UNIVERSITY OF CALIFORNIA
RIVERSIDE

User-Guided, Interpretable, Actionable Time Series Data Mining Methods

A Dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Computer Science

by

Audrey Elaine Der

September 2024

Dissertation Committee:

Dr. Eamonn J. Keogh, Chairperson
Dr. Jiasi Chen
Dr. Christian Shelton
Dr. Evangelos Papalexakis

Acknowledgments

Many thanks to my advisor, Eamonn J. Keogh. I am grateful your research modus operandi has shaped mine. Thank you to my committee, Professor Jiasi Chen, Professor Christian Shelton, and Professor Evangelos Papalexakis for your counsel throughout my time at UCR (undergraduate degree, included). Your past experiences and expertise have served as a map while I navigate new realms. Thank you to Professor Gareth Funning for serving on my Oral Examination committee. Thank you to Professor Jia Chen and Professor Paea Le Pendu for your mentorship. Thank you to Vanda Yamaguchi and Mindi Jo Mobley, for whomst all of us would be lost in the administrative sea of paperwork, bureaucracy, and unknown protocol. Your willingness and joy in ushering and guiding forth new generations of scholars inspires me.

Thank you to my colleagues, coauthors, and labmates for your support and friendship: Dr. Ryan Mercer, Akhil Srinivas (godspeed on your own PhD journey!), Dr. Michael Yeh, Dawon Ahn, Dr. Renjie Wu, Dr. Maryam Shahcheraghi, and Dr. Zachary Zimmerman. Let's keep in touch, and I hope to see you at conferences in the future. Our days in the lab together have been a comfort to me, knowing none of us were going at it alone. Thank you to my friends who have supported me even before I knew I was going to embark on this 43-thousand-and-some-hours long tutorial for "How to Conduct Research": Afsaneh Pouraghabagher, Ravneet Brar, Amanda Israel, Kaleb Branda, Sophia Vinegar, Leila Mariah Omar, Marvin Chowdhury, Justin Chandra, Matthew Choi, Kyle Choi, Kevin Hsieh, Breena Li, Omar Peraza, and Justin Lam. The real game begins now.

Thank you to my family, who have supported me and valued my education from the beginning; it truly took a village to raise me. Should my siblings or cousins choose to pursue this path, I am but a phone call away to help get you started.

Thank you to Calvin, the most steadfast partner I could have asked for in all of this, and in all of life. It's your turn next.

To my grandparents and great-grandparents who went through the ordeal of immigrating here, working sunup to sundown so their descendants might not have to.

ABSTRACT OF THE DISSERTATION

User-Guided, Interpretable, Actionable Time Series Data Mining Methods

by

Audrey Elaine Der

Doctor of Philosophy, Graduate Program in Computer Science
University of California, Riverside, September 2024
Dr. Eamonn J. Keogh, Chairperson

Many time series data mining algorithms are designed to be intuitive for domain experts, often including case studies and benchmark results. However, it is challenging to create truly domain-agnostic algorithms due to unique domain-specific invariances. In these works, we prioritize and leverage user input and interpretability over complexity. We empirically demonstrate that many time series data mining tasks are still a searchable space, allowing for interpretable methodologies. We adopt a "Soup-to-Nuts" design, envisioning how the method fits into a user's problem, from accepting their real-valued time series dataset to providing a direct answer to the data mining task. With these three characteristics in mind, we propose methods that are user-friendly, not requiring the user to have extensive experience with time series data mining methods and find the results of our methods actionable.

In the first work, we target a problem setting of long time series. Many time series datasets have been cleaned and preprocessed to be of uniform length and have guarantees that the time series contains some semantic data the dataset advertises. We relax these two constraints, considering signals without any assumptions about their contents. In the second work, we propose a time series anomaly explanation method and format. With the increasing relevance of time series anomaly detection, we formulate an intuitive and actionable anomaly explanation method, addressing the often complicated and unclear actionability of existing explanations. In the third work, we propose a method for time series clustering, leveraging feature sets for interpretability and user input for customizability, guiding our methodology to make better-suited and more intelligible decisions for users based on their specific needs.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

With the rise of vast quantities of data to mine, commonly referred to as Big-Data [76], mining such data became an area of great interest. Data Mining refers to extracting data from large collections of data [9, 47]. Time series data mining specifically focuses on *time series* data.

Definition 1 *A* ***time series*** *T is a sequence of real-valued numbers $t_i : T = [t_1, t_2, ..., t_n]$ where n is the length of T.*

Most data mining algorithms do not operate on the entire time series, but instead consider only local *subsequences* of the time series.

Definition 2 *A* ***subsequence*** *$T_{i,j}$ of a time series T is a continuous subset of data points from T of length $j - i$ starting at position i. $T_{i,j} = [t_i, t_{i+1}, ..., t_{i+j-1}], 1 \leq i \leq n - j + 1$.*

The length of the subsequence is typically set by the user based on domain knowledge. For example, for most human actions, half a second may be appropriate, but for considering traffic congestion patterns, subsequences of one or two days may be more fitting. This atime series data is ubiquitous; if one has the means to measure and record phenomena at a systematic rate in time, they can create a time series dataset. The work detailed in this dissertation keeps the user experience at the forefront of its design, and that led us to focusing on four characteristics when designing our methods.

First and foremost, we wanted our methods to be **interpretable**, and for the practitioners using them to understand how decisions were made, and why our methods output what they do. Interpretability often walks hand in hand with simplicity. Particularly in healthcare [37], there have been propositions that methods should be *interpretable by design* [14, 73], as explanations themselves are not inherently trustable. Additionally, some explanations themselves are more complicated than the phenomena they seek to explain.

Secondly, domain experts have unique needs for their unique tasks and domains. The work in Chapters 2 and 3 are **easily modifiable** to accomodate these unique needs. The work described in Chapter 4 incorporates a means of accepting human-in-the-loop, or **user-input**, by utilizing active learning, allowing users to guide the method as it runs. The ability to interact with and modify these methods reduces “algorithmic aversion”, and facilitates user trust in methods’ predictions [14, 32].

Thirdly, we wanted the insights our methods provided to be **actionable**. The actionability of explainable AI (XAI) is of such import, there are calls for XAI specifically with this characteristic (AXAI) [77, 90]. Each of the works in this dissertation describes an end-to-end methodology, or a “Soup to Nuts” approach. As a result, the aforementioned characteristics enable our methods to have a “low-barrier of entry”. The methods and code are designed to be easy to adopt and run, and do not require “considerable” computational resources. All experiments were conducted single-threaded on a shared server containing an Intel® Xeon® E5-2680 v3 at 2.50 GHz with 377 GB of RAM.

Chapter 2

Long Time Series

2.1 Introduction

Distance measures are perhaps the most fundamental data mining primitives. Armed only with an appropriate distance measure, we can perform clustering, classification, anomaly detection, summarization, repeated pattern discovery, etc. Given the ubiquity and variety of time series, there are dozens of distance measures defined for this data type. However, almost all such measures are designed for short time series, say individual gestures, individual heartbeats, etc. Suppose instead we wish to address long time series. Surprisingly, there is very little work that considers such data. One solution advocated in [34] is to convert the long time series into a fixed feature vector, for example: `A = [Mean(T), Skewness(T), Hurst(T)]`. Such an approach may be appropriate in some domains, however, in many cases the *shapes* of local subsequences may be the most discriminating features, and these are typically not well captured by global features.

Note that we cannot simply use classic time series distance measures. For example,

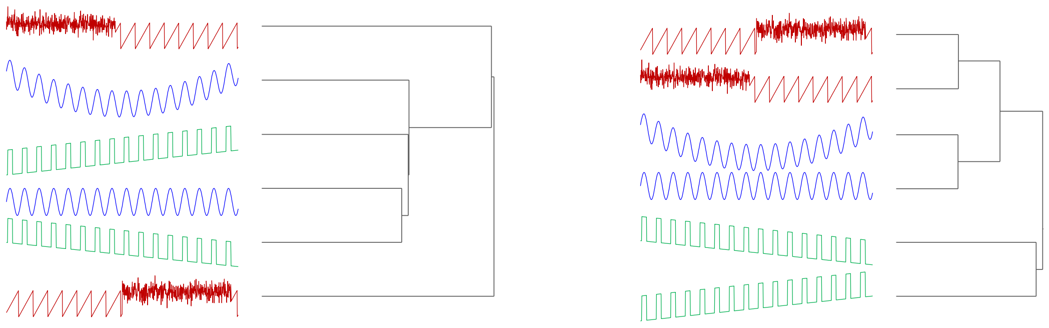


Figure 2.1: Six time series, comprising of three obvious pairs, clustered using single linkage clustering. *left*) The clustering produced by DTW. *right*) the clustering produced by PRCIS, the method proposed in this paper.

DTW is rightly lauded for its invariance to out-of-phase local segments [102]. However, suppose we wish to compare two one-minute ECG segments from the same person. Because the heart rate is constantly drifting, it is likely that the two one-minute ECG segments will have a different number of beats, say 67 and 72. DTW is simply unable to "explain" the five extra beats, and will be forced to return a large distance suggesting the two traces are very dissimilar.

In addition, DTW (which includes Euclidean distance as a special case) would not be suitable for long time series even if there was some mechanism that keep the periodicity fixed. Consider the example shown in Figure 2.1.*left*. While each colored pair has the same periodicity, minor corruptions of the global trends are enough to thwart DTW's attempt to group similar pairs.

The rest of this paper is organized as follows. In Section II we present our motivating observations. Section III describes the formal definitions and background and allows us to outline our approach. Section IV contains an extensive experimental evaluation. Finally, we offer conclusions and thoughts on future directions in Section V.

2.2 Motivation and Related Work

Because we have used the term “long time series” without defining it, we repair that omission now. For our purposes, “long” does not necessarily correlate with the number of datapoints. For example, a single heartbeat may be a second long, and represented with 128 datapoints. However, some ECG apparatus can record at up to 32,768 Hz. In a sense, a single heartbeat with either 128 or 32,768 datapoints covers the same wall clock time and has the same intrinsic dimensionality. By long time series we mean a time series that has at least hundreds of features, such as peaks and valleys.

While there are excellent distance measures for time series, such as Euclidean distance and DTW, these measures are suited to atomic subsequences, for example, comparing two heartbeats, or two gestures, or two days of road traffic density. Note that the UCR/UEA Archive [27] *exclusively* contains datasets of this type. The top performing methods on that archive include deep learning and non-deep learning based methods, including: HIVE-COTE and its variants [54, 60], ROCKET and its variants [30, 31], TS-CHIEF [83] and TS-QUAD [53], InceptionTime [46], BOSS (a dictionary learning algorithm) [79], and a good baseline in ResNet [96].

In contrast, we desire a distance measure that can compare items at a higher semantic level, for example comparing two hour-long ECG traces, before and after a medical intervention, or comparing two 90-minute soccer games, or comparing road traffic density of two freeways over a year, etc. It should be clear that neither Euclidean distance nor DTW will work well if directly applied in such circumstances. For example, two one-hour traces of walking gait may have a differing number of steps. DTW can compare two steps that are locally out of phase, but it cannot map say ten steps to eleven steps, and that single unexplained step will completely swamp any similarity that may or may not exist.

Likewise, imagine comparing the road traffic density of two freeways over a year. We could imagine that two freeways have near identical traffic patterns, but one of them comes from a suburb that has a slowly increasing population. This slight linear trend difference might be imperceptible over a day or a week, the typical scale considered by Euclidean distance. However, over a year, the small linear trend difference will again completely swamp any similarity that may or may not exist. These issues are understood in the community, but to date we are unaware of any automatic solution. Attempts to mine such data typically involve a lot of human intensive preprocessing steps, cleaning the data (detrending, normalizing, imputation of missing values, etc.), and the manual extraction of clean and representative examples to feed into downstream algorithms.

Our basic plan is to create a compact dictionary for each time series and calculate the distances between each pair of dictionaries. The term *dictionary creation/learning* is somewhat overloaded in computer science. It is often a synonym for sparse coding, a class of representation learning methods which aims at finding a sparse representation of the input data in the form of a linear combination of basic elements as well as those basic elements themselves. The elements of the representation learned do not typically come from the data. In contrast, we propose using dictionaries comprised of elements from the data itself.

2.3 Definitions and Background

We present four ways a dictionary of short time series patterns could be constructed from a long time series.

- **Yeh's Dictionary** [101]: Consider a time series T with conserved patterns and supply a parameter window length w (which ideally is close to the natural length of the patterns within T) and the number of patterns n desired for the resulting dictionary. Computing the Matrix Profile (MP) of T will reveal the locations of these motifs. Building the distance profile of the motif Q against T will indicate the locations of Q and subtracting this distance profile from MP will remove all locations of Q from MP. Repeating this process n times will produce up to n patterns. Unless otherwise stated, this is the dictionary creation algorithm we use in this work.

- **Time Series Snippets** [42]: A snippet is an unsupervised time series primitive that rewards both fidelity and coverage. Given an input time series T, the Snippet Selection Algorithm captures "typical" behavior in an ordered list of substrings. There are a handful of other research efforts that produce similar compact summarizations of time series, for example BeatLex (Beat Lexicons for Summarization) [40].

- **Random Dictionary**: Here we imagine building a dictionary by randomly extracting elements of a long time series. This is meant to be the most naïve baseline, nevertheless, random sampling is known to be competitive for some tasks.

- **Calendar Dictionaries**: This is an option only for datasets that are constrained by human circadian patterns (e.g., traffic density, network usage, etc.). Someone familiar with the domain can hand curate a dictionary by selecting a handful of informative days. For example, for a collection of time series representing traffic density in California, she might ensure that each dictionary has one weekday, one weekend day, one summer day, one winter day, one bank holiday, etc.

While our proposed distance measure is agnostic to the dictionary creation method used, unless otherwise stated, we will use Yeh's Dictionary in this work. Regardless of how the dictionary is created, we represent a dictionary of a single time series as a *dictionary exemplar.*

Definition 3 *A **dictionary exemplar** (henceforth simply referred to as a dictionary where there is no confusion) refers to any dictionary produced by a dictionary learning method where the input is a time series T and has an output of a dictionary $T_d = \{p_1, p_2, ..., p_n\}$ where there are n patterns.*

Figure 2.2 illustrates a dictionary that represents a time series containing two different semantic patterns. We refer to the values for the above notation to be the size of the dictionary S with patterns of length L. S and L are simply the parameters passed to the dictionary creation algorithm. Suppose a random dictionary algorithm created a $T_{d[S,L]}$ such that it extracted S patterns of L length as random non-overlapping subsequences from T. In this case, the notation is true to value. In contrast, Yeh's Dictionaries extract and then merge patterns with overlapping subsequence indices, making S an upper bound and L a lower bound [101].

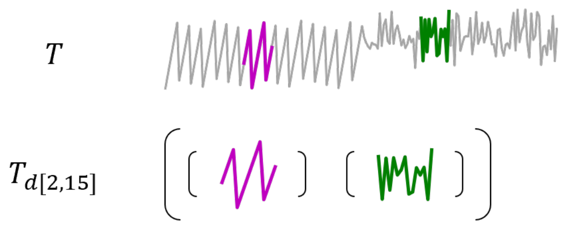


Figure 2.2: *top*) A time series T with two different semantic patterns, a sawtooth and a noisy signal. *bottom*) A dictionary that summarizes T, with a cardinality of two with patterns of length 15, denoted as $T_{d[2,15]}$.

2.3.1 Desirable Properties of a Distance Measure

Before introducing a distance measure defined for dictionaries, it will be instructive to consider what properties are desirable for such a measure. To allow us to be concrete, we will use discrete strings as proxies for time series. The reader will appreciate that continuing this analogy, the Hamming distance is an excellent proxy for the Euclidean distance. For example, the Hamming distance `d(cat,bat)=1`, represents two quite similar time series with a low Euclidean distance.

With the exception of perhaps calendar-based or other handcrafted dictionaries, it is obvious that we cannot expect dictionaries to be phase aligned. For example, suppose we have two similar long series:

```
A = ...catcatcatcatcatdogdogdogcatcat...
B = ...dogdogcatcatcatcatcatdogdogdog...
```

If we use Yeh's algorithm to create two dictionaries from these datasets, we may obtain:

$$A_d = \{\texttt{cat},\texttt{dog}\}$$

$$B_d = \{\texttt{atc},\texttt{dog}\}$$

We do not wish to penalize the apparently great distance between "`cat`" and "`atc`", as we realize that is just the result of the vagaries of the dictionary building algorithm. We can be invariant to this by holding one word fixed, and comparing to every circular shift of the other word. Here, `RI_dist` is a phase invariant distance or "k-shape" [65], computed in $O(mlog(m))$. This is equivalent to:

```
RI_dist(cat,atc) = min(d(cat,atc), d(cat,tca), d(cat,cat))
```

There is another issue that the dictionary creation algorithm may throw at us. Revisiting examples A and B above, we may be presented with:

$$A_d = \{\texttt{cat},\texttt{dog}\}$$

$$B_d = \{\texttt{dog},\texttt{cat}\}$$

Dictionary building algorithms may not be consistent in ordering patterns with respect to frequency or representativeness, and two time series could be very similar, but produce dictionaries with words in different orders. This is true even for calendar-based dictionaries. For example, because Bangkok and Lima are near perfect antipodes, they have similar climates, but exactly six months out of phase. Once again, this is easy to handle. For each pattern in dictionary A, find its nearest neighbor in dictionary B, and vice versa.

```
d_dict(A_d,B_d) =
{min(RI_dist(cat,dog), RI_dist(cat,cat)) + min(RI_dist(dog,dog),
RI_dist(dog,cat))} +
{min(RI_dist(dog,cat), RI_dist(dog,dog)) + min(RI_dist(cat,cat),
RI_dist(cat,dog))}
```

The first pair of curly braces is the computed distance from A_d to B_d, and the second pair is the computed distance from B_d to A_d. It is important to note that `d_dict` is not the functionality of PRCIS, but the expression is useful to communicate the intuition behind PRCIS. For flexibility, we desire the capability to compare patterns of varying lengths and dictionaries of varying cardinalities. Given two more time series:

```
C = ...catcatcatcatdogdogdogcowcowcow...
D = ...bobcatbobcogdogdogdbobcatbobcatbo...
```

This now gives us four time series dictionaries to consider:

$$A_d = \{\texttt{cat},\texttt{dog}\}$$

$$B_d = \{\texttt{dog},\texttt{cat}\}$$

$$C_d = \{\texttt{dog},\texttt{cat},\texttt{cow}\}$$

$$D_d = \{\texttt{ogd},\texttt{bobcat}\}$$

The issue of varying cardinalities is trivial, we would simply have more `RI_dist` computations inside the curly brace pairs. Likewise, the varying lengths issue is easy to handle. We just need to generalize the rotation invariant distance to allow strings of different lengths. We can do this by taking the longer of the two strings, concatenating it to itself, using the shorter string as a query, and finding the best substring within the longer string.

```
RI_dist(bobcatbobcat,cat) = d_substr(bobcatbobcat,cat)
```

Using these ideas, the distance between {cat,dog} and {bobcat,textcolorredcat} will be zero. This may initially be unintuitive until you review the strings that originally produced them, and recognize that they could be very similar:

```
A = ...catcatcatcatcatdogdogdogcatcat...
D = ...bobcatbobcogdogdogdbobcatbobca...
```

We are almost done, but there is one last desirable invariance. It may happen that while two dictionaries are generally very similar, one of them has one or more words that are unique. For example, imagine we have:

$$X\_d = \{\texttt{cat},\texttt{dog},\texttt{cow}\}$$

$$Y\_d = \{\texttt{cat},\texttt{dog},\texttt{cow},\wedge\%\$*\}$$

The unusual word in Y_d may have many causes. Perhaps the corresponding sensor had a long disconnection artifact that the dictionary algorithm thought worthy of representation. The problem is that although these two dictionaries are near identical, the single strange pattern will dominate the overall distance, and mask the similarity. Our solution is to forgo the brittle mean distance (between each word and its rotational invariant nearest neighbor in the other dictionary), but instead use the median distance. The median distance is much more forgiving of a handful of words that have no obvious match.

