding, G. Trajcevski, P. Scheuermann, X. Wang, and E. Keogh. Querying and mining of time series data: Experimental comparison of representations and distance measures. *Proc. VLDB Endow.*, 1(2):1542–1552, 2008. DOI 10.14778/1454159.1454226.

[77] P. Doukhan. *Stochastic Models for Time Series*, volume 80. Springer International Publishing, Cham, 2018. ISBN 978-3-319-76937-0. DOI 10.1007/978-3-319-76938-7.

[78] X. Du, L. Deng, and K. Qian. Current market top business scopes trend—a concurrent text and time series active learning study of nasdaq and nyse stocks from 2012 to 2017. *Applied Sciences*, 8(5), 2018. DOI 10.3390/app8050751. URL https://www.mdpi.com/2076-3417/8/5/751.

[79] J. J. Dudley and P. O. Kristensson. A review of user interface design for interactive machine learning. *ACM Trans. Interact. Intell. Syst.*, 8(2):8:1–8:37, 2018. DOI 10.1145/3185517.

[80] P. D'Urso and E. A. Maharaj. Wavelets-based clustering of multivariate time series. *Fuzzy Sets and Systems*, 193:33–61, 2012. DOI 10.1016/j.fss.2011.10.002.

[81] E. Eldele, M. Ragab, Z. Chen, M. Wu, C. K. Kwoh, and X. Li. Label-efficient time series representation learning: A review. *IEEE Transactions on Artificial Intelligence*, pages 1–16, 2024. DOI 10.1109/TAI.2024.3430236.

[82] R. F. Engle. Autoregressive conditional heteroscedasticity with estimates of the variance of united kingdom inflation. *Econometrica*, 50(4):987, 1982. DOI 10.2307/1912773.

[83] A. Ermshaus, P. Schäfer, and U. Leser. Window size selection in unsupervised time series analytics: A review and

benchmark. In T. Guyet, G. Ifrim, S. Malinowski, A. Bagnall, P. Shafer, and V. Lemaire, editors, *Advanced Analytics and Learning on Temporal Data*, volume 13812 of *Lecture Notes in Computer Science*, pages 83–101. Springer International Publishing, Cham, 2023. ISBN 978-3-031-24377-6. DOI 10.1007/978-3-031-24378-3_6.

[84] A. Ermshaus, P. Schäfer, and U. Leser. Clasp: Parameter-free time series segmentation. *Data Mining and Knowledge Discovery*, 37(3):1262–1300, 2023. DOI 10.1007/s10618-023-00923-x.

[85] P. Esling and C. Agon. Time-series data mining. *ACM Comput. Surv.*, 45(1), 2012. DOI 10.1145/2379776.2379788.

[86] P. Esling and C. Agon. Multiobjective time series matching for audio classification and retrieval. *IEEE Transactions on Audio, Speech, and Language Processing*, 21(10):2057–2072, 2013. DOI 10.1109/TASL.2013.2265086.

[87] M. Espadoto, G. Appleby, A. Suh, D. Cashman, M. Li, C. Scheidegger, E. W. Anderson, R. Chang, and A. C. Telea. Unprojection: Leveraging inverse-projections for visual analytics of high-dimensional data. *IEEE Transactions on Visualization and Computer Graphics*, 29(2):1559–1572, 2023. DOI 10.1109/TVCG.2021.3125576.

[88] M. Ester, H. P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96), Portland, Oregon, USA*, pages 226–231, 1996. URL http://www.aaai.org/Library/KDD/1996/kdd96-037.php.

[89] A. Fakhrazari and H. Vakilzadian. A survey on time series data mining. In *2017 IEEE International Conference on Electro Information Technology (EIT)*, pages 476–481, 2017. DOI 10.1109/EIT.2017.8053409.

[90] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos. Fast subsequence matching in time-series databases. In *Proceedings of the 1994 ACM SIGMOD International Conference on Management of Data*, pages 419–429, New York, NY, USA, 1994. Association for Computing Machinery. ISBN 0897916395. DOI 10.1145/191839.191925.

[91] P. Federico, M. Wagner, A. Rind, A. Amor-Amorós, S. Miksch, and W. Aigner. The role of explicit knowledge: A conceptual model of knowledge-assisted visual analytics. In *2017 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pages 92–103, 2017. DOI 10.1109/VAST.2017.8585498.

[92] D. Folgado, M. Barandas, M. Antunes, M. L. Nunes, H. Liu, Y. Hartmann, T. Schultz, and H. Gamboa. Tssearch: Time series subsequence search library. *SoftwareX*, 18:101049, 2022. DOI 10.1016/j.softx.2022.101049. URL https://www.sciencedirect.com/science/article/pii/S2352711022000425.

[93] G. Forestier, F. Petitjean, H. A. Dau, G. I. Webb, and E. Keogh. Generating synthetic time series to augment sparse datasets. In *2017 IEEE International Conference on Data Mining (ICDM)*, pages 865–870, 2017. DOI 10.1109/ICDM.2017.106.

[94] T. chung Fu. A review on time series data mining. *Engineering Applications of Artificial Intelligence*, 24(1):164–181, 2011. DOI 10.1016/j.engappai.2010.09.007. URL http://www.sciencedirect.com/science/article/pii/S0952197610001727.

[95] T. c. Fu, F. l. Chung, V. Ng, and R. Luk. Pattern discovery from stock time series using self-organizing maps. In *Workshop notes of KDD2001 workshop on temporal data mining*, volume 1, 2001.

[96] T. c. Fu, F. l. Chung, R. Luk, and C. m. Ng. Preventing meaningless stock time series pattern discovery by changing perceptually important point detection. In D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, F. Mattern, J. C. Mitchell, M. Naor, O. Nierstrasz, C. Pandu Rangan, B. Steffen, M. Sudan, D. Terzopoulos, D. Tygar, M. Y. Vardi, G. Weikum, L. Wang, and Y. Jin, editors, *Fuzzy Systems and Knowledge Discovery*, volume 3613 of *Lecture Notes in Computer Science*, pages 1171–1174. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005. ISBN 978-3-540-28312-6. DOI 10.1007/11539506_146.

[97] T. c. Fu, F. l. Chung, R. Luk, and C. m. Ng. Stock time series pattern matching: Template-based vs. rule-based approaches. *Engineering Applications of Artificial Intelligence*, 20(3):347–364, 2007. DOI 10.1016/j.engappai.2006.

07.003. URL https://www.sciencedirect.com/science/article/pii/S0952197606001278.

[98] J. Gao, W. Ji, L. Zhang, S. Shao, Y. Wang, and F. Shi. Fast piecewise polynomial fitting of time-series data for streaming computing. *IEEE Access*, 8:43764–43775, 2020. DOI 10.1109/ACCESS.2020.2976494.

[99] J. Gatheral and A. Jacquier. Arbitrage-free svi volatility surfaces. *Quantitative Finance*, 14(1):59–71, 2014. DOI 10.1080/14697688.2013.819986.

[100] A. Géron. *Hands-on Machine Learning With Scikit-Learn, Keras, and Tensorflow: Concepts, tools, and techniques to build intelligent systems*. O'Reilly and ProQuest Ebook Central, Sebastopol, CA and Ann Arbor, second edition edition, 2019. ISBN 9781492032618. URL https://ebookcentral.proquest.com/lib/kxp/detail.action?docID=5892320.