This section has been somewhat long and detailed. However, with these ideas in mind, the exposition of the real-valued PRCIS distance measure in the next section is straightforward.

2.4 PRCIS Dictionary Distance Measure

PRCIS stands for Pattern Representation Comparison in Series, and is the distance measure by which we propose to measure the similarity between two arbitrary real-valued dictionaries. We outline PRECIS in Algorithm 2, which calls Algorithm 1.

Algorithm 1 finds the nearest neighbors for every pattern in dictionary A from the

Algorithm 1 A subroutine in PRCIS (Algorithm 2).

```
1: function PRCIS_AtoB($T_{d1}, T_{d2}$)                         ▷ The distance from $T_{d2}$ to $T_{d1}$
2:     dists ← []
3:     for $p_A \in A_d$ do
4:         $nn\_dist \leftarrow \infty$
5:         for $p_B \in B_d$ do
6:             $l, s \leftarrow p_A, p_B$                  ▷ Switch assignments if len($p_B$) > len($p_A$)
7:             $ts, q \leftarrow$ concat(l,l), $s$
8:             $dp \leftarrow$ abs(MASS($ts, q$))
9:             $min\_dp \leftarrow$ min($dp$)
10:            if $min\_dp < nn\_dist$ then
11:                $nn\_dist \leftarrow min\_dp$
12:            end if
13:        end for
14:        dists.append($nn\_dist$)
15:    end for
16:    return $dists$
17: end function
```

patterns in dictionary B. On line 1, the list of distances is initialized. For every element in A, its distance to every element in B is calculated in part by running MASS [67]. Between the pairs of patterns, the shorter pattern is treated as the query, q, and the longer pattern is concatenated to itself and treated as the time series, ts (lines 5-7). In lines 8-12, the minimum of the minimums of these distance profiles is recorded as the distance from that pattern in A to its nearest neighbor in B. This occurs twice.

When the second time Algorithm 1 is called, for every pattern in B, calculate the distance to every element in A. When this is done, lines 3-4 of Algorithm 1 return the square of the median of this list of distances, and this represents the distance between the dictionaries A and B.

Algorithm 2 The algorithm for PRCIS.

```
1: procedure PRCIS($A_d, B_d$)                    ▷ Dist. between $A_d$ and $B_d$
2:     $AtoB \leftarrow$ PRECIS_AtoB($A_d, B_d$)
3:     $BtoA \leftarrow$ PRECIS_AtoB($B_d, A_d$)
4:     dists $\leftarrow$ concat($AtoB, BtoA$)
5:     return (median(all_dists))$^2$
6: end procedure
```

The resulting distance is a measure, but not a metric. However, Bronstein et al. [21] and others have forcefully argued that in some cases the metric property of triangular inequality must be violated to produce sensible results. Moreover, the ubiquitous DTW is also not a metric, but regarded as SOTA for comparing short time series [35, 102].

We say n is the largest number of words in any dictionary, and the number of dictionaries being compared is m. Generally, m is nontrivially large, and $n << m$. MASS is proven to be extremely fast [67] (progressively so with newer versions), so we abstract its runtime to $O(1)$. The time complexity of PRCIS is $O(m^2n^2)$. Additionally, the runtime of MASS only depends on the length of patterns. The number of dictionaries to compare or the cardinality of each dictionary have a greater effect on runtime. Memory is not a bottleneck for this distance measure.

2.5 Empirical Evaluation

To ensure our experiments are reproducible, we have created a supporting website [6] containing all data, code, and results. Additionally, it houses other experiments and comparisons omitted here for brevity. All experiments were conducted single-threaded on a shared server containing an Intel® Xeon® E5-2680 v3 at 2.50 GHz with 377 GB of RAM. An exception to the single-threaded constraint was made for the Yeh Dictionary creation process and training of NN classifiers, which were permitted 4 jobs. All series were z-normalized in preprocessing. In all of our comparisons to Catch22 [57] and k-Shape [6], we replace NaN values from input time series with the mean of non-NaN values.

2.5.1 Clustering

We begin by showing the utility of our algorithm for clustering, as this allows the most direct and visually intuitive demonstration of the invariances that our measure achieves. The full set of combinations of hierarchal clustering parameters (average, single,

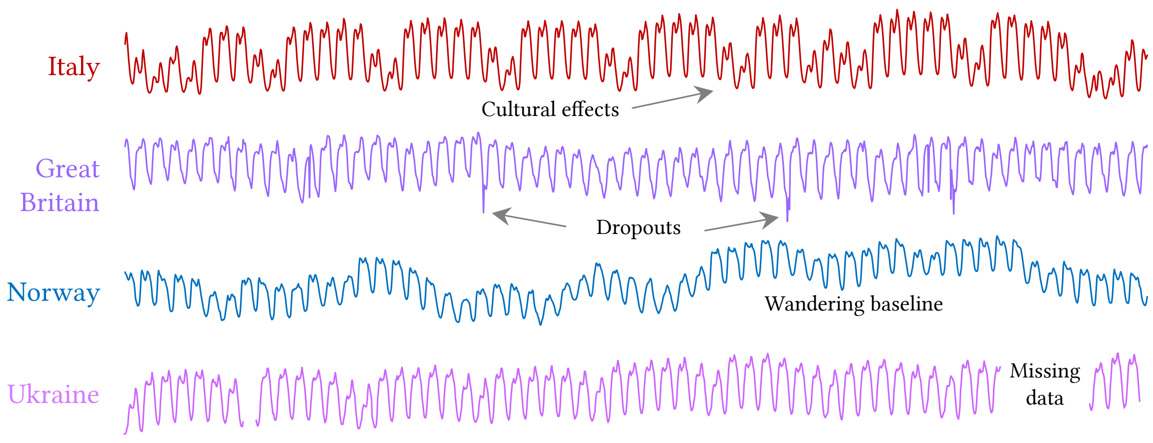


Figure 2.3: Two-month subsequences of the electrical power demand data from four countries in Europe.

or complete linkage under L1, L2, or cosine similarity) are available at [5].

Electrical Power Demand

We consider a three-year long dataset of electrical power demand from European countries [99]. As Figure 2.3 shows, the data are surprisingly diverse. There are at least three sources of variability. The first is cultural; many Mediterranean countries regard weekends as sacrosanct, and they have a visible decrease in power required for weekends, as many businesses close (see Italy in Figure 2.3).

There are also obvious weather effects, the Mediterranean climate has low variability, but countries in northern Europe have a power demand that clearly reflects the vagaries of the climate (see Norway in Figure 2.3). Finally, there may also be policy effects. For example, some countries incentivize off peak consumption, however this is implemented differently in different countries. For example, Denmark, Norway, and Sweden use spot-market-based pricing, whereas Estonia, Spain and the UK use dynamic real-time pricing [43]. Figure 2.3 further hints at why this dataset is challenging from most clustering algorithms. There are regions of missing data and dropout and spikes that seem to be meaningless (at least to the task at hand) but are likely to confuse most algorithms. Nevertheless, Figure 2.4 shows a successful clustering.

This dataset spans from December 31, 2014 23:00:00 GMT to September 30, 2020 23:00:00 GMT. The results in Figure 2.4 are very intuitive, reflecting the major regions of Europe. A handful of placements are debatable. Denmark is a Scandinavian country; however, it does share a 68-kilometer border with Germany but is only connected to Sweden by a bridge. While the overall clustering is very intuitive, Austria is a surprising outlier that we might otherwise have expected to group with Germany. We can only speculate as to the reason. Austria is famous for having "golden Sunday", a tradition (formally enshrined in federal law in 2003 [3]) for all shops to be closed on Sunday. This deference to the sabbath has all but disappeared in the rest of Europe. In any case, this is exactly why we do such clusterings, to discover interesting and unexpected findings that can be further investigated.

The dendrogram in Figure 2.4 was created using a PRCIS distance matrix under $T_{[4,48]}$. We opted $L = 48$ to cover the transition from weekends to weekdays (and weekdays to weekends) and chose $S = 4$ to cover different types of days we experience, such as a *weekday* (Monday–Thursday), *weekend*, *holiday*, and *Fridays*.

We attempted to achieve similarly intuitive results with other methods, including k-Shape [65], Catch22 [57], and obvious anomaly in any country and clustering using only that day. None of these attempts yielded a clustering that was significantly better than random. For brevity, we report these experiments in [5].

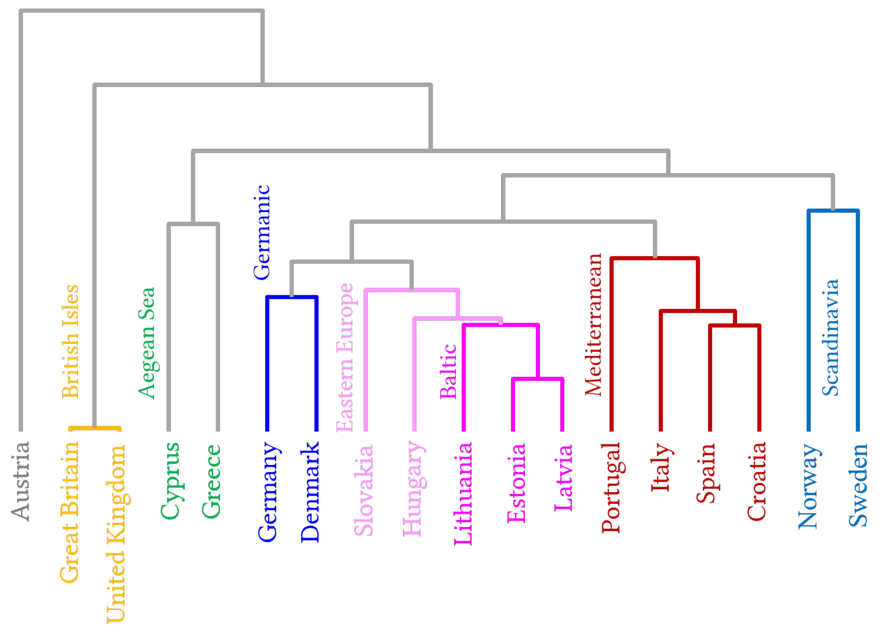


Figure 2.4: The electrical power demand data from 25 of the countries under complete linkage hierarchical clustering using $T_{[4,48]}$, where 48 datapoints represent two days.

Human Mobility/Gait

As challenging as the previous dataset was, there was some mechanism (diurnal cycles) that at least ensured that the periodicity is essentially constant. Moreover, all time series were the same length. In this experiment we consider a dataset that does not offer even those weak consistencies.

We consider the dataset created in [33], that records the gait of volunteers, some sighted, some blind, as they walk various routes in the corridors of a building. Unlike some gait datasets that are recorded on treadmills or other idealized situations, this dataset is recorded "in the wild". As noted by the original authors, "*our participants walked on multiple paths with different levels of complexity, including turns at 45, 90, and 180 degrees, as well as through doors that needed to be opened. While walking, our participants occasionally*

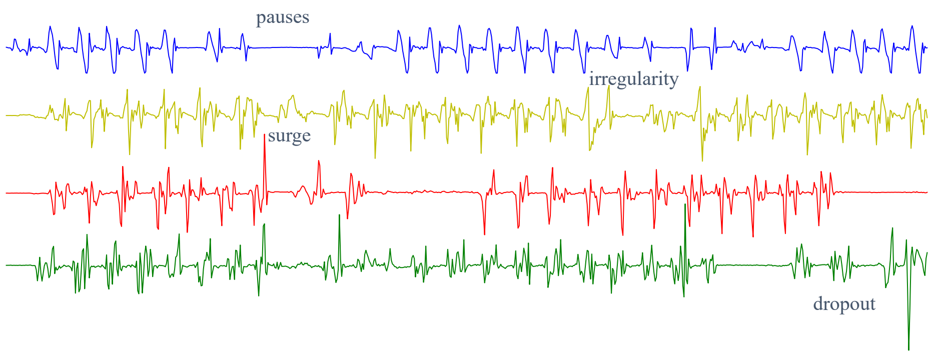


Figure 2.5: Four examples from the WeAllWalk dataset [33] that hint at the variability of the data.

veered off the straight path, got caught in wall openings, and collided with obstacles...". In Figure 2.5.*top* we hint at the diversity of this dataset, and in Figure 2.6.*top* shows the result of average linkage hierarchical clustering on the data.

We wanted to avoid over polishing the parameters. To achieve this clustering, we use $T_{[8,25]}$, where 25 time steps span one second. The eight-element dictionary size was our rough guess as to how many behaviors we might see, given the description of the data. For example, we hypothesized that we might see: *normal gait, gait slowing down approaching a door, strained gait while pushing a heavy door*, etc.

The clustering is objectively correct, except for record T4-ID6-WC clustering with the wrong group, belonging to ID7 (green). We speculate this is because ID6 and ID7 are the only two individuals who used a cane, apart from ID8 (blue). However, ID8 contributed twelve datapoints, half from walking with the use of a cane, and the other half with the aid of a guide dog. The inter-similarity of the ID8 cluster may have caused T4-ID6-WC to be clustered with ID7 instead. It is not clear what the best baseline comparison is here, excluding direct application of most standard distance measures. The lengths of the time

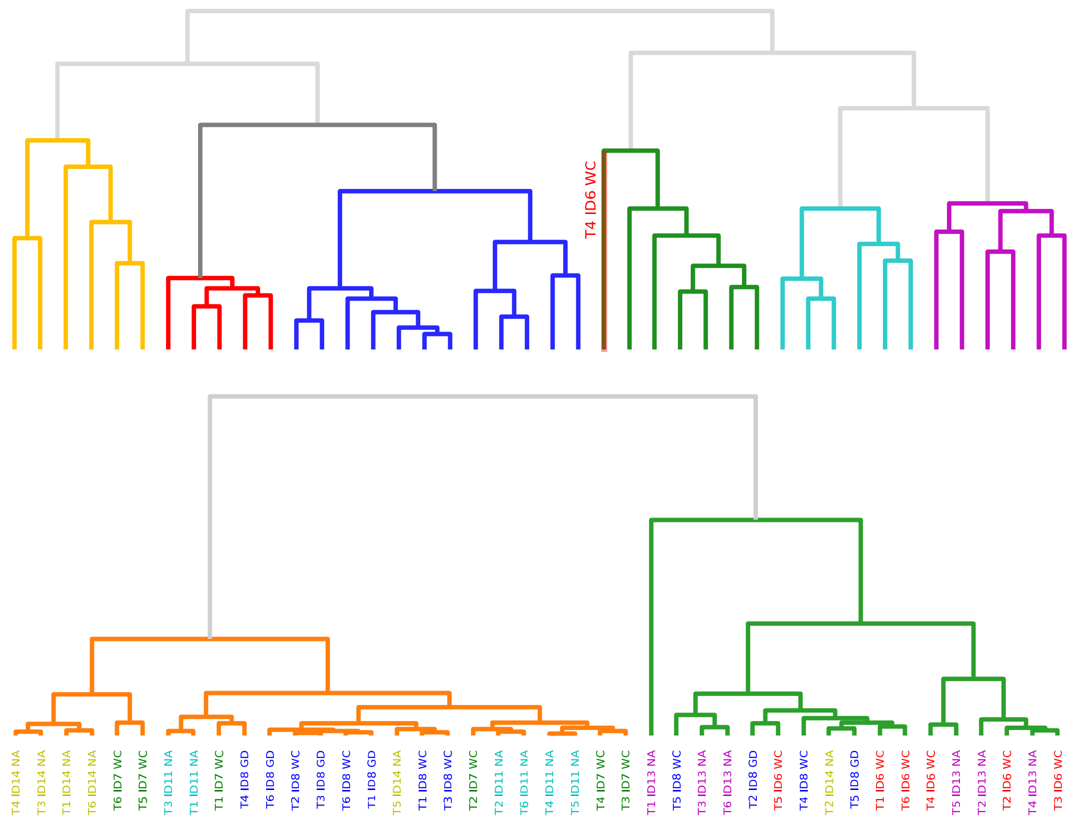


Figure 2.6: The complete linkage clustering of forty-two time series representing the Y-axis motion of hip-worn a 25 Hz accelerometer (an iPhone 6). The six colors correspond to six different individuals. *top*) PRCIS *bottom*) All features of Catch22, a SOTA time series feature ensemble [57].

series varied from 61 seconds to 309 seconds. However, as shown in Figure 2.6.*bottom*, a feature ensemble proposed for time series classification is better than random clustering assignments, but does not produce a clustering as well as PRCIS does (Figure 2.6.*top*). We relegate the figures for the best performing rival method to [5]. In all cases, even the best of the dozens of possibilities is not as good as our proposed method.