[101] T. Gevers and A. W. Smeulders. Pictoseek: Combining color and shape invariant features for image retrieval. *IEEE transactions on image processing : a publication of the IEEE Signal Processing Society*, 9(1):102–119, 2000. DOI 10.1109/83.817602.

[102] E. Ghaderpour, S. D. Pagiatakis, and Q. K. Hassan. A survey on change detection and time series analysis with applications. *Applied Sciences*, 11(13), 2021. DOI 10.3390/app11136141. URL https://www.mdpi.com/2076-3417/11/13/6141.

[103] S. Gharghabi, S. Imani, A. Bagnall, A. Darvishzadeh, and E. Keogh. Mpdist: A novel time series distance measure to allow data mining in more challenging scenarios. In *IEEE International Conference on Data Mining*, pages 965–970, Singapore, 2018. DOI 10.1109/ICDM.2018.00119.

[104] B. C. Giao and D. T. Anh. Similarity search for numerous patterns over multiple time series streams under dynamic time warping which supports data normalization. *Vietnam Journal of Computer Science*, 3(3):181–196, 2016. DOI 10.1007/s40595-016-0062-4.

[105] A. Gogolou, T. Tsandilas, T. Palpanas, and A. Bezerianos. Comparing similarity perception in time series visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):523–533, 2019. DOI 10.1109/TVCG.2018.2865077.

[106] X. Gong, S. Fong, J. H. Chan, and S. Mohammed. Nspring: the spring extension for subsequence matching of time series supporting normalization. *The Journal of Supercomputing*, 72(10):3801–3825, 2016. DOI 10.1007/s11227-015-1525-6.

[107] J. Grabocka and L. Schmidt-Thieme. Neuralwarp: Time-series similarity with warping networks. *CoRR*, abs/1812.08306, 2018. URL https://arxiv.org/abs/1812.08306.

[108] D. Graves and W. Pedrycz. Multivariate segmentation of time series with differential evolution. In *IFSA/EUSFLAT Conf*, pages 1108–1113, 2009.

[109] D. Gunopulos and G. Das. Time series similarity measures and time series indexing. In *Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data*, page 624, New York, 2001. DOI 10.1145/375663.375808.

[110] A. M. Haghighi, A. A. Kumar, and D. P. Mishev. *Higher Mathematics for Science and Engineering*. Springer Nature Singapore, Singapore, 2024. ISBN 978-981-99-5430-8. DOI 10.1007/978-981-99-5431-5.

[111] J. D. Hamilton. *Time Series Analysis*. Princeton University Press, Princeton, NJ, 2020. ISBN 0691218633. DOI 10.1515/9780691218632. URL https://www.degruyter.com/isbn/9780691218632.

[112] S. Harford, F. Karim, and H. Darabi. Generating adversarial samples on multivariate time series using variational autoencoders. *IEEE/CAA Journal of Automatica Sinica*, 8(9): 1523–1538, 2021. DOI 10.1109/JAS.2021.1004108.

[113] H. Hassani and D. Thomakos. A review on singular spectrum analysis for economic and financial time series. *Statistics and Its Interface*, 3(3):377–397, 2010. DOI 10.4310/SII.2010.v3.n3.a11.

[114] H. He, Q. Zhang, K. Yi, K. Shi, Z. Niu, and L. Cao. Distributional drift adaptation with temporal conditional variational autoencoder for multivariate time series forecasting. *IEEE Transactions on Neural Networks and Learning Systems*, PP, 2024. DOI 10.1109/TNNLS.2024.3384842.

[115] Z. He, J. Chen, and M. Gao. Feature time series clustering for lithium battery based on som neural network. In *2018*

13th IEEE Conference on Industrial Electronics and Applications (ICIEA), pages 358–363. IEEE, 2018. ISBN 978-1-5386-3758-6. DOI 10.1109/ICIEA.2018.8397742.

[116] J. Heer, N. Kong, and M. Agrawala. Sizing the horizon: The effects of chart size and layering on the graphical perception of time series visualizations. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1303–1312, New York, NY, USA, 2009. Association for Computing Machinery. ISBN 9781605582467. DOI 10.1145/1518701.1518897.

[117] G. E. Hinton and S. Roweis. Stochastic neighbor embedding. *Advances in Neural Information Processing Systems*, 15, 2002.

[118] T. N. Hoang, B. K. H. Low, P. Jaillet, and M. Kankanhalli. Nonmyopic ϵ-bayes-optimal active learning of gaussian processes. In E. P. Xing and T. Jebara, editors, *Proceedings of the 31st International Conference on Machine Learning*, volume 32 number 2 of *Proceedings of Machine Learning Research*, pages 739–747, Bejing, China, 22–24 Jun 2014. PMLR. URL https://proceedings.mlr.press/v32/hoang14.html.

[119] H. Hochheiser and B. Shneiderman. Dynamic query tools for time series data sets: Timebox widgets for interactive exploration. *Information Visualization*, 3(1):1–18, 2004. DOI 10.1057/palgrave.ivs.9500061.

[120] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997. DOI 10.1162/neco.1997.9.8.1735.

[121] C. Holder, M. Middlehurst, and A. Bagnall. A review and evaluation of elastic distance functions for time series clustering. *Knowledge and Information Systems*, 66(2):765–809, 2024. DOI 10.1007/s10115-023-01952-0.

[122] L. Hou, X. Jin, and Z. Zhao. Time series similarity measure via siamese convolutional neural network. In *2019 12th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*, pages 1–6, 2019. DOI 10.1109/CISP-BMEI48845.2019.8966048.

[123] T. Huamin, D. Qiuqun, and X. Shanzhu. Reconstruction of time series with missing value using 2d representation-based denoising autoencoder. *Journal of Systems Engineering*

and Electronics, 31(6):1087–1096, 2020. DOI 10.23919/JSEE.2020.000081.

[124] N. E. Huang and N. O. Attoh-Okine. *The Hilbert-Huang Transform in Engineering*. CRC Press, 2005. ISBN 9780429114311. DOI 10.1201/9781420027532.

[125] N. E. Huang and Z. Wu. A review on hilbert–huang transform: Method and its applications to geophysical studies. *Reviews of Geophysics*, 46(2), 2008. DOI 10.1029/2007RG000228.

[126] N. E. Huang, Z. Shen, S. R. Long, M. C. Wu, H. H. Shih, Q. Zheng, N. C. Yen, C. C. Tung, and H. H. Liu. The empirical mode decomposition and the hilbert spectrum for nonlinear and non-stationary time series analysis. *Proceedings of the Royal Society of London. Series A: mathematical, physical and engineering sciences*, 454(1971):903–995, 1998.

[127] Q. Huang, J. Feng, Y. Zhang, Q. Fang, and W. Ng. Query-aware locality-sensitive hashing for approximate nearest neighbor search. *Proceedings of the VLDB Endowment*, 9(1): 1–12, 2015.

[128] M. Hüsken and P. Stagge. Recurrent neural networks for time series classification. *Neurocomputing*, 50:223–235, 2003. DOI 10.1016/S0925-2312(01)00706-8.