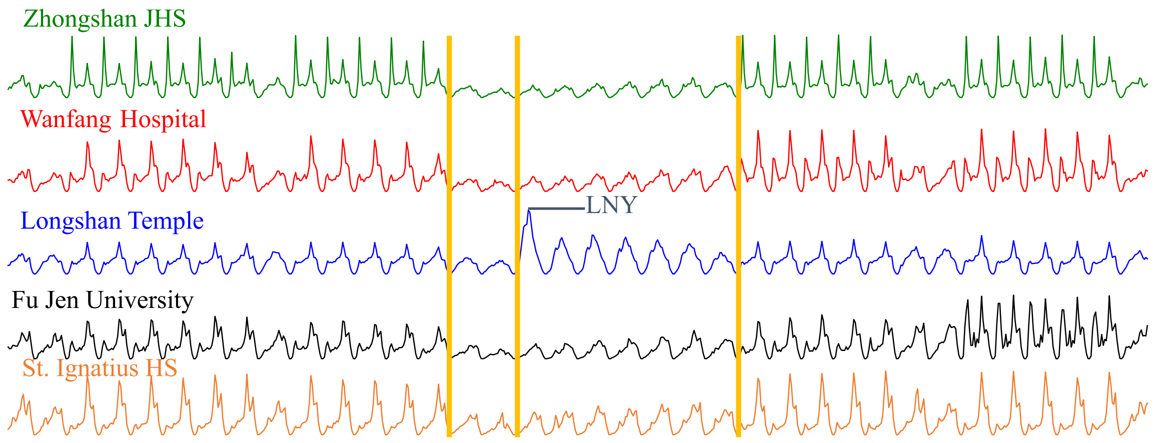


Figure 2.7: Five stations' data from the MRT dataset. The first golden line is placed at the start of Feb. 6th (a Saturday), the second is on Lunar New Year—Feb. 8th 07:00—(a Monday), and the third is the end of Feb. 14th (a Sunday).

2.5.2 Taipei Mass Rapid Transit (MRT)

The Taipei MRT dataset details the number of riders that enter and exit at each of the 108 stations from November 2015 to March 2017. Figure 2.7 shows the dates from January 24, 2016 to February 27, 2016. The original data is available at http://data.taipei/, but we have used a version from [102].

Unlike the previous example, there is no ground truth. However, we can see the dendrogram in Figure 2.8 has two halves that decidedly see different levels of traffic, the green cluster seeing the highest traffic on average. Taipei Main Station sees significantly more traffic than any other station. Within the green cluster, there is a subcluster of stations near department stores. Dingpu and Yongning, Qili'an and Qiyan, and Chaing Kai-Shek Memorial Hall and NTU Hospital are three pairs of adjacent stations.

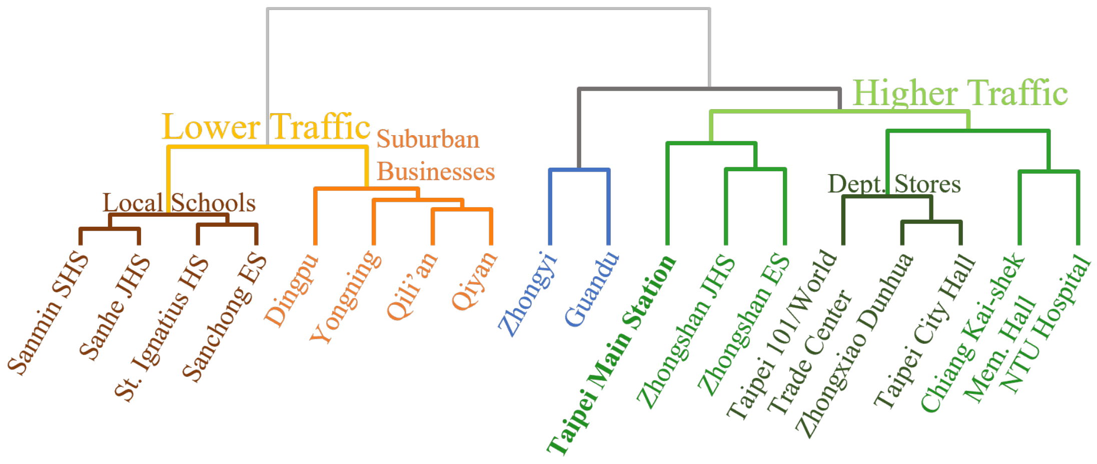


Figure 2.8: Station entry traffic under complete linkage clustering. Given the full series, $T_{d[8,40]}$ where 40 hours spans two days of operation. We select $S = 8$ due to the large degree of polymorphism seen in just the subsequences shown in Figure 2.7.

Milling Machine Table Acoustic Emissions

The previous datasets were relatively low frequency datasets. Here we consider data sampled at 250Hz. The NASA/Berkeley Milling Dataset [4] contains acoustic emission from a milling machine under different depths of cuts, feeds, and materials used. In Figure 2.9.*top*, we see that these cases are difficult to visually distinguish.

It might not be obvious why we need a dictionary-based method here. However, the documentation that accompanies the data makes it clear that the data is polymorphic. There is data corresponding to three phases: *entry* cut, *steady* cut, and *exit* cuts (although the boundaries between these behaviors are not explicitly given). In addition, there may be differences due to the direction of motion relative to the cutting head rotation, i.e., climb cut vs. conventional cut, although this is not documented. Table 2.1, contains details of the classes clustered in Figure 2.9.

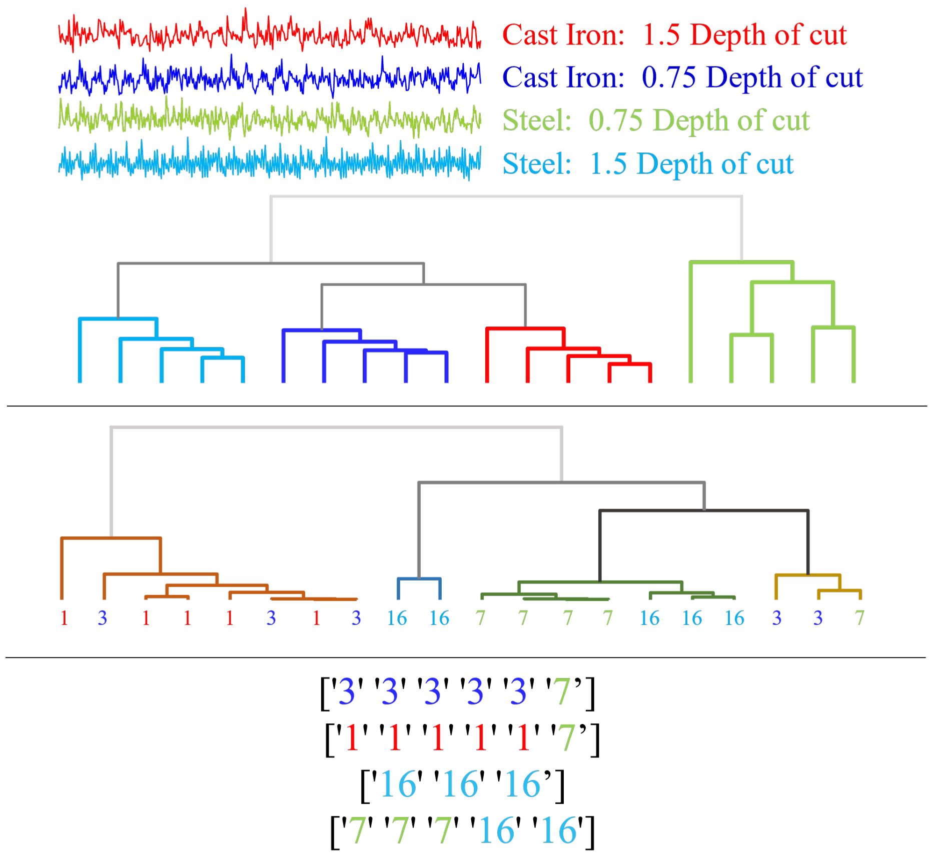


Figure 2.9: *top*) Two-second-long samples from four semantic classes in a machining dataset.
top-middle) The average linkage clustering of twenty such time series under $T_{d[4,512]}$.
bottom-middle) Catch22 under complete linkage clustering.
bottom) The clustering produced by k-Shape [65].

Case	**Depth of Cut**	**Feed**	**Material**
1	1.5	0.5	Cast Iron
3	0.75	0.25	Cast Iron
7	0.75	0. 25	Steel
16	1.5	0.5	Steel

Table 2.1: The classes and conditions of the time series in Figure 2.9.

Using PRCIS, we can achieve the clustering in Figure 2.9.*top-middle*, where each of the series is 9,000 datapoints, equivalent to 36 seconds of wall clock time. In Figure 2.9.*bottom-middle* we cluster the same series using the Catch22 [57] framework. Because this dataset is of both uniform length and has an available ground truth, we can make a third comparison in k-Shape [65] (Figure 2.9.*bottom*). These clusters are more homogenous than those produced by Catch22, but do not distinguish cases 7 and 16 nearly as well. Additional comparisons of clusterings are reported at [5].

2.5.3 Business Merchants

Identifying a merchant's true business type is vital to ensure the integrity of payment processing systems, as a merchant may intentionally or unintentionally report a false business type to the payment process network [104]. When representing merchants with hourly aggregated time series, time series analysis is effective for identifying a merchant's business type [104]. For example, toll collections peak during rush hours and meal delivery services are busy around lunch and dinner hours.

Thirty-eight merchants from five different categories (vending machines, online videogame retailers, toll collection, parking meters, and meal delivery services) are sampled for this study. The names of the merchants are not released to maintain confidentiality. For each merchant, we generate the number of transactions per hour from January 1, 2018 to June 30, 2021. In Figure 2.10, we show an example of two vending machines.

In Figure 2.11 we compare Yeh's Dictionaries + PRCIS with three baselines, visualizing each merchant as a coordinate computed using t-SNE. Our method and baselines'

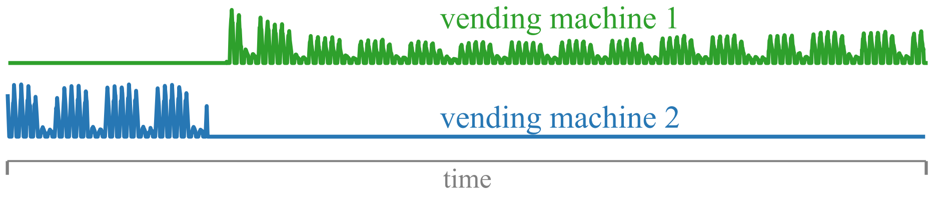


Figure 2.10: Vending machine 1 may be newly added, and vending machine 2 may be recently removed. Computing the distance between them from a global perspective is doomed to fail.

problem constructions are as follows:

- **Yeh's Dictionaries + PRCIS**: We construct a dictionary $T_{d[18,168]}$ for each series, where each pattern of L=168 spans one week. Should Yeh's dictionary algorithm return a T_d with the maximum 18 patterns allowed, the total length of the patterns in T_d will be no longer than 10% of the length of the full series.

- **Entire Series + DTW**: The time series are z-normalized and the DTW distance is computed pairwise. DTW computes distances between subsequences of the full series, each 182 weeks long.

- **Random 10% + DTW**: A random subsequence 10% of the original time series length (just over 18 weeks) are extracted then z-normalized, and the DTW distance is computed pairwise between these subsequences. This baseline is to study if the success of PRECIS is due to using only a subset of the original time series.

- **Systematic Random 10% + DTW**: Similar to Random 10%, each merchant is also represented with a subsequence in lieu of the full series. However, all merchants use subsequences covering the same period of time.

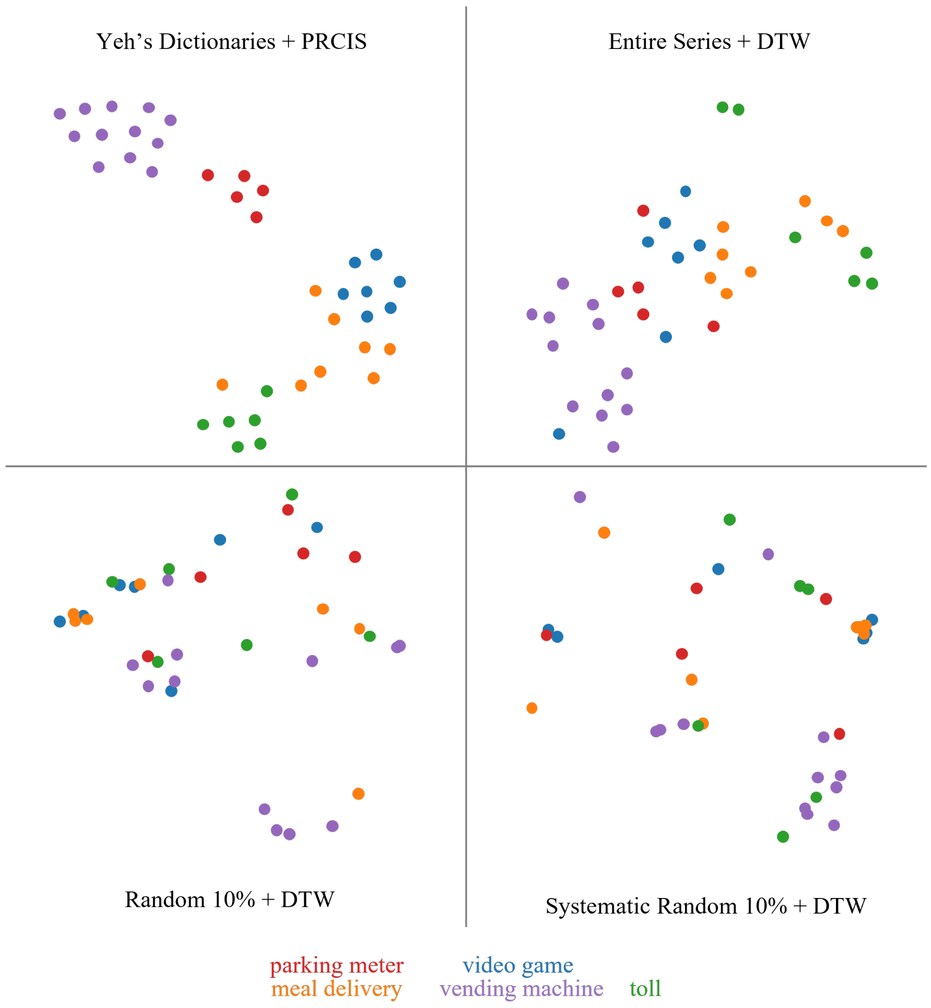


Figure 2.11: Four different combinations of inputs and distance measures. The scatterplot associated with PRCIS shows a clear separation of merchants from the five business types.

PRCIS successfully captures the business type of the merchants, attending to local subsequences at a meaningful scale (one week). The other baselines look at each series from a much longer time horizon. Series like those in Figure 2.10 have long "no signal" subsequences, and a distance measure will fail if unable to ignore these.

2.5.4 Classification

Here we turn our attention from clustering to classification. There are literally thousands of papers on classification of short time series, say individual heartbeats or gestures. (In essence, every paper that cites the UCR classification archive). However, these works assume that individual heartbeats/gestures/events have been extracted from the data streams. In many cases, this extraction may be a more difficult task than the classification problem itself. As the clustering examples we have considered hint at, we are interested in much longer, unstructured datasets in domain agnostic settings. There is very little literature on time series classification under these assumptions.

Here we consider two very different long time series classification challenges. While this is a subjective claim, we believe that in general neither of these tasks could be solved by human inspection. For example, for USC-HAD [106] `Walking-Forward` and `Walking-Upstairs` are visually very similar. To avoid over tuning the algorithm, we simply spent a few minutes "playing" with a tiny subset of dataset, to select the two parameters we needed to set. For each algorithm considered, we used 1NN Leave-One-Out (LOO) classification to evaluate its performance.

Despite that the classification works we reference in Section 2.2 are trained on single-event time series, we compare to ROCKET for completeness. ROCKET is the fastest of the SOTA mentioned in Section 2.2 and executes in approximately the same time scale as Catch22 and PRECIS ("minutes to hours" as opposed to "hours to days"). The out-of-the-box implementation from the authors [30] does require a uniform length time series dataset. We offer results using zero-padding and padding any shorter series to itself, clipping to the length of the longest series. As aforementioned, WeAllWalk series lengths varied from lengths 1,525-7,725 datapoints, and USC-HAD series lengths varied from 600-13,500 datapoints.

The feature-based approach we compare to is Catch22. Catch22's best subset of features was selected through simple greedy forward selection. If an input time series results in a Catch22 feature vector containing any NaN values, the series is simply discarded, and not counted against Catch22. This situation was not applicable to WeAllWalk, but in USC-HAD, 28 such series were discarded.

In the case of Random Dictionaries + PRCIS, reported results are the average of 10 runs. Six nonoverlapping subsequences of length 150 were selected randomly. In the cases where time series may be shorter than $S * L$ (in this case, $S * L = 900$), we opt to include as many patterns of length L as possible. For example, no more than four patterns can be extracted from a series of length 600 when $L = 150$. By sight, the smallest unit of cycles seemed to appear in lengths of 50 datapoints, so we opted to select a pattern length following a factor of 50.

We compared two variants of PRCIS. The random dictionary version offers evidence for our claim that using a *subset* of the data can be better than using *all* the data, and the Yeh's Dictionary variant offers evidence that well-chosen subset of the data is better again. Table 2.2 summarizes the results.

Not listed in Table 4 2.2, we reference the reported results on USC-HAD by [106] using handcrafted features and a tuned multilayer perceptron to obtain an accuracy of 74.71%. In contrast, we have opted to not preprocess the dataset to remove spurious

Dataset	**USC-HAD**	**WeAllWalk**
Dataset Size	630	42
Default Rate	11.11	28.75
Dictionary Parameters	$T_{d[6,150]}$	$T_d[8, 25]$
ROCKET, zero-padding	0.0	28.57
ROCKET, self-padding	0.0	28.57
Catch22, Best Feature	36.67	76.19
Catch22, All Features	44.13	66.67
Catch22, Best Feature Subset	64.92	97.62
PRCIS, Rand Dict.	71.08	92.14
PRCIS, Yeh's Dict.	73.33	100.0

Table 2.2: The LOO Accuracy of seven approaches on three datasets.

subsequences or extract individual gait cycles. Additionally, many papers edit the classes considered in nonstandard ways. We have set nothing except the pattern length to 150 based on a quick visual inspection of a few traces.

2.5.5 Effect of Dictionary Size

Classification experiments are the most direct way to evaluate the effect of dictionary size on PECIS's ability to effectively represent long time series. While there is a single semantic label for each class, i.e., "`Walking-Forward`", our claim is that for a long time series, it is unlikely that any single subsequence can represent the concept. For example, even the apparently monolithic "`Walking-Forward`" may actually consist of multiple regions: *getting-up-to-speed*, *cruising-at-constant-speed*, etc. If our assumption is true, we would expect to see poor results for the smallest dictionary size and increasing (but with diminishing returns) accuracy as we make the dictionaries larger.

Largely, the results in Figure 2.12 supports our hypothesis: while a dictionary of size one is significantly better than default rate random guessing, additional words do

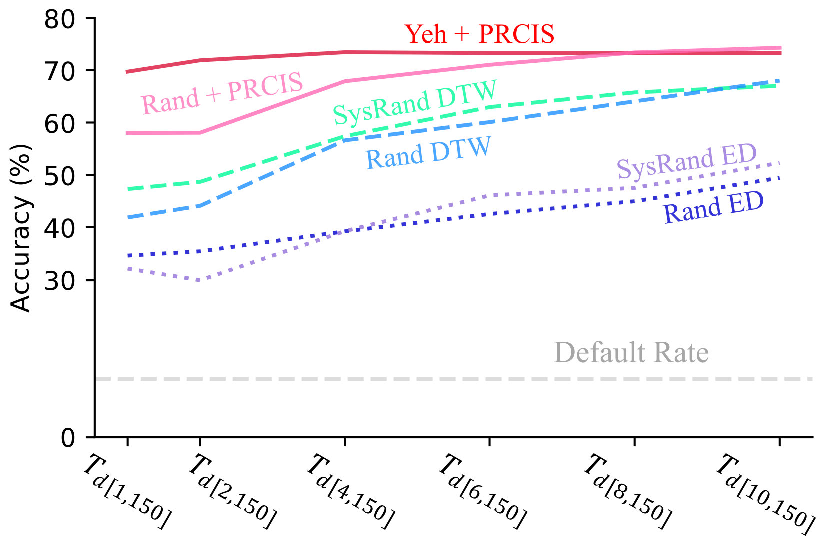


Figure 2.12: The dictionary size vs. accuracy on the USC-HAD over different pairs of dictionary algorithms and distance measures.

improve accuracy. However, the improvements yield diminishing returns as the larger dictionaries begin to fully represent the diversity of the behavior. In the case of Yeh+PRECIS, passing in a greater value for S seems to have little effect after six patterns. This is because most of the behaviors here are relatively homogamous, more diverse behaviors (i.e., the electrical power demand data of Section 2.5.1) with its different cultural and weather seasons) may require larger dictionaries.

Competitive accuracy is of little use unless the model executes in a workable time frame; to consider this, we measure throughput. We calculate throughput as the average wall clock time required for one dictionary-to-dictionary distance computation. PRCIS makes $(m^2 - m)/2$ such computations, where m is the number of time series in a dataset.

We mentioned in our discussion of Figure 2.12 that passing in a greater value for S produces diminishing returns after its peak at six patterns under Yeh+PRCIS. This is also

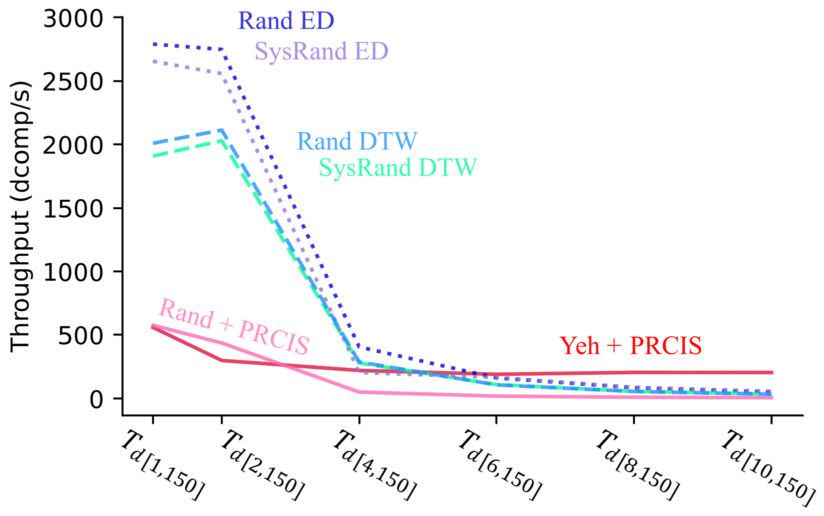


Figure 2.13: The throughput of each distance combination of distance measures and dictionary creation methods reported with respect to our experiments on USC-HAD in Figure 2.12.

reflected in Figure 2.13, where we can see the throughput stops decreasing. Throughput is in the units of dictionary comparisons per second (dcomps/s). This seems to suggest that, with PRCIS, a good dictionary creation method that minimizes spurious or redundant patterns is an efficient and accurate approach in the space of long time series. (We recall that Yeh's Dictionary method will *merge* individual patterns into a single pattern if their subsequence bounds overlap.)

2.5.6 Towards PRCIS Anomaly Detection

There has been a recent explosion of interest in time series anomaly detection. A wealth of techniques have been applied to this task (including dozens of variants of deep learning [13]. However, there is increasing evidence that simple distance-based methods are competitive in this area. For example, a recent paper compared over a dozen variants of SOTA deep learning approaches to a simple distance-based method (discords as represented by the Matrix Profile [42]) on 250 datasets, and discovered that the distance-based method was significantly better [13].

However, this result only addresses short anomalies, a single arrythmia or a single stumble in a long normal walk. Our warnings about the appropriateness of techniques for short time series being blindly applied to long time series apply here, perhaps even more so than for clustering or classification. In particular, we claim that there are some problems for which essentially all techniques in the literature are inappropriate. To demonstrate this, consider the following example. The example is slightly contrived, but uses real data [1], and reflects a real industrial problem.

The performance of electrical motors is often monitored with accelerometers. Some types of motor anomalies are easy to detect from such data, for example a faulty bearing or worn brushes. However, there are much subtler patterns that may be anomalous. For example, an S3-class industrial motor is designed for intermittent periodic duty, a sequence of cycles containing periods of constant loads and (typically) a period at rest. There may be periods of different levels of constant loads; a machine may cycle between *rest*, *high* load (to raise a conveyor), *low* load (to lower the conveyor), back to *rest*, in an endless cycle. It should be clear that the normal behavior of this device is polymorphic, no single time series shape can represent all the version cycles.

It may be possible to build an anomaly detector in this domain with detailed domain knowledge. However, we propose to address this problem with a simpler approach. We propose to build a dictionary from a sample of normal data. We will not “tell” the algorithm where the different cycles begin (in any case, we generally may not know this).

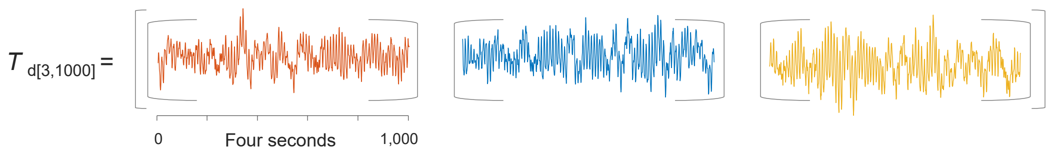


Figure 2.14: A Yeh's Dictionary $T_{d[3,1000]}$, computed from a trace of normal behavior from an industrial shear.

Instead, we assume that the dictionary creation algorithm will automatically choose representative patterns, so long as the dictionary size is greater than or equal to the true number of atomic behaviors. In Figure 2.14 we have done exactly this on an example of normal behavior of an industrial paper shear for a large volume book printer.

In this case, the data is so complex that it is impossible to visually confirm that it represents the diversity of duty cycles in the industrial process. However, we can test this. In Figure 2.15, we create a PRCIS *distance profile*, "sliding" each pattern of the dictionary across an eleven-minute trace from the same motor. Because PRCIS the distance measure (Algorithm 2) is a distance measure between a *dictionary* and a *dictionary*, we briefly cover the PRCIS distance profile in Algorithm 3, which is defined as the distance from a *dictionary* to a *time series*.

The data in Figure 2.15.*top* has a 32-second anomaly beginning at about five minutes in. Gratifyingly, the PRCIS distance profile (Figure 2.15.*bottom* peaks at the location of the anomaly, caused by one of the cycles drawing a higher load due to a paper jam. Before continuing discussion, we want to ward off a possible misunderstanding. PRCIS is designed for long time series. Here long does not refer to the data being monitored, in any case that is data normally assumed to be effectively unbounded. Instead, long here refers to

Algorithm 3 The algorithm to compute the PRCIS Distance Profile.

```
1: function PRCIS_DistProf($D, T$)          ▷ A dictionary $D$ and a time series $T$.
2:     $dps \leftarrow []$
3:     for $p_d \in D$ do
4:         $dps$.append(MASS($T, p_d$))
5:     end for
6:     $meta_{dp} \leftarrow$ mean($dps$)          ▷ Elementwise mean.
7:     $PRCIS_{dp} \leftarrow$ movemean($meta_{dp}, L$)
8:     return $PRCIS_{dp}$
9: end function
```

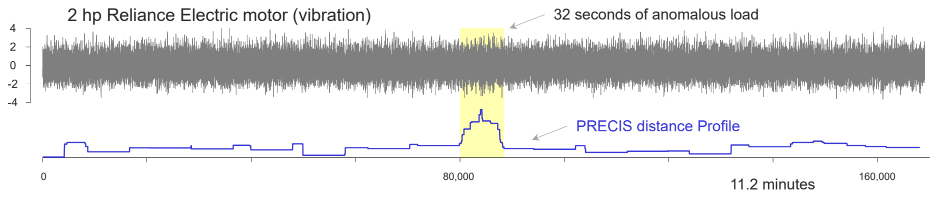


Figure 2.15: *top*) An eleven-minute trace of motor powering an industrial paper shear. *bottom*) The distance between the dictionary shown in Figure 2.14 to local subsequences of length 1,000 measures with the PRCIS distance measure. This curve was smoothed for clarity, not affecting the result. The parameters for smoothing were a factor of L.

the length of the subsequences being considered. For example, the patterns in Figure 2.14 are much longer than the length of subsequences considered in all anomaly detection studies we are aware of [13, 42].

We have shown that PRCIS can detect (or at least peak at) at the anomaly; can other approaches? In Figure YY we consider the Matrix Profile [42], and Telemanom [41], a widely cited deep learning approach. The Matrix Profile was computed using a subsequence window of 256. The Telemanom Profile is an error curve under exponential weighted moving

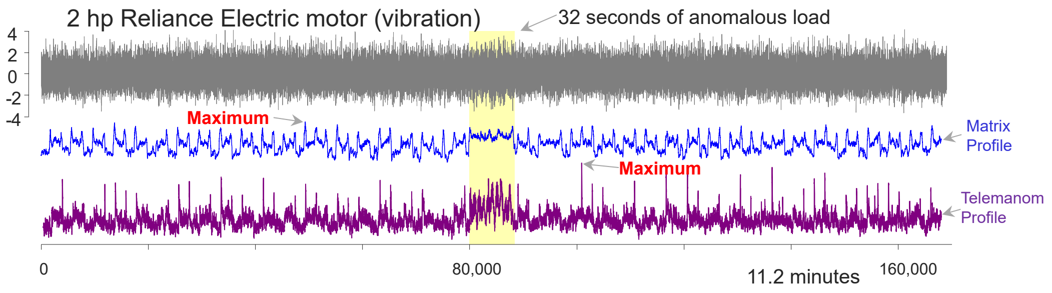


Figure 2.16: Telemanom and the Matrix Profile on the same eleven-minute trace in Figure 2.15. The distance profiles for these algorithms peak in the wrong places.

average smoothing (EWMA smoothing). In fairness to the inventors of Matrix Profile and Telemanom, while they do not explicitly state this, it is clear that they intended their algorithms to work with short time series subsequences, not the longer semantically diverse subsequences/regions we are considering here.

This example is somewhat tentative and speculative. Unlike the more common "short anomaly" problem, there are currently no standard benchmarks to evaluate this task. However, we believe that this example shows the promise of considering anomaly detection problems at the higher level of combinations of behavior, rather than a single irregular shape.

2.6 Conclusions and Future Work

We have introduced PRCIS, a novel distance measure for comparing long time series. We have shown that using PRCIS can produce intuitive and semantically correct clusterings and embeddings of data. Moreover, it can do this in the presence of noise, spikes, dropouts, and even missing data. In future work we plan to consider further downstream uses of PRCIS, and investigate ways to set its two parameters automatically, probably by exploiting recent advances in using Minimum Description Length (MDL) for time series [40]. We have made all our code and datasets freely available in perpetuity, to allow others to confirm and extend our findings.

Chapter 3

Time Series Anomaly Explanations

3.1 Introduction

Recent years have seen significant advancements in Time Series Anomaly Detection, or TSAD, with state-of-the-art algorithms now outperforming even domain experts [55]. For example, consider the one-minute snippet of Electrocardiogram (ECG) data shown in Figure 3.1. Even with careful visual inspection, is not apparent to the human eye that this time series has any anomalies. Nevertheless, as Figure 3.2 shows, an anomaly can be discovered in the data.

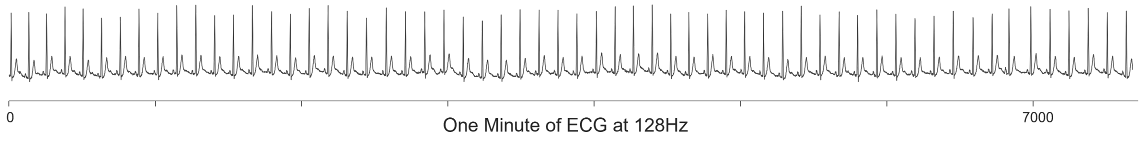


Figure 3.1: One minute of ECG data. From out-of-band data, we know that it contains a single anomaly, which must be very subtle.

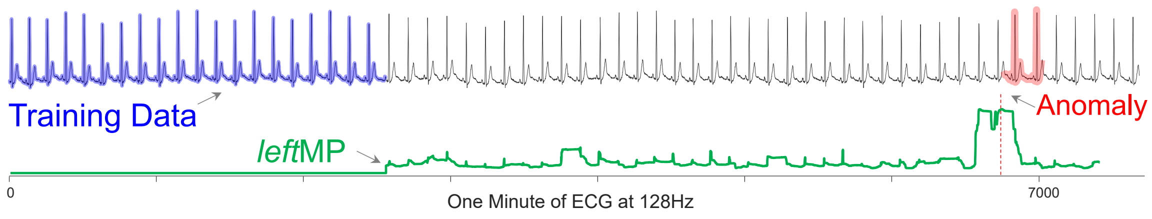


Figure 3.2: The ECG shown in Figure 3.1 processed with the leftMP [109] set to find two-second long anomalies. The algorithm indicates an anomaly (highlighted in red) at location 6,763 (red dashed line).

Without the out-of-band confirmation that the algorithm did indeed find a true anomaly, it is not clear that we would trust this result. Even if we did trust it, perhaps recalling that this algorithm had performed very well on similar data, we would surely be curious as to why the algorithm had flagged this. To this end, we conducted both interviews with domain experts and an extensive review of their literature on anomalous findings. We noted that domain experts often explain anomalies by noting what would have to change to make the anomaly appear normal, essentially completing the following template:

`Anomaly would be like A except for corruption B.`

Let us attempt this for the anomaly we discovered in Figure 3.2. Under the Euclidean distance (ED) that leftMP algorithm used to detect this anomaly, it was 15.3 units from its nearest neighbor in the training data. As shown in Figure 3.3, if we simply replaced ED with the dynamic time warping (DTW) distance, that number drops to just 2.3. This dramatic change suggests the cause of the anomaly was not a change in *shape* per se, but simply local changes in the timing of the beats.

As Figure 3.3 shows, from the computed warping alignment, it appears that the anomaly has a short beat followed by a long beat. In fact, if we Google that exact phase,

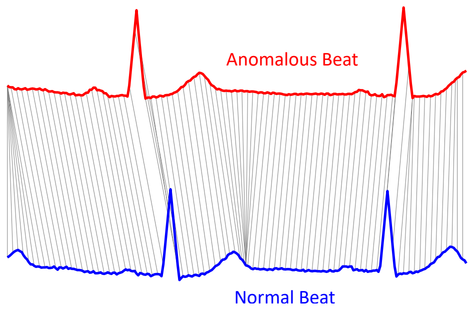


Figure 3.3: The anomaly compared to its DTW nearest neighbor in the training data.

the number 1 hit is a book that contains the text "*..compensation beats: a short beat* ***followed by a long beat****, which together last the same duration as two normal beats..*" [59]. According to noted cardiologist Dr. Gregory Mason, the anomaly we discovered is indeed a *compensation beat*, a particular type of supraventricular arrhythmia.

In Figure 3.4.*left*, we show another anomaly discovered later in the same dataset. For this anomaly using the DTW distance instead of ED makes no difference. However, as Figure 3.4.*right* suggests, a little smoothing drastically reduces the distance between the anomaly and its nearest neighbor. This suggests that this anomaly is caused by, and therefore best explained by, *noise.* These examples outline our basic approach to time series anomaly explanation. Given an (perhaps tentative) anomaly, we will test a small set of operators that may be able to transform the anomaly into a normal data pattern. The most parsimonious transformation (or rather, its *inverse*) can be seen as the best explanation of the anomaly.