[129] S. Imani, S. Alaee, and E. Keogh. Qute: Query by text search for time series data. In K. Arai, S. Kapoor, and R. Bhatia, editors, *Proceedings of the Future Technologies Conference (FTC) 2020, Volume 2*, pages 412–427, Cham, 2021. Springer International Publishing. ISBN 978-3-030-63089-8.

[130] P. Indyk and R. Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, page 604–613, New York, NY, USA, 1998. Association for Computing Machinery. ISBN 0897919629. DOI 10.1145/276698.276876.

[131] Y. Ishikawa, R. Subramanya, and C. Faloutsos. Mindreader: Querying databases through multiple examples. In *Proceedings of 24rd International Conference on Very Large Data Bases, August 24-27, 1998, New York City, New York, USA*, pages 218–227, 1998. URL http://www.vldb.org/conf/1998/p218.pdf.

[132] A. Kane and N. Shiri. Multivariate time series representation and similarity search using pca. In P. Perner, editor, *Advances in Data Mining. Applications and Theoretical Aspects*, volume 10357 of *Lecture Notes in Computer Science*, pages 122–136. Springer International Publishing and Imprint: Springer, Cham, 2017. ISBN 978-3-319-62700-7. DOI 10.1007/978-3-319-62701-4_10.

[133] R. Karimi, C. Freudenthaler, A. Nanopoulos, and L. Schmidt-Thieme. Non-myopic active learning for recommender systems based on matrix factorization. In *2011 IEEE International Conference on Information Reuse & Integration*, pages 299–303, 2011. DOI 10.1109/IRI.2011.6009563.

[134] F. Karl Pearson. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901. DOI 10.1080/14786440109462720.

[135] K. Kashino, G. Smith, and H. Murase. Time-series active search for quick retrieval of audio and video. In *1999 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings. ICASSP99 (Cat. No.99CH36258)*, pages 2993–2996 vol.6. IEEE, 1999. ISBN 0-7803-5041-3. DOI 10.1109/ICASSP.1999.757470.

[136] E. Keogh. Fast similarity search in the presence of longitudinal scaling in time series databases. In *Proceedings Ninth IEEE International Conference on Tools with Artificial Intelligence*, pages 578–584, 1997. DOI 10.1109/TAI.1997.632306.

[137] E. Keogh. A fast and robust method for pattern matching in time series databases. In *Proceedings of Western Users of SAS Software*, Universal City, 1997.

[138] E. Keogh. Indexing and mining time series data. In S. Shekhar and H. Xiong, editors, *Encyclopedia of GIS*, pages 493–497. Springer US, Boston, MA, 2008. ISBN 978-0-387-35973-1. DOI 10.1007/978-0-387-35973-1_598.

[139] E. Keogh and P. Smyth. A probabilistic approach to fast pattern matching in time series databases. In *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining*, pages 24–30. AAAI Press, 1997.

[140] E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra. Dimensionality reduction for fast similarity search in large

time series databases. *Knowledge and Information Systems*, 3 (3):263–286, 2001. DOI 10.1007/PL00011669.

[141] E. Keogh, L. Wei, X. Xi, S. H. Lee, and M. Vlachos. Lb_keogh supports exact indexing of shapes under rotation invariance with arbitrary representations and distance measures. In *Proceedings of the 32nd international conference on Very large data bases*, pages 882–893, 2006.

[142] E. J. Keogh and M. J. Pazzani. An enhanced representation of rime series which allows fast and accurate classification, clustering and relevance feedback. In *The 4th International Conference on Knowledge Discovery and Data Mining (KDD98)*, volume 98, pages 239–243, 1998.

[143] E. J. Keogh and M. J. Pazzani. A simple dimensionality reduction technique for fast similarity search in large time series databases. In T. Terano, H. Liu, and A. L. P. Chen, editors, *Knowledge Discovery and Data Mining. Current Issues and New Applications*, pages 122–133, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg. ISBN 978-3-540-45571-4.

[144] S. Ketu and P. K. Mishra. Hybrid classification model for eye state detection using electroencephalogram signals. *Cognitive Neurodynamics*, 2021. DOI 10.1007/s11571-021-09678-x.

[145] A. Khoshrou, A. B. Dorsman, and E. J. Pauwels. Svd-based visualisation and approximation for time series data in smart energy systems. In *2017 IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT-Europe)*, pages 1–6. IEEE, 2017. ISBN 978-1-5386-1953-7. DOI 10.1109/ISGTEurope.2017.8260303.

[146] D. P. Kingma and M. Welling. Auto-encoding variational bayes, 2013. URL http://arxiv.org/pdf/1312.6114.

[147] P. Kinyua and N. Jouandeau. Sample-label view transfer active learning for time series classification. In I. Farkaš, P. Masulli, S. Otte, and S. Wermter, editors, *Artificial Neural Networks and Machine Learning – ICANN 2021*, pages 600–611, Cham, 2021. Springer International Publishing. ISBN 978-3-030-86383-8.

[148] H. Koga, T. Ishibashi, and T. Watanabe. Fast agglomerative hierarchical clustering algorithm using locality-sensitive hashing. *Knowledge and Information Systems*, 12(1):25–53, 2007. DOI 10.1007/s10115-006-0027-5.

[149] J. R. Koza, F. H. Bennett, D. Andre, and M. A. Keane. Automated design of both the topology and sizing of analog electrical circuits using genetic programming. In J. S. Gero and F. Sudweeks, editors, *Artificial Intelligence in Design*, pages 151–170. Springer Netherlands, Dordrecht, 1996. ISBN 978-94-010-6610-5. DOI 10.1007/978-94-009-0279-4_9.

[150] M. A. Kramer. Nonlinear principal component analysis using autoassociative neural networks. *AIChE Journal*, 37(2): 233–243, 1991. DOI 10.1002/aic.690370209.

[151] A. Krause and C. Guestrin. Nonmyopic active learning of gaussian processes: An exploration-exploitation approach. In *Proceedings of the 24th International Conference on Machine Learning*, page 449–456, New York, NY, USA, 2007. Association for Computing Machinery. ISBN 9781595937933. DOI 10.1145/1273496.1273553.

[152] R. Kumar, P. Kumar, and Y. Kumar. Time series data prediction using iot and machine learning technique. *Procedia Computer Science*, 167:373–381, 2020. DOI 10.1016/j.procs.2020.03.240.

[153] E. Laftchiev and Y. Liu. Finding multidimensional patterns in multidimensional time series. In *KDD workshop on MiLeTS 2018, London*, 2018.

[154] M. Längkvist, L. Karlsson, and A. Loutfi. A review of unsupervised feature learning and deep learning for time-series modeling. *Pattern Recognition Letters*, 42:11–24, 2014. DOI 10.1016/j.patrec.2014.01.008.

[155] D. J. L. Lee, J. Lee, T. Siddiqui, J. Kim, K. Karahalios, and A. Parameswaran. 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. DOI 10.1109/TVCG.2019.2934666.

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

[157] G. Li and J. J. Jung. Deep learning for anomaly detection in multivariate time series: Approaches, applications, and challenges. *Information Fusion*, 91:93–102, 2023. DOI 10.1016/j.inffus.2022.10.008.

[158] H. Li. Multivariate time series clustering based on common principal component analysis. *Neurocomputing*, 349: 239–247, 2019. DOI 10.1016/j.neucom.2019.03.060.