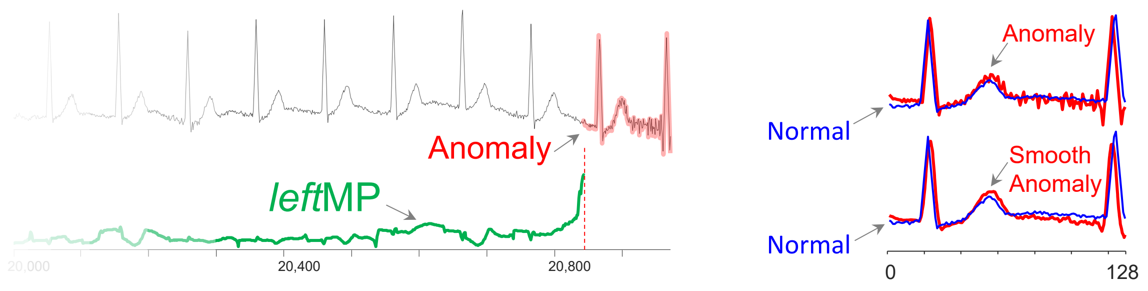


Figure 3.4: *left*) Another anomaly found in the ECG. *right*) The anomaly overlayed with its nearest neighbor before and after smoothing.

This idea opens two challenges:

- What is the correct set of operators that is expressive enough to explain all anomalies?
- Assuming that some of these operators are in different units, how do we make them commensurate, so that they can be ranked, and the best explanation offered?

In this work we introduce principled answers to both questions. We demonstrate the intuitiveness, and the correctness and actionability of our explanations on diverse domains.

The rest of this paper is organized as follows. Section 3.2 contains our motivation and Section 3.3 our closest related works. In Section 3.4, we introduce our methodology and the operators it leverages. In Section 3.4.11, we discuss how we made them commensurate and comparable. In Section 3.5, we present our empirical results in the form of a benchmark.

3.2 Motivation

The recent successes in the community's ability to find anomalies have outpaced its ability to explain them, once discovered. It is not even clear what format an anomaly explanation should take. The literature is replete with ideas [26, 92]. Weight matrices, heatmaps and feature-saliency maps [39,93], rule-based systems [16], and weighted trees [20] have all been employed.

However, many of these approaches seem very indirect. In some cases, the explanations seem to be more complex than the data they seek to explain. In addition, for most of these techniques, the method of anomaly explanation in inextricability tied to the method of anomaly discovery. As we will later argue, it is advantageous to divorce these two steps.

Returning to our original question, what format should an anomaly explanation take? Rather than impose our preferred method on the users of anomaly detection systems, it makes sense to understand how domain experts normally talk, reason and communicate about anomalies. The follow examples give a flavor of this (emphasis ours):

- "..similar to pattern 1 except for a deep valley" [23]
- "..like ATMs 1 and 2, except for a large spike" [97]
- "..similar to the vertical spectra except for the peak" [50]
- "..similar to western SIO except it shows two crests" [64]
- "This is similar to inhibition-induced spiking, except that the response is a burst." [51]

Note that while these examples are from diverse domains, they all have the same basic structure. In each case the explanation says what corruption must be removed (equivalently, what edit could be performed) in order to make the anomaly look like what normal pattern. For example, the authors of [23] suggest that if the anomaly is edited to remove the deep valley, it would look like a normal pattern 1. Such explanations are often written in the form of `Anomaly would be like A except for corruption B`. The reader will note that such explanations are *counterfactual explanations*. As our examples suggest, such explanations appear to be universally used in industry [12, 87], science, and medicine.

Counterfactual examples (or *counterfactuals*) are a popular form for explanations in explainable AI (XAI), perhaps because they are amenable to the way people "select to mutate in their representation of reality" [22]. Counterfactuals describe what should be different in a system's input to produce a different outcome, allow for imagining better (as opposed to worse) outcomes, and do not focus on improbable events [22]). Counterfactuals are thought to be particularly helpful if they are *sparse* and *proximate* [29].

These observations motivate PUPAE (Personalizable Universal Plausible Anomaly Explanations), our proposed definition of anomaly explanation. The formal definition is in Section 3.4.12, but can be informally stated as: A PUPAE is the minimum change that can be made to an anomaly to make it similar to some normal data at location *Loc* in the training data. Note the location *Loc* in the training data is an intrinsic part of the explanation. For example, consider these two following explanations:

- The pattern looks like Monday Dec 25th 2023 if we remove the spike at the beginning of the anomaly.
- The pattern looks like Monday Dec 4th 2023 if we remove the spike at the beginning of the anomaly.

Note the distortion is the same in both cases, but in the former case, we can see that this anomaly is more similar to a holiday, than a regular working day.

3.3 Related Work

Several works explore the use of natural language counterfactuals for various domains, including NLP [68] and computer vision [38]. However only a handful of research efforts have considered counterfactual explanations for time series. Additionally, out of the methods we discuss in this section, PUPAE is the only method that uses natural language counterfactual explanations for time series. Some methods to explain and visualize time series tasks include by way of clustering [86], feature importance and heatmaps [39, 93], or in terms of other subsequences of the time series accompanied by natural language [87]. TSXplain [63] generates a paragraph of textual based on results from a statistical feature extractor, listing numerical values for statistical features.

PUPAE can be best be categorized as a *perturbation-based method*, which directly computes the contribution of the input features by altering them, computing over the altered input, and measuring the difference between the original input [10]. Other perturbation-based methods include Native Guide [29] and TeRCe [16]. However, these methods generate synthetic time series subsequences to visually juxtapose with the anomaly. In contrast, PUPAE uses only real data and generates *natural language* counterfactuals.

3.4 Counterfactual Operators

We are now able to explain our proposed operators. These operators are based on our inspection of all publicly available TSAD benchmarks [36, 55, 98, 100], reviewing the literature [8, 23, 50, 59, 78], and interviews with several domain experts in cardiology (Dr. Greg Mason), oil&gas processing, and entomology (Dr. Kerry Mauck). While we shall show that these operators are very expressive, we do not claim that they are complete. However, our framework is general enough that it would be easy for a practitioner to augment our system with a custom domain-specific operator.

3.4.1 A Motivating Domain in Entomology.

To ground our examples in reality, we searched for a single domain that could illustrate our proposed operators. Entomology offers such a possibility. Phytophagous (plant eating) insects such as the Asian citrus psyllid (*Diaphorina citri*) can have their behavior monitored by a device called an electrical Penetration Graph (EPG). Datasets collected this way are routinely searched for anomalies, which may be novel insect behaviors, or data artifacts that must be excluded from analysis.

Normally we envision the training data for a TSAD algorithm as being a single long time from which subsequences are sampled [55, 98, 100]. However, for simplicity, in Figure 3.5 we show just a curated (by entomologist, Dr. Kerry Mauck) set of subsequences.

In our training data, the mean z-normalized Euclidean distance between an instance and its nearest neighbor is 2.105, with a standard deviation of 0.114. Thus, we can define as anomalous, any time series that is not within $\mu + 3\sigma$ (2.448) of at least one subsequence. This is the anomaly threshold shown in Figure 3.5, just below the dendrogram. This may seem like a trivial TSAD algorithm, but it is essentially equivalent to the Matrix Profile algorithm, which is considered among the state of the art (SOTA) for time series anomaly detection [55]. In any case, recall that our aims here are completely independent of the TSAD algorithm used.

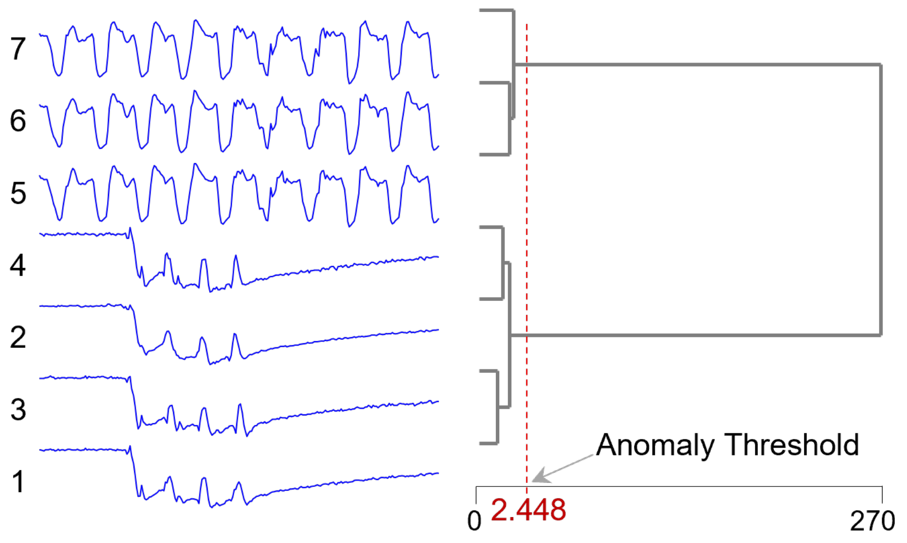


Figure 3.5: A set of subsequences which act as a training dataset for our simple TSAD model. Note that the dataset [107] is polymorphic, there are two types of normal behaviors, known to entomologists as G (1 to 4) and E2 (5 to 7).

Below, we discuss the set of operators in our framework. For brevity we mostly discuss them at a conceptual level. However, in every case, we have both detailed pseudocode and actual code available at [6].

3.4.2 Operator: Uniform Rescaling

The first anomaly we consider is the anomaly denoted in red, in Figure 3.6. It is not obvious why this is an anomaly, as it strongly resembles patterns 5 to 7 in the training set. However, it is 24.57 from its nearest neighbor in the training data, which is much larger than the anomaly threshold.

In Figure 3.6.*left* we show the effect of taking this anomaly, "stretching" it in the time axis, and (re)comparing it to pattern 7. In order to make the stretched versions of the anomaly commensurate with the fixed length training data, we must truncate off the

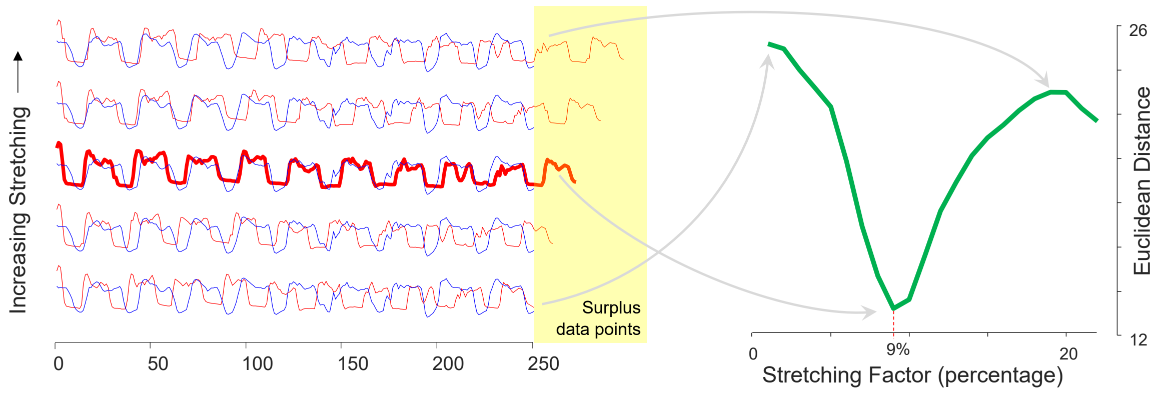


Figure 3.6: *left*) The anomaly (red) is far from its nearest neighbor (blue) because it is faster (has higher periodicity) than its nearest neighbor. By "stretching" it and recomputing the distance, we find it closely matches normal data if it is nine percent longer (*right*).

surplus datapoints (highlighted in yellow). In Figure 3.6.*right* we show the effect that the rescaling has on the Euclidean distance. As we increase the stretching, the distance begins to drop dramatically, minimizing when the anomaly is 109% of its original length, and then rising sharply again. We can interpret this as an explanation in our proposed format:

```
Anomaly would be like 6, except for corruption +9% uniform rescaling.
```

In Figure 3.6 we only consider stretching the anomaly, however we also shrink the anomaly, by reversing the roles in the computation. There is an important caveat to this operator: we must place some limits on the amount of rescaling. When any time series stretched arbitrarily long, it effectively becomes a straight line, and straight lines have a surprisingly low distance to any time series [19]. Without loss of generality, we limit the rescaling to between 80% to 120% of the original length, but this can be tweaked to fit the user's needs.

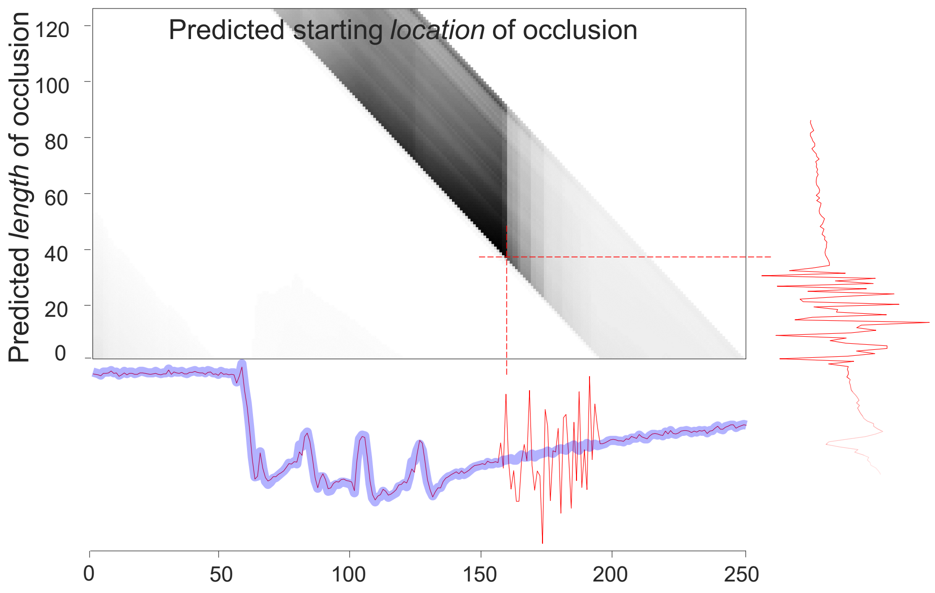


Figure 3.7: The algorithm outlined in Algorithm 4 creates a heat map whose darkest value predicts the length and location of the subsequence we should ignore to maximize its similarity to a normal data object.

3.4.3 Operator: Occlusion

Anomalies like that in Figure 3.7 are very common in many domains. The overall shape is familiar, but there is a *local* burst of noise or other distortion. This has a natural explanation in our framework, the anomaly would be like a normal object if we ignored a sub-region of some *length*, starting at some *location*. This means that we need to have some technique to predict the appropriate sub-region to occlude. It is clear we must charge some penalty for any data we ignore, otherwise we can always reduce the distance between any pair of time series by ignoring some data.

As shown in Figure 3.7 we can conduct a grid search over all possible values for location and length, scoring each combination. For our sample dataset, there is a strong peak (the darkest pixels) at location=157 and length=37, which seems to be the correct answer. We now can interpret this in our proposed format:

`Anomaly would be like 3, except for corruption region of length 37, at location 157.`

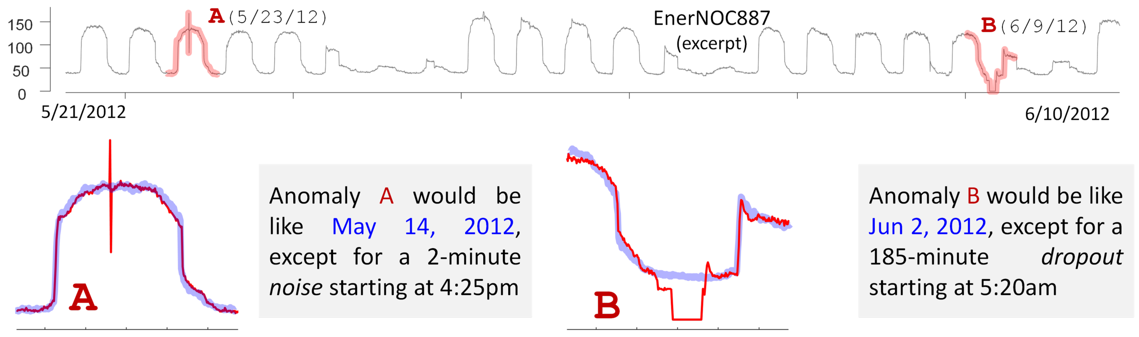
We can further expand our explanation, by noting that there are three common cases for the occluded data. If the mean of the occluded data is about the same as the mean of the remaining data (as in the above case), we can echo `noise`, if the mean is at least one standard deviation greater than the mean of the remaining data, we can echo `spike`, and if the mean is at least one standard deviation less than the mean of the remaining data, we can echo `dropout`.

In Algorithm 4, we formalize the algorithm that conducts the grid search to find the best occlusion explanation. Lines 1-3 initialize relevant constants and the data structures that the results will be stored in. Then in Lines 4-16 we compute and store the Occlusion Euclidean distance (OED) in the 2D distance matrix. For any occlusion length and occlusion location, the OED is computed in lines 7-13. In lines 7-9, The ED distance between the model series T and anomaly series A for the subsequences before and after the occluded subsequence. The core logic of the operator is in lines 10-12.

Framing this as a counterfactual, we ask, "*If the anomalous region were to instead resemble a normal signal instead, what would the distance realistically be?*" Following this line of thought, we substitute the distance of each time step in the anomalous region with the

Algorithm 4 Occlusion Euclidean Distance.

```
1: procedure OED($D, K$)    ▷ Anomaly-free Series $T_N$ and Anomaly Series $A$
2:     $L \leftarrow \texttt{len}(N_T)$
3:     $dists \leftarrow Inf(L, L/2)$
4:     $OCCL\_SCALE \leftarrow linspace(0, 2, L/2)$
5:     for $olen = 1 : L/2$ do    ▷ Test lengths
6:         for $oloc = 1 : L - olen$ do    ▷ Test locations
7:             $b, a \leftarrow oloc - 1, oloc + olen$
8:             $before \leftarrow ED(T_N[1 : b] - A[1 : b]$
9:             $after \leftarrow ED(T_N[a : end] - A[a : end]$
10:            $O1 \leftarrow abs(mean(T_N[1 : b] - A[1 : b]))$    ▷ Mean dist. between each time step.
11:            $O2 \leftarrow abs(mean(T_N[a : end] - A[a : end]))$
12:            $O \leftarrow ((O1 + O2) * 0.5) * olen$
13:            $dists[oloc, olen] \leftarrow before + after + O + OCCL\_SCALE[olen]$
14:        end for
15:    end for
16:    $oed \leftarrow min(dists(:))$
17:    return $oloc, olen, oed$
18: end procedure
```



Anomaly A would be like May 14, 2012, except for a 2-minute *noise* starting at 4:25pm

Anomaly B would be like Jun 2, 2012, except for a 185-minute *dropout* starting at 5:20am

Figure 3.8: *top*) A three-week long snippet of EnerNOC887 has two anomalies. DAMP found them, and PUPAE (*bottom*) explains them. Note that `A` is reported as noise, as the spike is followed by a dropout of about the same magnitude, giving the offending region a mean value that is similar to the original data.

average distance between the model and anomaly series where the signal indicates typical behavior. The penalty term based on `OCCL_SCALE` (line 13) is to prevent the algorithm from choosing to occlude as much of the series as possible.

There is another more general thing we can do to enhance the intuitiveness of our explanations. In some cases, instead of reporting the *relative* location of the anomalous subregions, we can report the absolute time. This may give the user additional insights. For example, suppose that a UK electrical grid operator noticed an anomalous `spike` at 3:45pm on 18/12/2022. She may search her mind for an explanation, "What could have caused this? Ahh, it was half time in the World Cup final!" (It has long been known that in England there is a large spike in electrical power demand during the half time of games, as millions of households simultaneously turn on their kettles [85]).

To illustrate both ideas, we applied the DAMP algorithm [55] to EnerNOC887, a year-long energy usage dataset for an industrial site [8]. DAMP easily found the true anomalies, and as Figure 3.8 shows, two of them happen to be in the same month.

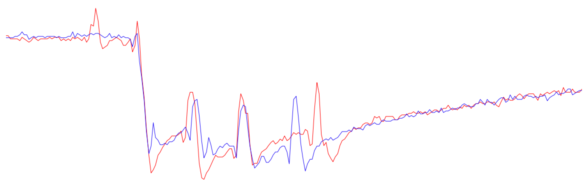


Figure 3.9: An anomaly with warping (red) and its nearest anomaly-free neighbor (blue).

3.4.4 Operator: Warping

Consider the anomaly in Figure 3.9, once again is not obvious why this is an anomaly, as it appears to resemble patterns 1 to 4 in the training set. However, if we overlay it with its nearest neighbor in the training data (exemplar 3, shown in blue), we can see that some of the peaks are out of phase.

The reader will appreciate that this misalignment of features is called informally called "warping", and dynamic time warping (DTW) is a distance measure that is designed to be invariant to this. In particular, the anomaly has a Euclidean distance of 4.2 to its nearest neighbor in the training set, and this is reduced to 2.3 under the DTW distance. Note that for any two time series, their DTW distance is always less than or equal to their Euclidean distance. Thus, just as with Occlusion operator and OED, a penalty distance needs to be incorporated into the DTW distance if it is to be compared to the ED of the other operators.

Where A is the anomaly subsequence, T_N is its nearest anomaly-free neighbor, L_A the length of the anomaly, A' the warping path, and $L_{A'}$ the length of the warping path, compute and set $I_{warping}$ as follows:

$$I_{warping} = \frac{DTW(A, T_N) * (\frac{L_A}{|L_{A'} - L_A|})}{ED(A, T_N)} \tag{3.1}$$

As discussed in more detail below, a greater reduction in distance relative to the Euclidean distance between the anomaly and its nearest neighbor will semantically indicate that the operator is a plausible explanation for the anomaly. Scaling $DTW(A, T_N)$ according to the reciprocal of difference in length according to the warping path allows PUPAE to communicate that anomalies fully utilizing the DTW warping window are more likely to be appropriately explained as a *warped* anomaly.

While we recognize this is subject to an edge case when the lengths of L_A and $L_{A'}$ are of equal length, but the reader will appreciate that this particular edge case of a warping path of length zero is semantically Euclidean distance, and this is easy accounted for before computing $I_{warping}$.

3.4.5 Operator: Smoothing

In Figure 3.10, there are various anomaly types and an operator applied to each. In the left column by row, various anomalies are in red overlaid with their nearest anomaly-free neighbor in blue. In the right column by row, the same anomalies with the corresponding perturbing operator applied to minimize the resulting time series' Euclidean distance to the nearest anomaly-free neighbor.

Consider the noisy anomaly (top) in Figure 3.10. It is easy to see that it resembles patterns 1 to 4 in the training set but is simply *globally* noisy (as opposed to the example in Figure 3.10 which had *local* noise). The anomaly has a Euclidean distance of 3.0 to its nearest neighbor in the training set, and this is reduced to 2.2 after it is smoothed using the simple moving mean algorithm with a window length of three.

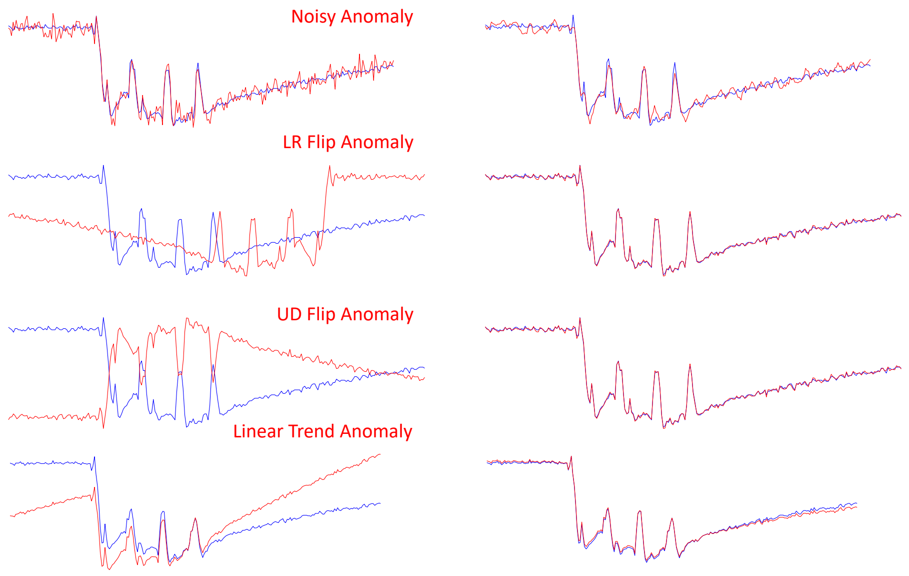


Figure 3.10: *Left column*) By row, various anomalies are in red overlaid with their nearest anomaly-free neighbor in blue. *Right column*) The Noisy Anomaly has been smoothed, the LR Flip Anomaly has been horizontally flipped, the UD Flip anomaly has been vertically flipped, and the Linear Trend Anomaly has been detrended.

We have found that such anomalies are important to be able to explain, as their meaning is highly domain dependent and actionable. For example, in many medical contexts such noise is virtually always due to "Power Line Interference" [18], and uninteresting to the clinicians. In fact, it is often cited as the number one source of *false positive fatigue* [80]. In contrast, in some oil&gas operations, short bursts of high frequency noise can be caused by cavitation due to large pressure drops. Such anomalies need to be prioritized for investigation, as they are often precursors to dramatic failures. While there is significant literature on smoothing algorithms, we have found a simple moving average filter seems to be universally useful here.

3.4.6 Operator: Reversal (LR Flip)

We synthetically created the LR Flip Anomaly in Figure 3.10 to remain consistent with our insect dataset. However, these reversed shapes do appear as anomalies in the wild [108]. The anomaly has a Euclidean distance of 17.6 to its nearest neighbor in the training set, and this is dramatically reduced to 0.54 after it is flipped horizontally.

3.4.7 Operator: Flip Upside Down (UD Flip)

This UD Flip example in Figure 3.10 is also a synthetically created anomaly to remain consistent with our insect theme. However, once again these Y-axis reversed shapes do appear in the wild as anomalies [108]. The anomaly has a Euclidean distance of 23.27 to its nearest neighbor in the training set. This is reduced to 1.95 after it is flipped vertically.

3.4.8 Operator: Linear Trend

Consider the Linear Trend anomaly in Figure 3.10. As with some of the previous anomalies, it resembles patterns 1 through 4 in the training set. However, with careful visual inspection the reader may realize that it has a slightly greater upward linear trend. There are standard algorithms for detrending time series, which are logically equivalent to *finding* the best fitting line to the data, then subtracting that line from the data. Before we apply this correction to the above, the distance to its nearest neighbor in the training set is 8.7, afterwards the distance is reduced to just 0.93.

This is another example of an anomaly explanation with clear actionability. In patient monitoring in an ICU setting, a change in linear trend of a *PPG signal* (oxygen saturation) is usually ignorable. It is typically due to patient movement. However, a change

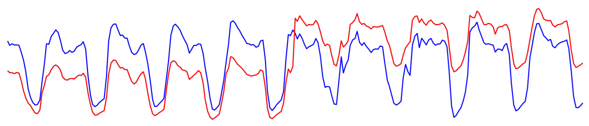


Figure 3.11: A mean-shifted anomaly, also commonly referred to as a *step* anomaly.

in the linear trend of a respiration signal may indicate dyspnea (difficulty breathing) caused by upper airway obstruction and should be attended to immediately.

3.4.9 Operator: Piecewise Normalization

Consider the anomaly in Figure 3.11, which we show overlayed with a normal time series (in blue) for context. The reader will appreciate the two time series look very much alike, except that the anomaly has a “level-change” or “shift” in its middle. Such anomalies seem to be very common across multiple domains. The change can be a mean change (as in the above, and/or an amplitude change.

The cause of such changes is domain dependent. In our insect EPG example, there is a situation that is known to cause this anomaly. The insect being monitored has a fine uninsulated gold wire (about 1/100th the thickness of a human hair) glued to it and the insect is allowed to roam around on, and feed on a plant. If that wire should touch a part of the plant, it reduces the electrical resistance, and causes a level shift. In the example shown above, the change point occurs in the middle of the time series, however we cannot be sure that this is generally true, especially as we wish to be TSAD algorithm independent. Thus, we will need to test all possible split points.

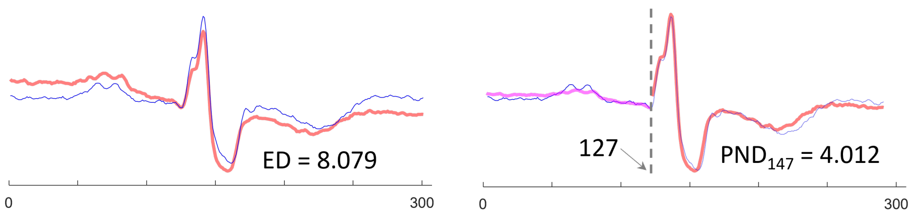


Figure 3.12: *left*) An apparent anomaly from the MITDB [62] benchmark overlaid on its nearest neighbor. *right*) The PND operator suggests cutting the anomaly at 127 to produce a dramatically reduced distance, and to visually reveal the latent similarity.

This anomaly can be corrected by piecewise normalization. In particular, for two time series A, and T_N, both of length m, we can compute the Piecewise Normalized Distance:

$$PND(A, T_N) = arg \min_{\frac{m}{3} \leq i \leq m - \frac{m}{3}} (\sqrt{ssd(A_{1:i}, B_{1:i}) + ssd(A_{i+1:m}, B_{i+1:m})}) \tag{3.2}$$

Where ssd is the sum of squared distances between two z-normalized vectors:

$$ssd(X, Y) = \sum_{j=1}^{|X|} (X_j - Y_j)^2 \tag{3.3}$$

The PND reduces to the ED if the left and right sides of both time series have the same means and standard deviations but produces a dramatically lower score in the presence of a shift of mean and/or standard deviation. In Figure 3.11 we show an example of PND applied to a section from a classic benchmark.

The time series in Figure 3.12 is not the labeled anomaly in this dataset, but is flagged as an anomaly by some algorithms including the Matrix Profile and Telemanom [41] (on some runs; Telemanom is stochastic). This hints at the value of time series explanations. If a user is experiencing many false positives, but they all have the same explanation, she can suppress the reports that have that explanation, or refine her algorithm.

3.4.10 Discussion of Operators

As we will show in our experimental evaluation section, the list of operators above is very expressive, and covers the vast majority of anomalies available in the communities' benchmark datasets [98, 100]. However, we emphasize that we are not claiming that these are a *complete* set. We believe that our framework is general enough to allow practitioners to add custom operators specialized for their domain.

PUPAE only allows a single operator to be an explanation. The reader may argue that an anomaly may have two or more causes, say a spike and some warping. Again, based on the standard benchmark datasets [98, 100] and our discussion with domain experts, this seems very rare, thus we defer further discussion of this to future work.

In addition to adding new bespoke operators, occasionally a user may wish to *remove* operators based on their domain knowledge. For example, they may know that their domain has a lot of inconsequential warping, thus warping should not be used to explain an anomaly.

3.4.11 Making Operators Commensurate

In the previous section, we listed the full set of operators we consider in this work. However, the operators need to be commensurate so that we can compare and rank them, and only report the most plausible explanation to the user. All our proposed operators return distances that are either ED or based on ED (note that DTW is simply the Euclidean distance on the "dewarped" time series). However, this does not mean that they are commensurate. For example, the occlusion operator may be reporting a distance based on only 50% of the data, whereas the reversal operator considers all the data. For all operators we define a scalar I (Improvement), which is simply the ratio of the new, post-operator distance divided by the original distance. For example, for an anomaly A, and its nearest neighbor T_N in the training data we define:

$$I_{LinearTrend} = \frac{ED(A', T_N)}{ED(A, T_N)} \tag{3.4}$$

Where in this example, A' is the transformed anomaly parameterized by the amount of linear trend to add to A such that $ED(A'_{LinearTrend}, T_N)$ is minimized. The other operators are defined similarly, and other operator specifics are noted in Section 3.4. Note that there is no possibility of a division-by-zero error, as the operators are only used on time series that have a distance to their nearest neighbor that greater than some positive anomaly threshold. We note that I score values can be greater than one, for example, I_{LRFlip} can be much greater than one.

When selecting an explanation, the reader will appreciate that the method is rather simple and interpretable. For each operator, compute the $ED(A', T_N)$ for all for all combinations of values its parameters might take on. PUPAE reports the transformation (and any parameters) that minimize the $ED(A', T_N)$, as a lower I score indicates an operator's success. This is formalized in Section 3.4.12. While the idea of conducting a search may seem computationally daunting at first glimpse, as aforementioned, only Occlusion has more than one parameter. Additionally, due to the semantic nature of these operators, the search space of each parameterized operator is further restricted.

For example, Piecewise Normalization communicates that, primarily, normalizing two subsequences piecewise will yield the greatest reduction in ED whilst being the most systematic change. This indicates that if PUPAE's best attempt at using Piecewise Norm to explain an anomaly results in normalizing a subsequence of small length separately, piecewise normalization is unlikely to be the most proximate explanation.

3.4.12 A formal definition for PUPAE

With our operators defined and the intuitive understanding of Improvement scores discussed, we can formally define the anomaly explanation that PUPAE provides. Given an anomaly-free time series T_N, an anomaly time series A, where $|T_N| = |A|$, and a set of operators $\mathcal{F} = \{f_0(\theta), f_1(\theta), ..., f_n(\theta)\}$:

$$f^*_\theta = arg \min_{f_{i,\theta} \in \mathcal{F}} (\min \frac{ED(\hat{A}, T)}{ED(A, T)}), \hat{A} = f_\theta(A) \tag{3.5}$$

The input and output are mappable to the natural language counterfactual explanation template in Section 3.1:

A `would be like` T_N`, except for corruption` $-f^*_\theta$`.`

Where f^*_θ is an optimal transformation and $-f^*_\theta$ is the perturbation for which it reverses. In the cases of parameter-free operators (e.g. LR Flip, UD Flip), there is no sign.

3.4.13 Selecting Operators to Include in $\mathcal{F}$

While we assert that the operators discussed in this section are a comprehensive base set of operators, the operators in $\mathcal{F}$ should be selected with domain expertise, adding and removing operators as the user sees fit. For example, in medical telemetry, changes in the linear trend may be considered a "wandering baseline" and are medically irrelevant. However, in industrial processes, a change in linear trend can be significant, and may indicate a batch process is “running hot”. However, for simplicity, and to stress test our ideas, we do not do domain customization in our experiments.

3.5 Experimental Evaluation

To ensure that our experiments are reproducible, we have built a website [6] which contains all data and code for the results. We have documented concrete details of our experiments to allow for reproduction of all our experiments. All experiments were conducted single-threaded on an Intel® Core i7-10710U CPU at 1.10GHz with 16 GB of memory. We provide two types of empirical evaluation.

- Case studies that are designed to test our method on real world datasets. By nature, these studies are mostly anecdotal. However, in every case we work with a domain expert to help interpret our findings.

- Large scale experiments on corrupted datasets. Here, to avoid the conflict of interest of creating new corruption models to test our ideas, we work only with existing corruption models specially designed for TSAD.

The time needed to compute all our operators is inconsequential compared to the time needed for the original anomaly detection [55]. hus, our evaluation focuses on the following questions: where objective ground truth exists, does PUPAE correctly discover it? And, in the absence of ground truth, is the explanation plausible to a domain expert? Without loss of generality, expect where otherwise stated, we use the leftMatrix Profile to discover the anomalies [55]. However, we remind the reader that PUPAE is completely independent of the algorithm used to do the actual TSAD.

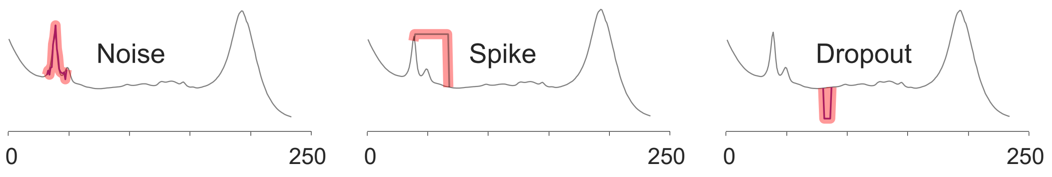


Figure 3.13: An exemplar from the UCR Wine [27] dataset, with three types of synthetic anomalies (highlighted in red) added.