[159] H. Li, S. Yu, and J. Principe. Causal recurrent variational autoencoder for medical time series generation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(7):8562–8570, 2023. DOI 10.1609/aaai.v37i7.26031.

[160] J. Li and N. M. Allinson. Relevance feedback in content-based image retrieval: A survey. In M. Bianchini, M. Maggini, and L. C. Jain, editors, *Handbook on Neural Information Processing*, volume 49 of *Intelligent Systems Reference Library*, pages 433–469. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013. ISBN 978-3-642-36656-7. DOI 10.1007/978-3-642-36657-4_13.

[161] L. Li, J. Yan, H. Wang, and Y. Jin. Anomaly detection of time series with smoothness-inducing sequential variational auto-encoder. *IEEE Transactions on Neural Networks and Learning Systems*, 32(3):1177–1191, 2021. DOI 10.1109/TNNLS.2020.2980749.

[162] X. Li, H. Cui, T. Jiang, Y. Xu, H. Jia, and F. Li. Multichannel continuous wavelet transform approach to estimate electromechanical oscillation modes, mode shapes and coherent groups from synchrophasors in bulk power grids. *International Journal of Electrical Power & Energy Systems*, 96: 222–237, 2018. DOI 10.1016/j.ijepes.2017.09.043.

[163] T. W. Liao. Clustering of time series data - a survey. *Pattern Recognition*, 38(11):1857–1874, 2005. DOI 10.1016/j.patcog.2005.01.025. URL https://www.sciencedirect.com/science/article/pii/S0031320305001305.

[164] J. Lin, E. Keogh, S. Lonardi, and B. Chiu. A Symbolic Representation of Time Series, with Implications for Streaming Algorithms. In *Proceedings of 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, pages 2–11, New York, NY, USA, 2003. Assoc. for Comp. Mach. DOI 10.1145/882082.882086.

[165] J. Lin, E. Keogh, L. Wei, and S. Lonardi. Experiencing sax: a novel symbolic representation of time series. *Data Mining and knowledge discovery*, 15(2):107–144, 2007.

[166] J. Lin, S. Williamson, K. Borne, and D. DeBarr. Pattern recognition in time series. *Advances in Machine Learning and Data Mining for Astronomy*, 1(617-645):3, 2012.

[167] J. Lines and A. Bagnall. Time series classification with ensembles of elasticdistance measures. *Data Mining and Knowledge Discovery*, 29(3):565–592, 2015. DOI 10.1007/s10618-014-0361-2.

[168] X. Liu, Z. Lin, and H. Wang. Novel online methods for time series segmentation. *IEEE Transactions on Knowledge and Data Engineering*, 20(12):1616–1626, 2008. DOI 10.1109/TKDE.2008.29.

[169] S. Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982. DOI 10.1109/TIT.1982.1056489.

[170] A. M. Lopes and J. A. Tenreiro Machado. Analysis of temperature time-series: Embedding dynamics into the mds method. *Communications in Nonlinear Science and Numerical Simulation*, 19(4):851–871, 2014. DOI 10.1016/j.cnsns.2013.08.031.

[171] M. Lovrić, M. Milanović, and M. Stamenković. Algoritmic methods for segmentation of time series: An overview. *Journal of Contemporary Economic and Business Issues*, 1(1):31–53, 2014. ISSN 1857-9108. URL https://hdl.handle.net/10419/147468.

[172] M. Lu and X. Xu. Trnn: An efficient time-series recurrent neural network for stock price prediction. *Information Sciences*, 657:119951, 2024. DOI 10.1016/j.ins.2023.119951.

[173] Y. Lu, C. Hu, X. Zhu, H. Zhang, and Q. Yang. A unified framework for semantics and feature based relevance feedback in image retrieval systems. In S. Ghandeharizadeh, S. F. Chang, S. Fischer, J. A. Konstan, and K. Nahrstedt, editors, *Proceedings of the eighth ACM international conference on Multimedia*, pages 31–37, New York, NY, USA, 2000. ACM. ISBN 1581131984. DOI 10.1145/354384.354403.

[174] Y. Lu, R. Wu, A. Mueen, M. A. Zuluaga, and E. Keogh. Matrix profile xxiv: Scaling time series anomaly detection to trillions of datapoints and ultra-fast arriving data streams. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 1173–1182,

New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450393850. DOI 10.1145/3534678.3539271.

[175] Y. Lu, R. Wu, A. Mueen, M. A. Zuluaga, and E. Keogh. Damp: Accurate time series anomaly detection on trillions of datapoints and ultra-fast arriving data streams. *Data Mining and Knowledge Discovery*, 37(2):627–669, 2023. DOI 10.1007/s10618-022-00911-7.

[176] C. Luo and A. Shrivastava. Ssh (sketch shingle & hash) for indexing massive-scale time series. In Oren Anava, Azadeh Khaleghi, Marco Cuturi, Vitaly Kuznetsov, and Alexander Rakhlin, editors, *Proceedings of the Time Series Workshop at NIPS 2016*, volume 55 of *Proceedings of Machine Learning Research*, pages 38–58, Barcelona, Spain, 2017. PMLR. URL http://proceedings.mlr.press/v55/luo16.html.

[177] Q. Ma, Z. Liu, Z. Zheng, Z. Huang, S. Zhu, Z. Yu, and J. T. Kwok. A survey on time-series pre-trained models. *IEEE Transactions on Knowledge and Data Engineering*, pages 1–20, 2024. DOI 10.1109/TKDE.2024.3475809.

[178] E. A. Maharaj and A. M. Alonso. Discriminant analysis of multivariate time series: Application to diagnosis based on ecg signals. *Computational Statistics & Data Analysis*, 70:67–87, 2014. DOI 10.1016/j.csda.2013.09.006.

[179] S. Maheshwari and A. Kumar. Empirical mode decomposition: Theory & applications. *International Journal of Electronic and Electrical Engineering*, 7(8):873–878, 2014.

[180] P. Mannfolk, R. Wirestam, M. Nilsson, F. Ståhlberg, and J. Olsrud. Dimensionality reduction of fmri time series data using locally linear embedding. *Magma (New York, N.Y.)*, 23 (5-6):327–338, 2010. DOI 10.1007/s10334-010-0204-0.

[181] M. Mannino and A. Abouzied. Qetch: Time series querying with expressive sketches. In *Proceedings of the 44th International Conference on Management of Data*, pages 1741–1744, New York, 2018. DOI 10.1145/3183713.3193547.

[182] M. Mannino and A. Abouzied. Expressive time series querying with hand-drawn scale-free sketches. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, pages 1–13, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450356206. DOI 10.1145/3173574.3173962.

[183] M. Mansour, A. Gamal, A. I. Ahmed, L. A. Said, A. Elbaz, N. Herencsar, and A. Soltan. Internet of things: A comprehensive overview on protocols, architectures, technologies, simulation tools, and future directions. *Energies*, 16(8):3465, 2023. DOI 10.3390/en16083465.

[184] R. P. Masini, M. C. Medeiros, and E. F. Mendes. Machine learning advances for time series forecasting. *Journal of Economic Surveys*, 37(1):76–111, 2023. DOI 10.1111/joes.12429.

[185] L. McInnes, J. Healy, and J. Melville. Umap: Uniform manifold approximation and projection for dimension reduction, 2018.