3.5.1 Testing the Occlusion Operator

Before testing and demonstrating the PUPAE framework, we will begin by testing Occlusion in isolation. The occlusion operator needs to predict the *location* and *length* of a subregion to delete from a time series to make match the ground truth of a distortion (noise, spike, dropout) that some anomalous process added to otherwise normal data. In the absence of large amounts of data for which the ground truth is known, we created synthetic examples by corrupting exemplars from the UCR archive [27].

We randomly select an exemplar A, from a UCR dataset, we then randomly select a value for location from 1 to $\frac{1}{2}m$, and a value for length from 0 to $\frac{1}{4}m$. Note that the length of the distortion could be zero, and we would hope that our algorithm predicts that. We then choose randomly, with equal probably from the following three distortions: add *noise*, add a *spike*, add a *dropout*. Figure 3.13 shows some examples in the Wine dataset.

Can we correctly recover the location and length of these distortions? We find the exemplar's (non-self) nearest neighbor T_N under Euclidean distance in the same dataset and use the operator in Section 3.4.3 to predict the *location* and *length*. Figure 3.14 shows the results of 1,000 such tests. The histogram bin sizes in Figure 3.14 are set to be as granular as a single datapoint to best communicate the precision and accuracy of the Occlusion

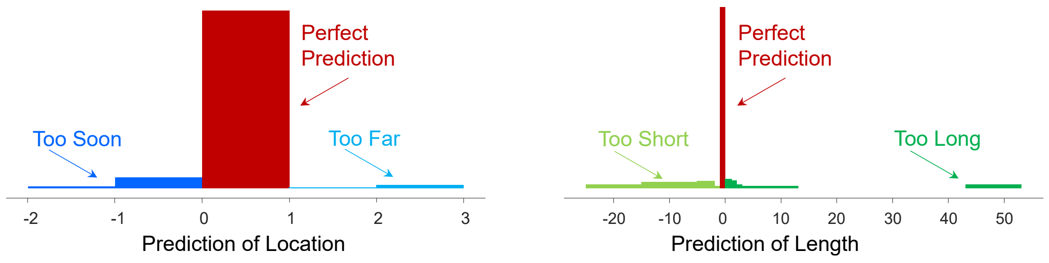


Figure 3.14: A benchmark of predictions created from the Wine dataset [27], where one of three anomalies of random lengths are inserted into exemplars at random locations. From there, the Occlusion operator is run on each anomaly series and the source dataset, excluding the donor series the anomaly was inserted in. PUPAE selects the nearest neighbor to compare the anomaly to, computing a location and length of an occlusion to apply.

operator's success at locating the location and length of the anomaly. Note that in almost all cases, PUPAE is able to correctly locate the anomaly (spike, dropout, or noisy subsequence) location and length. Perfect results are probably not possible, due to natural anomalies within the data.

3.5.2 Multiple Short Case Studies

To show the generality of PUPAE, here we will show some brief examples on diverse real datasets. In every case, the reader can find detailed provenance at [6].

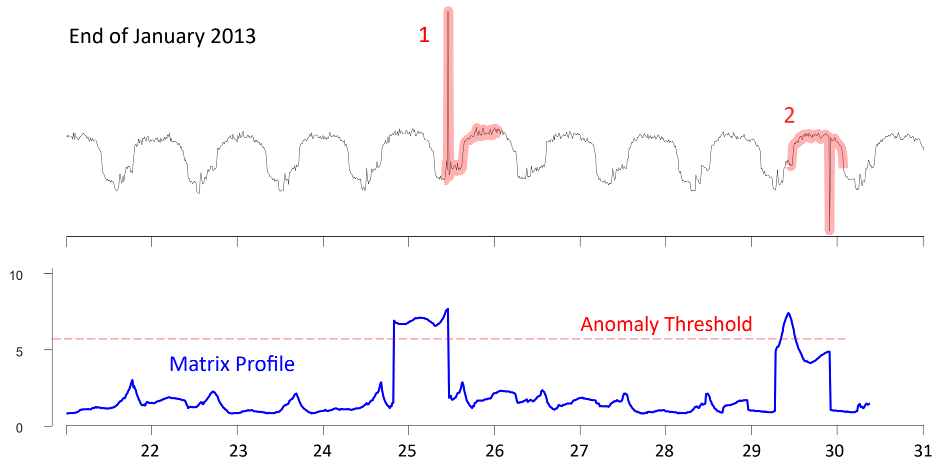


Figure 3.15: A snippet of a four-year electrical demand dataset happens to have two (easy to find) anomalies in the same week.

UCI Power Demand

The power demand dataset from UCI [11] has several annotated anomalies. As Figure 3.15, two of them happen to be in the same week and can be found by the Matrix Profile. PUPAE described these as:

- `Anomaly 1 would be like 21/01/2013, except for Spike at 4:30 AM.`
- `Anomaly 2 would be like 29/01/2013, except for Dropout at 7:45 PM.`

Here PUPAE's distinctions between "dropout" and "spike" may be actionable. A line engineer may prioritize the investigation of spikes (can be caused by lightning strikes or upstream tripped circuit breakers) over dropouts (typically caused by a faulty sensor, not a real change in power demand).

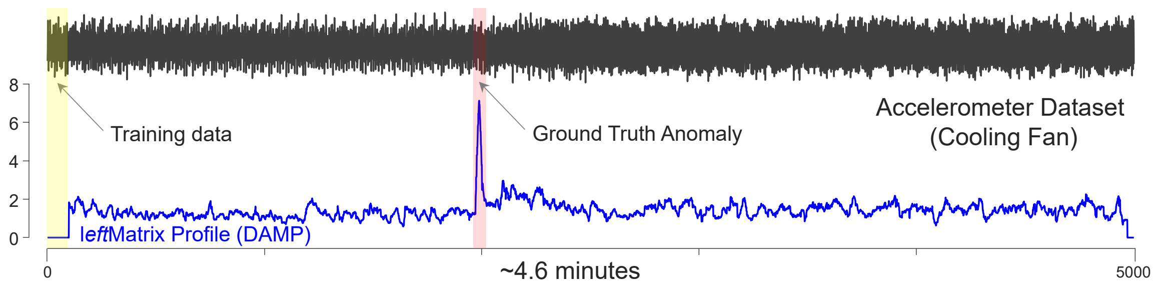


Figure 3.16: *top*) A five-minute snippet from the accelerometer dataset. *bottom*) DAMP correctly finds the anomaly, but what was its cause?

Cooling Fan Accelerometer

This accelerometer dataset measures the health of a cooling fan [36]. By manipulating magnets around the fan (whose blade was magnetic), the creators were able to induce various types of anomalies. The original authors performed some data transformations and created a sophisticated domain-informed neural network then discover these anomalies. However, as shown in Figure 3.16 we found that a direct application of DAMP on original data could find the anomalies.

The cause of the anomaly is not obvious, even with a careful visual inspection of the data, however PUPAE correctly states:

```
Anomaly would be like data at 70.2 s, except for -9.1% uniform rescaling.
```

The anomaly reflects a sudden braking effect to the motor.

3.5.3 Amazon's Benchmark

A recently published paper by a research group in Amazon [36] includes a sophisticated tool for generating plausible synthetic anomalies. In Figure 3.17 we show their plot

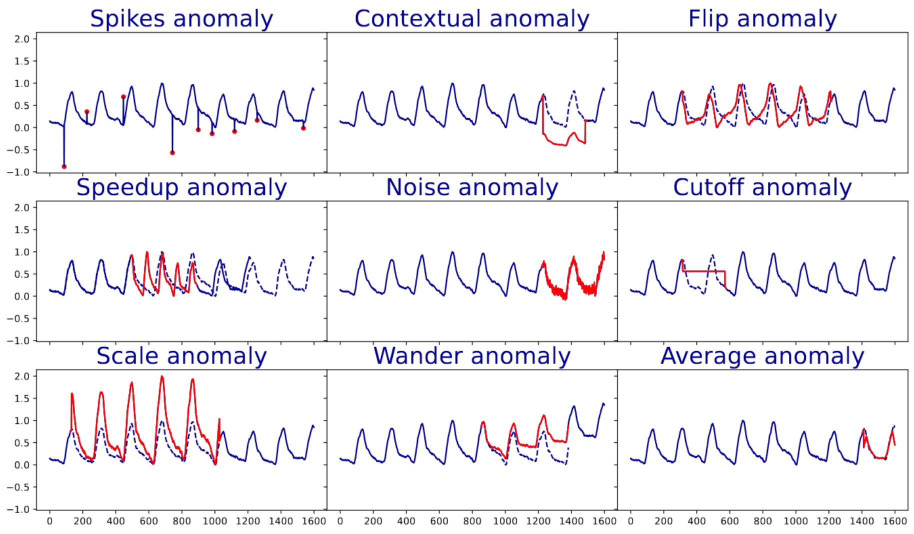


Figure 3.17: A screen capture of a figure from [36], showing examples of nine types of synthetic anomalies. The reader will appreciate that almost all the listed anomalies have counterparts in our operator set.

introducing this tool. The authors argue (in a similar vein to our discussion of which operators to use in PUPAE) that the majority real-world anomalies are well modeled by these nine cases. The reader will appreciate that this model offers the opportunity to evaluate our ideas. We propose to corrupt anomaly-free data using a random choice of one of the above corruption models, and then test our ability to first *discover*, then *explain* the anomaly.

While Amazon's vision mostly overlaps with ours, before proceeding, we need to do a little (re)mapping between their terminology/model and our own, and slightly extend Amazon's model to include additional distortions. In particular:

- We limit Amazon's *spikes* to a single *spike*.
- Warped anomalies are added as an extra corruption.
- Flip Upside Down (UD Flip) are added as a corruption.

We omit a few of Amazon's distortions:

- Both *Contextual* and *Average* anomalies are polymorphic, and one of several different operators might be a solution for any instance, confounding our evaluation metric.
- *Cutoff* anomalies may be solved by an occlusion (as demonstrated in Section 3.5.1). Moreover, they are trivial anomalies in the sense noted by Wu and Keogh [100].

Finally, to clarify differences in our naming conventions:

- Amazon's *Flip* anomalies are our *LR Flip* anomalies.
- Amazon's *Speedup* maps to our *Uniform Scaling* anomalies.
- Amazon's *Noise* maps to our *Noisy* anomalies.
- Amazon's *Scale* maps to our *Step* anomalies.
- Amazon's *Wander* maps to our *Linear Trend* anomalies.

We can now describe our experiment in detail. We:

1. Randomly select a 10,000 datapoint long anomaly-free time series from a set of four: PPG, Gait, electrical demand fingerprinting [71], or conveyor belt system. To the best of our knowledge, after consulting with domain experts and examining out-of-band data, these datasets are anomaly-free.
2. With uniform probability, randomly select one of the nine anomaly types, and insert an anomaly of this type at a random location between 3,001 and 9,000. The first 3,000 datapoints are reserved for training data, and we stop at 9,000 to avoid "edge-effects" with our sliding window.

	LR Flip	**LT**	**U. Res.**	**Noisy**	**UD Flip**	**Step**	**Warp**	**Spike**
LR Flip	1198							
LT		985	7			4		
U. Res.			1200					
Noisy				1200				
UD Flip			1	24	662			
Step	25	230	182	3	26	671	5	31
Warp		38	286	164			396	23
Spike			16	156				298

Table 3.1: A confusion matrix, where the rows indicate the type of anomaly inserted and teh columns indicate PUPAE's explanation. Where there are no values listed, there were no misclassifications of that type made.

3. Using DAMP [55], with the first 3,000 datapoints as training data, attempt to find the anomaly. If the anomaly was successfully discovered, use MASS [67] to locate the anomaly's nearest neighbor in the anomaly-free training data.

4. • Use PUPAE to predict the corruption producing the anomaly.

Our benchmark has 300 exemplars of each anomaly type per dataset, totaling 2400 anomalies sourced from each dataset, with a benchmark size of 9,600 time series. Table 3.1 summarizes the results of PUPAE's ability to correctly identify the anomaly inserted, based on the benchmark formulation described above. Of the 9,600 trials, DAMP successfully discovered the anomaly 7,831 times. Confusion matrices separating this benchmark by dataset are available on our companion website at [6].

The results show PUPAE correctly explains the inserted anomalies with a mean per-dataset accuracy of 84.25%, and a general accuracy of 84.41%. Out of the 15.59% of misclassified exemplars, 3.65% of those misclassifications were correcting warped anomalies using uniform scaling. This is understandable, as it is known that a special case of DTW

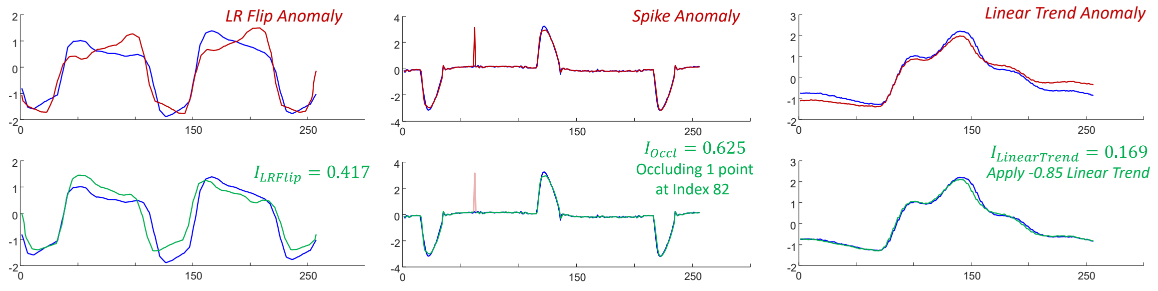


Figure 3.18: The domains from left to right are insect gait, electrical demand fingerprinting, and PPG. *Top*) The nearest anomaly-free ED neighbor (blue) to the anomaly, and the anomaly (red). *Bottom*) The nearest anomaly-free ED neighbor (blue) to the anomaly, and the transformed anomaly (green).

(open-ended DTW) is essentially equivalent to uniform scaling. The second most common misclassification was a Step anomaly remedied by applying some amount of downward linear trend. Oftentimes, if the step anomaly is a higher mean in the latter half of the anomalous subsequence, applying some downward linear trend will produce a similar effect as piecewise normalization. These accounted for 2.9% of the errors.

PUPAE can explain the vast majority of anomalies in the benchmark successfully. We show three random examples of successful predictions on the generated benchmark dataset in Figure 3.18. The transformed anomaly is transformed according to the PUPAE, according to Equation 3.5.

For brevity, we relegate other examinations of misclassifications and details about DAMP's success w.r.t. each anomaly class and domain to be housed at [6].

3.6 Conclusions and Future Work

We have shown a simple framework for explaining anomalies. We have shown both significant anecdotal evidence that it works and demonstrate its performance on a large-scale benchmark data based on an Amazon anomaly generator [36]. Beyond the direct utility of explaining anomalies, we believe our ideas can be used to improve the user experience of using any TSAD algorithm. One the most cited problems with deployments of TSAD algorithm is false alarm fatigue [80]. If it is noted that most false alarms have similar explanations, that observation could be exploited to improve the algorithms. We leave such explorations for future work. Finally, we have created a new benchmark of 9,600 time series with ground truth annotations. We believe that that this dataset will help others to research the task-at-hand at [6].

Chapter 4

Time Series Feature-based Clustering with Active Learning

4.1 Introduction

Time series clustering is one of the most basic tools used by the data mining community. While it is sometimes the sole task for the analyst, it is more often a preliminary step for further downstream tasks, such as forecasting, fraud detection, classification, anomaly detection, etc. As such, it is a critical task, as the success of the downstream task clearly hinges on the quality of the clustering. Unsurprisingly, there has been significant work on time series clustering, however, most of it assumes that semantically similar behaviors are manifest in the conservation of the shape of the time series. This assumption seems to be at least partly attributable to an overreliance on the UCR Archive [27]. While the UCR Archive has doubtless been an excellent resource for the community, at least some of the

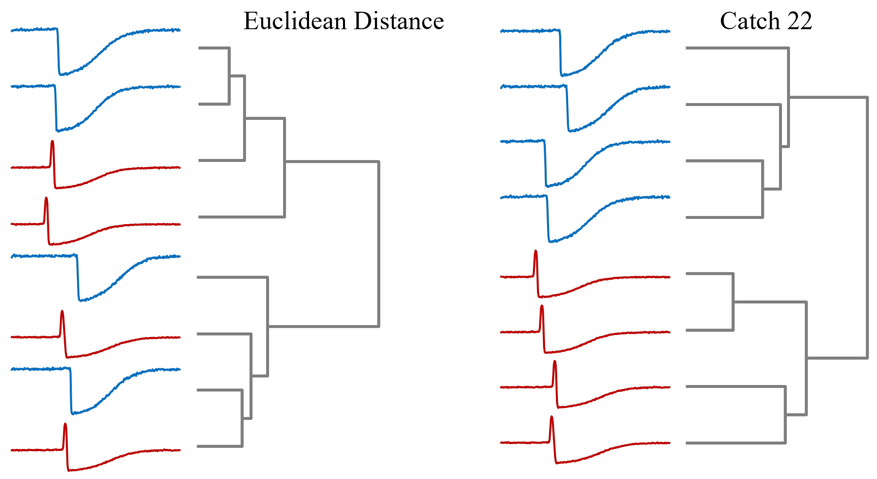


Figure 4.1: A subset of the UCR Trace [27] dataset clustered using Euclidean Distance *(left)* and Catch22 [57] features *(right)*.

datasets were built using the Euclidean distance (ED) to edit and label similar examples. As such, it is unsurprising that shape-based clustering would work well.

However, when we expand our interests to more general problems, shape-based clustering algorithms can fail to capture obviously conserved structure. For example, consider the clustering shown in Figure 4.1. Here, shape-based Euclidean distance fails to produce the correct (and visually obvious) clustering. However, a common feature-based method originally developed for classification, catch22 (C22) [57], can correctly cluster the data. Before continuing, we briefly note that we actually use C24, which is C22 augmented with two additional features. In this simple example, using all the C24 features works well. However, on more challenging datasets, we generally need to use a domain dependent subset of clusters to obtain better clusterings. This seems to offer an insurmountable challenge, as there are $2^{24} - 1$ subsets to consider, and we do not have labels to guide our search. There is a potential solution that exploits active learning. Assuming a user could rank clusterings (even just ranking pairs of clusterings), we could use their rankings to guide the search for an effective feature subset. This idea seems to require an untenable effort of our user, however as we will show, we can create a function to first filter unpromising clusterings, and then only task our user to make a handful of choices. As we will show, this simple idea allows us to produce clusterings which outperform the current state-of-the-art in time series clustering efforts.