[186] R. Mercer and E. Keogh. Matrix profile xxv: Introducing novelets: A primitive that allows online detection of emerging behaviors in time series. In *2022 IEEE International Conference on Data Mining (ICDM)*, pages 338–347, 2022. DOI 10.1109/ICDM54844.2022.00044.

[187] R. Mercer, S. Alaee, A. Abdoli, S. Singh, A. Murillo, and E. Keogh. Matrix profile xxiii: Contrast profile: A novel time series primitive that allows real world classification. In *2021 IEEE International Conference on Data Mining (ICDM)*, pages 1240–1245, 2021. DOI 10.1109/ICDM51629.2021.00151.

[188] Y. C. C. Michael, Z. Yan, U. Liudmila, B. Nurjahan, D. Yifei, D. H. Anh, S. D. Furtado, M. Abdullah, and K. Eamonn. Matrix profile i: All pairs similarity joins for time series: A unifying view that includes motifs, discords and shapelets. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*, pages 1317–1322, 2016. DOI 10.1109/ICDM.2016.0179.

[189] M. Middlehurst, P. Schäfer, and A. Bagnall. Bake off redux: A review and experimental evaluation of recent time series classification algorithms. *Data Mining and Knowledge Discovery*, 2024. DOI 10.1007/s10618-024-01022-1.

[190] N. Mohammadi Foumani, L. Miller, C. W. Tan, G. I. Webb, G. Forestier, and M. Salehi. Deep learning for time series classification and extrinsic regression: A current survey. *ACM Comput. Surv.*, 56(9):1–45, 2024. DOI 10.1145/3649448.

[191] R. Moraffah, P. Sheth, M. Karami, A. Bhattacharya, Q. Wang, A. Tahir, A. Raglin, and H. Liu. Causal inference

for time series analysis: Problems, methods and evaluation. *Knowledge and Information Systems*, 63(12):3041–3085, 2021. DOI 10.1007/s10115-021-01621-0.

[192] K. P. Murphy. *Machine Learning: A Probabilistic Perspective*. Adaptive computation and machine learning series. MIT Press, Cambridge, Mass., 4. print. (fixed many typos) edition, 2013. ISBN 0262018020.

[193] T. Negi and V. Bansal. Time series: Similarity search and its applications. In *Proceedings of the International Conference on Systemics, Cybernetics and Informatics: ICSCI-04, Hyderabad, India*, pages 528–533, 2005.

[194] N. Nguyen and B. Quanz. Temporal latent auto-encoder: A method for probabilistic multivariate time series forecasting. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(10):9117–9125, 2021. DOI 10.1609/aaai.v35i10.17101.

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

[196] U. Y. Ogras and H. Ferhatosmanoglu. Online summarization of dynamic time series data. *The VLDB Journal*, 15(1): 84–98, 2006. DOI 10.1007/s00778-004-0149-x.

[197] M. Ouzzani, H. Hammady, Z. Fedorowicz, and A. Elmagarmid. Rayyan-a web and mobile app for systematic reviews. *Systematic reviews*, 5(1):210, 2016. DOI 10.1186/s13643-016-0384-4.

[198] A. Pandey, H. de Meulemeester, B. de Moor, and J. A. Suykens. Multi-view kernel pca for time series forecasting. *Neurocomputing*, 554:126639, 2023. DOI 10.1016/j.neucom.2023.126639.

[199] L. Pang, J. Xu, Q. Ai, Y. Lan, X. Cheng, and J. Wen. Setrank: Learning a permutation-invariant ranking model for information retrieval. In J. Huang, Y. Chang, X. Cheng, J. Kamps, V. Murdock, J. R. Wen, and Y. Liu, editors, *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 499–508, New York, NY, USA, 2020. ACM. ISBN 9781450380164. DOI 10.1145/3397271.3401104.

[200] B. Pavlyshenko. Machine-learning models for sales time series forecasting. *Data*, 4(1):15, 2019. DOI 10.3390/data4010015.

[201] W. Pei, D. M. J. Tax, and L. van der Maaten. Modeling time series similarity with siamese recurrent networks. *arXiv preprint arXiv:1603.04713*, 2016.

[202] B. Peng, P. Fatourou, and T. Palpanas. Sing: Sequence indexing using gpus. In *IEEE 37th International Conference on Data Engineering*, pages 1883–1888, Chania, 2021. DOI 10.1109/ICDE51399.2021.00171.

[203] J. Peng. Multi-class relevance feedback content-based image retrieval. *Computer Vision and Image Understanding*, 90 (1):42–67, 2003. DOI 10.1016/S1077-3142(03)00013-4.

[204] F. Petitjean, G. Forestier, G. I. Webb, A. E. Nicholson, Y. Chen, and E. Keogh. Dynamic time warping averaging of time series allows faster and more accurate classification. In *2014 IEEE International Conference on Data Mining*, pages 470–479, 2014. DOI 10.1109/ICDM.2014.27.

[205] F. Petitjean, A. Ketterlin, and P. Gançarski. A global averaging method for dynamic time warping, with applications to clustering. *Pattern Recognit.*, 44(3):678–693, 2011. DOI 10.1016/j.patcog.2010.09.013.

[206] I. Pitas and A. Karasaridis. Multichannel transforms for signal/image processing. *IEEE Transactions on Image Processing*, 5(10):1402–1413, 1996. DOI 10.1109/83.536889.

[207] I. Popivanov and R. J. Miller. Similarity search over time-series data using wavelets. In *Proceedings 18th International Conference on Data Engineering*, pages 212–221, 2002. DOI 10.1109/ICDE.2002.994711.

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

[209] P. Przybyła, A. J. Brockmeier, G. Kontonatsios, M. A. Le Pogam, J. McNaught, E. von Elm, K. Nolan, and S. Ananiadou. Prioritising references for systematic reviews with robotanalyst: A user study. *Research synthesis methods*, 9(3): 470–488, 2018. DOI 10.1002/jrsm.1311.

[210] T. Rakthanmanon, B. Campana, A. Mueen, G. Batista, B. Westover, Q. Zhu, J. Zakaria, and E. Keogh. Searching and mining trillions of time series subsequences under dynamic time warping. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 262–270, New York, NY, USA, 2012. Association for Computing Machinery. ISBN 9781450314626. DOI 10.1145/2339530.2339576.

[211] T. Rakthanmanon, B. Campana, A. Mueen, G. Batista, B. Westover, Q. Zhu, J. Zakaria, and E. Keogh. Addressing big data time series: Mining trillions of time series subsequences under dynamic time warping. *ACM Trans. Knowl. Discov. Data*, 7(3), 2013. DOI 10.1145/2500489.

[212] E. Ramanujam and S. Padmavathi. Comprehensive review on time series motif discovery using evolutionary techniques. *International Journal of Advanced Intelligence Paradigms*, 23(1/2):155, 2022. DOI 10.1504/IJAIP.2022.125239.

[213] X. Ran, Y. Xi, Y. Lu, X. Wang, and Z. Lu. Comprehensive survey on hierarchical clustering algorithms and the recent developments. *Artificial Intelligence Review*, 56(8):8219–8264, 2023. DOI 10.1007/s10462-022-10366-3.

[214] K. D. Rao and M. Swamy. *Digital Signal Processing*. Springer Singapore, Singapore, 2018. ISBN 978-981-10-8080-7. DOI 10.1007/978-981-10-8081-4.

[215] A. Raveh and C. S. Tapiero. Finding common seasonal patterns among time series: An mds approach. *Journal of Econometrics*, 12(3):353–363, 1980. DOI 10.1016/0304-4076(80)90061-5.

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

[217] M. Rhif, A. Ben Abbes, I. Farah, B. Martínez, and Y. Sang. Wavelet transform application for/in non-stationary time-series analysis: A review. *Applied Sciences*, 9(7):1345, 2019. DOI 10.3390/app9071345.

[218] B. Roach. Eeg schizophrenia, 2019. URL `https://www.kaggle.com/datasets/broach/button-tone-sz`.

[219] J. Rodrigues, D. Folgado, D. Belo, and H. Gamboa. Ssts: A syntactic tool for pattern search on time series. *Information Processing & Management*, 56(1):61–76, 2019. DOI 10.1016/j.ipm.2018.09.001. URL http://www.sciencedirect.com/science/article/pii/S0306457318302577.

[220] J. Ronkainen and A. Iivari. Designing a data management pipeline for pervasive sensor communication systems. *Procedia Computer Science*, 56:183–188, 2015. DOI 10.1016/j.procs.2015.07.193.

[221] O. Rösler. EEG eye state, June 2013. URL https://doi.org/10.24432/C57G7J. Accessed on 2024-02-01.

[222] O. Rösler and D. Suendermann. A First Step Towards Eye State Prediction Using EEG. *Proc. of the AIHLS*, 1:1–4, 2013.

[223] Y. Rui and T. Huang. Optimizing learning in image retrieval. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000 (Cat. No.PR00662)*, pages 236–243. IEEE Comput. Soc, 2000. ISBN 0-7695-0662-3. DOI 10.1109/CVPR.2000.855825.

[224] A. L. Rukhin. Analysis of time series structure ssa and related techniques. *Technometrics*, 44(3):290, 2002. DOI 10.1198/004017002320256477.

[225] N. Ruta, N. Sawada, K. McKeough, M. Behrisch, and J. Beyer. Sax navigator: Time series exploration through hierarchical clustering. In *2019 IEEE Visualization Conference (VIS)*, pages 236–240, 2019. DOI 10.1109/VISUAL.2019.8933618.

[226] I. A. Ruthven and M. Lalmas. A survey on the use of relevance feedback for information access systems. *The Knowledge Engineering Review*, 18(2):95–145, 2003. DOI 10.1017/S0269888903000638.

[227] K. Ryall, N. Lesh, T. Lanning, D. Leigh, H. Miyashita, and S. Makino. Querylines: Approximate query for visual browsing. In *CHI '05 Extended Abstracts on Human Factors in Computing Systems*, pages 1765–1768, New York, NY, USA, 2005. Association for Computing Machinery. ISBN 1595930027. DOI 10.1145/1056808.1057017.

[228] O. M. Saad and Y. Chen. Deep denoising autoencoder for seismic random noise attenuation. *GEOPHYSICS*, 85(4): V367–V376, 2020. DOI 10.1190/geo2019-0468.1.

[229] Y. Sakurai, C. Faloutsos, and M. Yamamuro. Stream monitoring under the time warping distance. In *2007 IEEE 23rd International Conference on Data Engineering*, pages 1046–1055, 2007. ISBN 2375-026X. DOI 10.1109/ICDE.2007.368963.

[230] R. Salles, K. Belloze, F. Porto, P. H. Gonzalez, and E. Ogasawara. Nonstationary time series transformation methods: An experimental review. *Knowledge-Based Systems*, 164:274–291, 2019. DOI 10.1016/j.knosys.2018.10.041.

[231] K. K. R. Samal, K. S. Babu, S. K. Das, and A. Acharaya. Time series based air pollution forecasting using sarima and prophet model. In *Proceedings of the 2019 International Conference on Information Technology and Computer Communications*, pages 80–85, New York, NY, USA, 2019. ACM. ISBN 9781450372282. DOI 10.1145/3355402.3355417.

[232] T. Schlegl, S. Schlegl, D. Tomaselli, N. West, and J. Deuse. Adaptive similarity search for the retrieval of rare events from large time series databases. *Advanced Engineering Informatics*, 52:101629, 2022. DOI 10.1016/j.aei.2022.101629. URL https://www.sciencedirect.com/science/article/pii/S1474034622000945.

[233] J. Schleich, M. Klobasa, M. Brunner, S. Gölz, K. Götz, and G. Sunderer. Smart metering in germany–results of providing feedback information in a field trial. *Proceedings of the ECEEE 2011 Summer Study*, pages 1667–1674, 2011.

[234] S. Schmidl, P. Wenig, and T. Papenbrock. Anomaly detection in time series: A comprehensive evaluation. *Proc. VLDB Endow.*, 15(9):1779–1797, 2022. DOI 10.14778/3538598.3538602.

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

[236] L. Shao, M. Behrisch, T. Schreck, T. von Landesberger, M. Scherer, S. Bremm, and D. A. Keim. Guided sketching for visual search and exploration in large scatter plot spaces. In M. Pohl and J. C. Roberts, editors, *5th International EuroVis Workshop on Visual Analytics, EuroVA@EuroVis*

2014, Swansea, UK, June 9-10, 2014. Eurographics Association, 2014. DOI 10.2312/eurova.20141140.

[237] M. Shokoohi-Yekta, J. Wang, and E. Keogh. On the non-trivial generalization of dynamic time warping to the multi-dimensional case. In *Proceedings of the 2015 SIAM International Conference on Data Mining*, pages 289–297, 2015. DOI 10.1137/1.9781611974010.33.

[238] R. H. Shumway and D. S. Stoffer. *Time Series Analysis and Its Applications*. Springer International Publishing, Cham, 2017. ISBN 978-3-319-52451-1. DOI 10.1007/978-3-319-52452-8.

[239] T. Siddiqui, P. Luh, Z. Wang, K. Karahalios, and A. Parameswaran. Shapesearch: A flexible and efficient system for shape-based exploration of trendlines. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, pages 51–65, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450367356. DOI 10.1145/3318464.3389722.

[240] S. Simon, S. Mittelstädt, D. A. Keim, and M. Sedlmair. Bridging the gap of domain and visualization experts with a liaison. In *Proceeding of Eurographics Conference on Visualization*, pages 127–131, Cagliari, 2015. DOI 10.2312/eurovisshort.20151137.

[241] S. Singh and A. Yassine. Big data mining of energy time series for behavioral analytics and energy consumption forecasting. *Energies*, 11(2):452, 2018. DOI 10.3390/en11020452.

[242] U. B. de Souza, J. P. L. Escola, and L. C. Da Brito. A survey on hilbert-huang transform: Evolution, challenges and solutions. *Digital Signal Processing*, 120:103292, 2022. DOI 10.1016/j.dsp.2021.103292.

[243] P. K. V. Sowdaboina, S. Chakraborti, and S. Sripada. Learning to summarize time series data. In A. Gelbukh, editor, *Computational Linguistics and Intelligent Text Processing*, pages 515–528, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg. ISBN 978-3-642-54906-9.

[244] A. Stefan, V. Athitsos, and G. Das. The move-split-merge metric for time series. *IEEE Transactions on Knowledge and Data Engineering*, 25(6):1425–1438, 2013.

[245] K. Sternickel. Automatic pattern recognition in ecg time series. *Computer Methods and Programs*

in Biomedicine, 68(2):109–115, 2002. DOI 10.1016/S0169-2607(01)00168-7. URL http://www.sciencedirect.com/science/article/pii/S0169260701001687.

[246] S. J. Taylor and B. Letham. Forecasting at scale, 2017.

[247] A. Tharwat and W. Schenck. A survey on active learning: State-of-the-art, practical challenges and research directions. *Mathematics*, 11(4):820, 2023. DOI 10.3390/math11040820.

[248] M. Thill, W. Konen, H. Wang, and T. Bäck. Temporal convolutional autoencoder for unsupervised anomaly detection in time series. *Applied Soft Computing*, 112:107751, 2021. DOI 10.1016/j.asoc.2021.107751.

[249] S. Torkamani and V. Lohweg. Survey on time series motif discovery. *WIREs Data Mining and Knowledge Discovery*, 7 (2):e1199, 2017. DOI 10.1002/widm.1199.

[250] P. Trirat, Y. Shin, J. Kang, Y. Nam, J. Na, M. Bae, J. Kim, B. Kim, and J. G. Lee. Universal time-series representation learning: A survey, 2024.

[251] P. E. Tsinaslanidis and D. Kugiumtzis. A prediction scheme using perceptually important points and dynamic time warping. *Expert Systems with Applications*, 41(15):6848–6860, 2014. DOI 10.1016/j.eswa.2014.04.028.

[252] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need, 2017.

[253] O. Vechtomova. Query expansion for information retrieval. In L. LIU and M. T. Özsu, editors, *Encyclopedia of Database Systems*, pages 2254–2257. Springer US, Boston, MA, 2009. ISBN 978-0-387-39940-9. DOI 10.1007/978-0-387-39940-9_947.

[254] J. F. Vera and J. M. Angulo. An mds-based unifying approach to time series k-means clustering: Application in the dynamic time warping framework. *Stochastic Environmental Research and Risk Assessment*, 37(12):4555–4566, 2023. DOI 10.1007/s00477-023-02470-9.

[255] B. C. Wallace, K. Small, C. E. Brodley, J. Lau, and T. A. Trikalinos. Deploying an interactive machine learning system in an evidence-based practice center. In G. Luo, J. Liu,

and C. C. Yang, editors, *Proceedings of the 2nd ACM SIGHIT International Health Informatics Symposium*, pages 819–824, New York, NY, USA, 2012. ACM. ISBN 9781450307819. DOI 10.1145/2110363.2110464.

[256] J. Y. Wang and Z. Zhu. Image retrieval system based on multi-feature fusion and relevance feedback. In *2010 International Conference on Machine Learning and Cybernetics*, pages 2053–2058. IEEE, 2010. ISBN 978-1-4244-6526-2. DOI 10.1109/ICMLC.2010.5580505.

[257] J. Wang and S. Tang. Time series classification based on arima and adaboost. *MATEC Web of Conferences*, 309:03024, 2020. DOI 10.1051/matecconf/202030903024.

[258] L. Wang, F. Lu, M. Cui, and Y. Bao. Survey of methods for time series symbolic aggregate approximation. In X. Cheng, W. Jing, X. Song, and Z. Lu, editors, *Data Science*, volume 1058 of *Communications in Computer and Information Science*, pages 645–657. Springer Singapore, Singapore, 2019. ISBN 978-981-15-0117-3. DOI 10.1007/978-981-15-0118-0_50.

[259] M. Wattenberg. Sketching a graph to query a time-series database. In *CHI '01 Extended Abstracts on Human Factors in Computing Systems*, pages 381–382, New York, NY, USA, 2001. Association for Computing Machinery. ISBN 1581133405. DOI 10.1145/634067.634292.

[260] Q. Wen, T. Zhou, C. Zhang, W. Chen, Z. Ma, J. Yan, and L. Sun. Transformers in time series: A survey, 2022.

[261] X. Weng and S. Qin. Classification of multivariate time series using supervised isomap. In *2012 Third Global Congress on Intelligent Systems*, pages 136–139. IEEE, 2012. ISBN 978-1-4673-3072-5. DOI 10.1109/GCIS.2012.31.

[262] X. Weng and J. Shen. Classification of multivariate time series using two-dimensional singular value decomposition. *Knowledge-Based Systems*, 21(7):535–539, 2008. DOI 10.1016/j.knosys.2008.03.014.

[263] C. K. I. Williams and C. E. Rasmussen. *Gaussian Processes for Machine Learning*, volume 2. MIT press Cambridge, MA, 2006.

[264] S. J. Wilson. Data representation for time series data mining: Time domain approaches. *WIREs Computational Statistics*, 9(1), 2017. DOI 10.1002/wics.1392.

[265] K. Y. Wong and F. l. Chung. Visualizing time series data with temporal matching based t-sne. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2019. ISBN 978-1-7281-1985-4. DOI 10.1109/IJCNN.2019.8851847.

[266] L. Wu, J. T. Zhou, H. Zhang, S. R. Wang, T. Ma, H. Yan, and S. H. Li. Time series analysis and gated recurrent neural network model for predicting landslide displacements. *Georisk: Assessment and Management of Risk for Engineered Systems and Geohazards*, 18(1):172–185, 2024. DOI 10.1080/17499518.2022.2138918.

[267] L. Yan, Y. Liu, and Y. Liu. Interval feature transformation for time series classification using perceptually important points. *Applied Sciences*, 10(16):5428, 2020. DOI 10.3390/app10165428.

[268] K. Yang and C. Shahabi. A pca-based similarity measure for multivariate time series. In S. C. Chen and M. L. Shyu, editors, *Proceedings of the 2nd ACM international workshop on Multimedia databases*, pages 65–74, New York, NY, USA, 2004. ACM. ISBN 1581139756. DOI 10.1145/1032604.1032616.

[269] L. Ye and E. Keogh. Time series shapelets: A new primitive for data mining. In J. Elder, F. S. Fogelman, P. Flach, and M. Zaki, editors, *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 947–956, New York, NY, USA, 2009. ACM. ISBN 9781605584959. DOI 10.1145/1557019.1557122.

[270] C. Yin, S. Zhang, J. Wang, and N. N. Xiong. Anomaly detection based on convolutional recurrent autoencoder for iot time series. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 52(1):112–122, 2022. DOI 10.1109/TSMC.2020.2968516.

[271] U. Yokkampon, A. Mowshowitz, S. Chumkamon, and E. Hayashi. Robust unsupervised anomaly detection with variational autoencoder in multivariate time series data. *IEEE Access*, 10:57835–57849, 2022. DOI 10.1109/ACCESS.2022.3178592.

[272] C. Yu, L. Luo, L. L. H. Chan, T. Rakthanmanon, and S. Nutanong. A fast lsh-based similarity search method for multivariate time series. *Information*

Sciences, 476:337–356, 2019. DOI 10.1016/j.ins.2018.10.026. URL http://www.sciencedirect.com/science/article/pii/S0020025518308430.

[273] W. Yu, I. Y. Kim, and C. Mechefske. Analysis of different rnn autoencoder variants for time series classification and machine prognostics. *Mechanical Systems and Signal Processing*, 149:107322, 2021. DOI 10.1016/j.ymssp.2020.107322.

[274] Y. Yu, T. Becker, and M. Behrisch. Multivariate time series retrieval with symbolic aggregate approximation, regular expression, and query expansion. In *EuroVis Workshop on Visual Analytics*, Rome, 2022. DOI 10.2312/eurova.20221081.

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

[276] Y. Yu, T. Becker, L. M. Trinh, and M. Behrisch. Saxregex: Multivariate time series pattern search with symbolic representation, regular expression, and query expansion. *Computers & Graphics*, 112:13–21, 2023. DOI 10.1016/j.cag.2023.03.002. URL https://www.sciencedirect.com/science/article/pii/S0097849323000316.

[277] Z. Yu, N. A. Kraft, and T. Menzies. Finding better active learners for faster literature reviews. *Empirical Software Engineering*, 23(6):3161–3186, 2018. DOI 10.1007/s10664-017-9587-0.

[278] Z. Zamanzadeh Darban, G. I. Webb, S. Pan, C. Aggarwal, and M. Salehi. Deep learning for time series anomaly detection: A survey. *ACM Comput. Surv.*, 57(1):1–42, 2025. DOI 10.1145/3691338.

[279] A. Zeiler, R. Faltermeier, I. R. Keck, A. M. Tome, C. G. Puntonet, and E. W. Lang. Empirical mode decomposition - an introduction. In *The 2010 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2010. ISBN 978-1-4244-6916-1. DOI 10.1109/IJCNN.2010.5596829.

[280] C. Zhang and C. Czado. Vine copula-based bayesian classification for multivariate time series of electroencephalography eye states. *Journal of the Royal Statistical Society Series C: Applied Statistics*, page qlad038, 2023. DOI 10.1093/jrsssc/qlad038.

[281] D. Zhang, M. Ma, and L. Xia. A comprehensive review on gans for time-series signals. *Neural Computing and Applications*, 34(5):3551–3571, 2022. DOI 10.1007/s00521-022-06888-0.

[282] J. Zhang and K. F. Man. Time series prediction using rnn in multi-dimension embedding phase space. In *SMC Conference Proceedings. 1998 IEEE International Conference on Systems, Man, and Cybernetics (Cat. No.98CH36218)*, pages 1868–1873. IEEE, 1998. ISBN 0-7803-4778-1. DOI 10.1109/ICSMC.1998.728168.

[283] K. Zhang, Q. Wen, C. Zhang, R. Cai, M. Jin, Y. Liu, J. Y. Zhang, Y. Liang, G. Pang, D. Song, and S. Pan. Self-supervised learning for time series analysis: Taxonomy, progress, and prospects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PP, 2024. DOI 10.1109/TPAMI.2024.3387317.

[284] J. Zhao and L. Itti. shapedtw: Shape dynamic time warping. *Pattern Recognition*, 74:171–184, 2018. DOI 10.1016/j.patcog.2017.09.020.

[285] D. Zhou, J. Li, and W. Ma. Clustering based on lle for financial multivariate time series. In *2009 International Conference on Management and Service Science*, pages 1–4. IEEE, 2009. ISBN 978-1-4244-4638-4. DOI 10.1109/ICMSS.2009.5305089.

BIOGRAPHY

Yuncong Yu is a fourth-year Ph.D. student in the Department of Information and Computing Sciences, Faculty of Science at the Utrecht University in the Netherlands. Simultaneously, he is working in the technical solution provider IAV GmbH Ingenieurgesellschaft Auto und Verkehr as a data scientist.

He received a master of science degree in the Department of Mechanical Engineering at Karlsruhe Institute of Technology (KIT) in Germany. His master thesis titled *Analysis, Comparison, and Interpretation of Multivariate Time Series* at the Institute of Vehicle System Engineering at KIT and in AUDI AG addressed multivariate time series segmentation & comparison, and subsequent system parameter estimation for automotive climatization and thermal management system.

He is interested in the study, development, and application of physics-based and data-driven modeling and simulation. In his study and career, he acquired knowledge and experience especially in time series analysis and visual analytics, in addition to general data science and software development.

ACKNOWLEDGMENTS

Words cannot express my gratitude to my supervisor Prof. Michael Behrisch, my industrial supervisor Dr. Tim Becker, and my promoter Prof. Alexandru C. Telea, who generously provided knowledgeable expertise and guidance. Most of the brilliant ideas originated from them. I would also like to express my deepest appreciation to my assessment committee for the invaluable feedback.

This endeavor would not have been possible without the support from IAV GmbH, who financed my research. I would like to thank my colleagues, especially Dr. Peter Schichtel, who introduced me to IAV to proceed with my research in time series analysis and co-supervised me at the beginning of the research; Dr. Christian Kruschel, who also co-supervised me at the beginning of my industrial practice; Dr. Anna Bertram, who carefully reviewed my publication and coordinated user studies; Le Minh Trinh, who contributed significantly to the development of the web applications appeared in my publications; Dr. Thomas Müller, Dr. Christian Nitschke, and Dr. Jaroslav Vondrejc, who created and maintained the foundational Data Science Toolkit that greatly expedited my algorithm development and patiently provided advice. I would like to thank Thomas von der Ohe and Matthias Schultalbers sincerely, who took an interest in my research and became my stakeholder. Many thanks also to Dr. Tobias Niebuhr, who supervised our projects and lavished me with approbation that I would like to live up to.

I would also like to recognize my Visualization and Graphics group. I'm very proud to be a member of this motivated, productive, and friendly group. In particular, I'd like to acknowledge the contributions of Dylan Kryuff and Jiao Jiao to our publication. Furthermore, I would like to extend my sincere thanks to Jet Haasbroek, my Ph.D. mentor, who built up my confidence as I embarked on my Ph.D. journey.

Special thanks go to Prof. Wolfgang Aigner at St. Pölten University of Applied Sciences for his direct contribution in our recent work and to Prof. Eamonn Keogh at University of California, Riverside for his professional feedback on our work.

I owe my profound gratitude to Dr. Thomas Mayer and Dr. Thomas Weutenfeld, who supervised my master thesis and encouraged me to further my research as a Ph.D. student. Immense

gratitude also goes to my colleges in TLK Thermo GmbH, who supported me in seeking a Ph.D. position after my graduation as a master student and kindly offered me a job as close as possible to my future research before I found the current Ph.D. position.

Lastly, I would be remiss in not mentioning my family, especially my mother Shenghong Feng, father Xiaobo Yu, my aunt Xiaoming Yu, and my wife Bingbin Shen who urged me to get out of my comfort zone, challenge myself, and try to contribute to the advancement of mankind. Their belief in me has kept my spirits and motivation high during this undertaking.

Yuncong Yu
June 6th, 2024

COLOPHON

This document was typeset using the typographical look-and-feel `classicthesis` developed by André Miede. The style was inspired by Robert Bringhurst's seminal book on typography "*The Elements of Typographic Style*". `classicthesis` is available for both LaTeX and LyX:

`http://code.google.com/p/classicthesis/`

Final Version as of December 16, 2024 (`classicthesis`).