The rest of this paper is organized as follows. In Section 4.2 we motivate the need for a new algorithm in this saturated research area. Section 4.3 contains the definitions and notation, and Section 4.4 discusses related work, and we introduce our algorithm in Section 4.5. In Section 4.6 we present a detailed empirical evaluation of our ideas.

4.2 Motivation

Before continuing, we need to answer the following question. In such a crowded research area, why do we believe that clustering based on C24 subsets could be so fruitful? Consider Figure 4.2, which shows the performance of four highly cited distance measures for time series clustering [57, 65], evaluated on the UCR GunPointMaleVersusFemale dataset [27]. Here we see that the relative ranking of the three shape-based measures happens to be K-Shape [65] $>$ DTW $>$ ED. This is not always the case for every datasets; it depends on whether the invariance to phase (K-Shape) or minor local misalignments (DTW) help or hurt the domain in question.

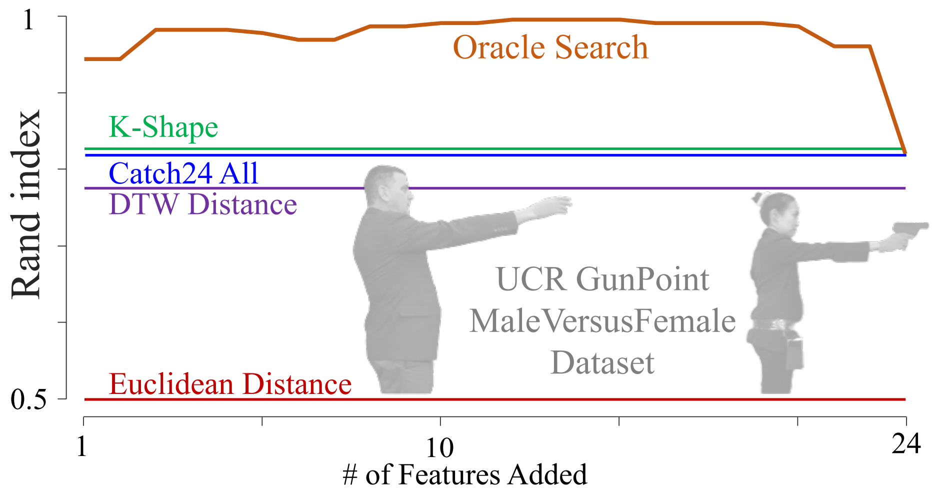


Figure 4.2: The performance of greedy Oracle search of C24 features.

Note that in this case, using *all* of the C24 features also produces relatively good results, just slightly worse than K-Shape. However, it is natural to ask if a smaller, domain appropriate subset of C24 features could improve things? By definition, in the real world we do not have access to class labels while clustering, however let us consider an experiment that provides us with the upper bound for the performance of C24 greedy feature selection for clustering.

In Figure 4.2 we also show the results of performing a forward greedy *Oracle Search*, using the Rand index (RI) to guide the search. To be clear, this is "cheating"; naturally in real deployments we will not have access to the class labels needed to compute the Rand index. Using only two features (features 3 and 21) the algorithm can achieve a Rand index of 0.943, completely dwarfing all the other approaches, and with a few more features the Rand index peaks at a near perfect 0.996. Not all the datasets show the same results, but a remarkable number of them do [7]. This result offers motivation for our proposed work. If we could even approximate this Rand index driven feature search, we could potentially dramatically improve the accuracy of clustering of time series, at least for some datasets. **Key Insight**: Approximating the Rand index with a heuristic is challenging, and using human annotations to approximate the Rand index is possible but tedious and (human) time consuming. However, we will show that by using a combination of a heuristic and a modicum of human effort, we can produce accurate clusterings.

4.3 Definitions and Background

We are interested in clustering, not classification, but we will use the class labels in a post-hoc evaluation of success. Given two clusterings, each pair of time series will have some cluster assignment in Clustering A and Clustering B. The Rand index is a measure of similarity between A and B, and is the ratio of *pairs* of time series with the same cluster assignments normalized to the total number of time series pairs clustered.

Definition 1. Let there be a time series dataset $T_D = \{t_1, t_2, ..., t_N\}$, there are N time series, the Nth time series denoted t_N. Let there be two clusterings of $T : C_1 = \{X_1, X_2, ..., X_p\}$, where $X_{1:p}$ denote the cluster labels, and $C_2 = \{Y_1, Y_2, ..., Y_q\}$ where $Y_{1:q}$ denote its cluster labels. Every pair of time series $(t_a, t_b) \in T_D, a \neq b$ is assigned to one of four mutually exclusive sets $\{S_1, S_2, S_3, S_4\}$, in the following way:

- $S_1 : t_a, t_b \in X_i$ and $t_a, t_b \in Y_j$ where $i \leq p, j \leq q$
- $S_2 : t_a \in X_i, t_b \in X_j, i \neq j$ and $t_a \in Y_k, t_b \in Y_l, k \neq l$ where $i, j \leq p$ and $k, l \leq q$

- $S_3 : t_a, t_b \in X_i$ and $t_a \in Y_k, t_b \in Y_l, k \neq l$ where $i \leq p$ and $k, l \leq q$
- $S_4 : t_a \in X_i, t_b \in X_j, i \neq j$ and $t_a, t_b \in Y_k$ where $i, j \leq p$ and $k \leq q$

The *Rand index* (also commonly known as Rand error) RI is described as $RI = \frac{|S_1|+|S_2|}{|S_1|+|S_2|+|S_3|+|S_4|}$.

The time series pair is placed in S_1 if they are in the same cluster in both clusterings. They are placed in S_2 if they are assigned to different clusters in both C_1 and C_2. When this is not the case, the pair of time series meets the qualifications for S_3 or S_4. Rand index values are bounded $[0, 1]$, where 0 indicates C_1 and C_2 agree on no pairs of points, and 1 indicating an identical clustering.

A limitation of the Rand index is that it still "rewards" similarities between two clusterings that occur with chance. The *Adjusted Rand index* (ARI) adjusts for this by considering the expected Rand index, and is bounded between $[-0.5, 1]$. The *Normalized Mutual Information* (NMI) score is a normalized version of the Mutual Information (MI) score, and is bounded from $[0, 1]$. The *Adjusted Mutual Information* (AMI) score is an adjusted version of MI that accounts for chance, accounting for the fact the MI score will be higher for clustering problems with several clusters [66]. The ARI will be higher when the clustering solution is balanced, and the AMI will be higher when ground truth clusters are imbalanced (and there exists small clusters) [70].

For simplicity, we present our arguments using the Rand index, but the reader will appreciate that this is a pessimistic improvement in performance, still awarding all methods some credit for pairs of time series similarly clustered together by chance. Consequently, we perform our *benchmark* experiments using the RI, ARI, NMI, and AMI scores. We present

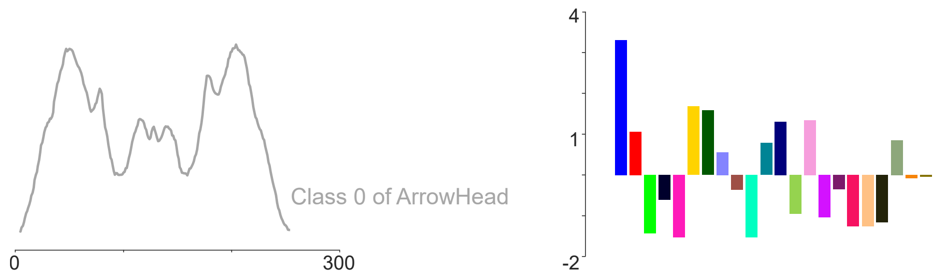


Figure 4.3: *left)* A time series from Class 0 of UCR ArrowHead [27]. *right)* A visualization of its C24 features. Note: this exemplar was z-normalized; its mean and std feature values are zero, and thus, some color bars are not visible.

our experimental results using all of these scores in Section 4.6.

Most algorithms for time series clustering directly ingest raw time series and use a distance measure such as ED or DTW to attempt to cluster the data [15, 57]. We call such algorithms *shape*-based. We propose instead to consider only features derived from these time series. In particular, we compute a feature vector of length twenty-two, using the catch22 features:

Definition 2. A C22 feature vector $F = f_1, f_2, ..., f_{22}$ is an ordered set of twenty-two real-values. When considering a training dataset composed of N time series of varying classes, we convert these N time series into N feature vectors, $F^1, F^2, ..., F^N$, each of length twenty-two.

Two simple features, *mean* and *standard deviation* are missing from the catch22 set. They make little difference to most reesearchers who evaluated their algorithms solely on the UCR Archive [27]. However, these two features can be useful in more general applications, thus many researchers have augmented C22 to C24, as we do in this work. Figure 4.3 shows a time series from the UCR ArrowHead dataset [27], and its C24 feature vectors plotted as colored bar charts.

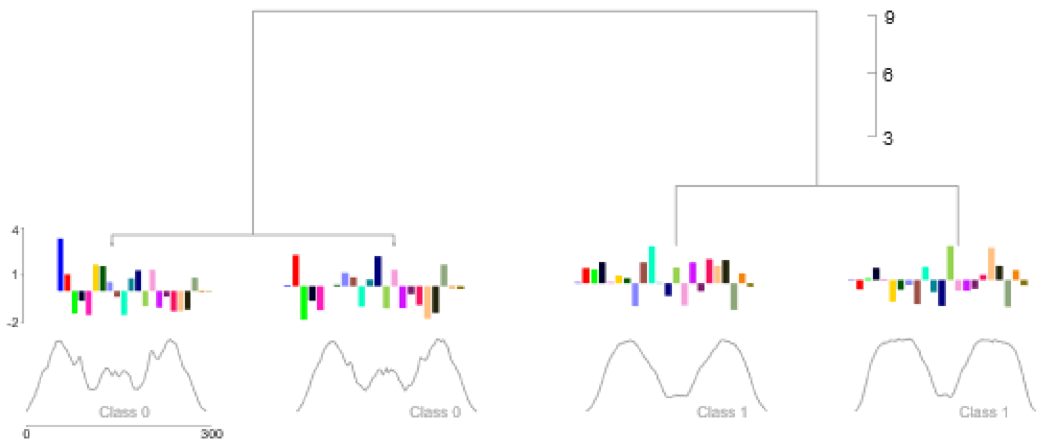


Figure 4.4: A single-linkage clustering of four exemplars from the ArrowHead dataset using the distance between their C24 vectors.

We follow the guidelines of the original authors who suggested using a standard set of colors when visualizing C24 features. If we produce C24 vectors for each exemplar in our dataset, we can use the Euclidean Distance between them to perform a clustering. In Figure 4.4, we show the hierarchical clustering of four samples. Note that in Class 0, the third feature (green) is negative, and the fourteenth feature (light pink) is positive; however, in Class 1 the opposite is true. This supports our hypothesis that C24 feature vectors are preserved within a cluster, and can offer a method of discrimination *between* clusters.

4.4 Related Work

Time series clustering is a popular are of research, and is often used in applications with further downstream tasks (and occasionally vice-versa). Clustering is used for and, has in turn, been improved by efforts in forecasting [17, 82], discord discovery [24, 49], anomaly detection [52], and fraud detection [75, 81]. The quest for a domain-agnostic method (regardless of task) is a particularly complex task when dealing with time series. For example, when time series datasets are created, the specific and full semantic meaning the conditions the data was gathered under are not always quantifiable or documented. Domain invariances are often unknown, and a dataset might contain time series with varying sampling rates across exemplars. Despite these challenges, many efforts (including those we have just referenced) have succeeded.

However, the quest for *interpretability* naturally follows. Our proposed method, CEREBRIC, claims both characteristics. While features in the original C24 work were not made intuitive, recent work [88] has made progress in this area.

Active learning has effectively been utilized in sampling strategies for time series [84], learning time series models under safety constraints [110], the classification of vibration and industrial process data [28], and anomaly detection [74, 89, 95, 105]. While these efforts have been successful, they all note that obtaining user feedback is nontrivial. Most of them ask the user to provide a label for a single time series or subsequence, such as `normal|anomaly`, or task the user with giving a `must-link|must-not-link` constraint for a single pair of time series. In our active learning approach, we propose the user choose the better clustering of two dendrograms. In this work, we argue this approach is easier for the user whilst providing richer information and context.

4.5 CEREBRIC

We call our system CEREBRIC, **C**lusterings **ERE**cted **by** **R**educed **I**nstances of **C**24. Our overall approach is to first use a heuristic to estimate the quality of a of tentative clusterings, then have a human refine this set by examining only the top-K performers. We present our arguments using the Rand index, but as we demonstrate in our experiments in Section 4.6 the RI can be replaced with ARI, NMI, AMI, or another clustering similarity score of choice. To do this, in all places we describe using RI, another similarity score may be used instead. We begin our introduction to CEREBRIC with a discussion of our heuristic.

4.5.1 Heuristic to Approximate Rand Index

We seem to have set ourselves a difficult task: to create a heuristic function that can examine tentative clusterings and predict the RI *without* access to any labels. However, our task is a little easier in reality. We do not need to accurately predict the *actual* RI values. What matters is that its returned estimates are correlated with the true Rand index values. When given C clusterings (as we will see, $C \leq 300$), it is sufficient if the top-K feature sets suggested by the heuristic is likely to contain the true top-1. In essence, the job of our heuristic is just to prune away the C-K least promising candidates.

Our proposed heuristic is based on the observation that a good clustering will have *cohesion* (most objects will be similar to other objects), and *separation* (most objects will be far from a significant fraction of the other objects). The reader will appreciate this is essentially a description of Rousseeuw's *Silhouette Score* [72]. High values indicate objects are well matched to their own cluster and poorly matched to neighboring clusters.

The silhouette score has mild assumptions; it is most appropriate when the clusters are convex-shaped, and the classes are not drastically imbalanced. These assumptions are

Algorithm 5 Our heuristic function for approximating our clustering similarity measure.

```
1: procedure EVALUATE($fvs, pred$)        ▷ Feature vectors, KMeans cluster predictions.
2:     $clusters, sizes \leftarrow unique(pred)$
3:     $l, s \leftarrow \texttt{max}(sizes), \texttt{min}(sizes)$
4:     $diff \leftarrow (l - s)/l$
5:     $SS' \leftarrow \texttt{max}(0, \texttt{SILHOUETTE\_SCORE}(fvs, pred))$
6:     return $SS' - diff$
7: end procedure
```

often true in practice, particularly in the task of clustering. We refer to our heuristic as `Evaluate`, and describe it in Algorithm 5.

In lines 2-3, we consider the number of time series assigned to each cluster, and the sizes of the largest and smallest of these clusters. In line 4, we create a penalty term $diff$ which considers the size difference of the largest and smallest clusters, normalized to the size of the largest cluster. In line 5, We consider a modified Rousseeuw's Silhouette Score [72], where negative Silhoutte Score values are remapped to 0. `Evaluate` is simply the difference between the 0-minimum silhouette score with a penalty our notion of relative cluster imbalance according to K-Means' cluster assignments.

4.5.2 Observations on Active Learning

Most research efforts on semi-supervised clustering use pairwise constraints [56]. The user is shown a pair of objects and is asked if they `must-link` or `must-not-link`. We propose to instead show the user two tentative clusterings of a random subset of the data and ask them which they prefer. This greatly improves the quality of the user feedback

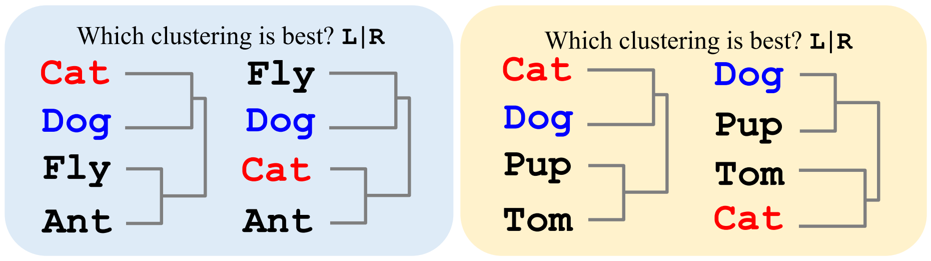


Figure 4.5: Two decision panels ask us to decide if {`Cat|Dog`} belong in the same cluster. *left*) Here most people choose L, uniting Cat and Dog in the same cluster. *right*) Here most people pick R, dividing Cat and Dog into separate clusters.

to the algorithm. Suppose during clustering, you were asked to provide a constraint for the pair {`Cat|Dog`}. There is a sense in which we might see these two examples as being essentially identical, for example as in *"Cats and Dogs for Adoption"* [2]. However, there is another sense we might see {`Cat|Dog`} as completely *opposite*, as in *"it wasn't always such a pretty sight 'cause we used to fight like cats and dogs"* [58].

However, consider Figure 4.5, which asks the same question with the additional context of clustering. When the user sees Figure 4.5.*left*, they immediately see that there is a natural clustering corresponding to mammals vs. insects. In contrast, if the user were to see Figure 4.5.*right*, they would see a natural clustering corresponding to feline vs. canine.[1] Note that the example above assumes that the user is accessing out-of-band data, in this case their common sense understanding of animal taxonomy. Likewise, we assume our users are exploiting their understanding of their domain.

A simple pairwise constraint only considers the relationship between a single pair of time series. However, even a relatively small dendrogram with ten instances summarizes forty-five pairs of pairwise comparisons, presenting a much richer offering to evaluate. It

[1] *Pup* is the young of several species of animals, but principally dogs. A *Tom* is a male cat.

is important to note that while dendrograms are a natural way to elicit feedback, the underlying clustering algorithm does not have to be a hierarchical clustering algorithm. Our use of the dendrograms is solely to use the binary choice from the user to steer the greedy search; we do not extract pairwise constraints by analyzing their topology.

4.5.3 Obtaining User's Feedback

As we will later show in Section 4.6, it is possible to create a heuristic scoring function that prunes most of the unpromising feature sets. This leaves us with a small collection of feature sets that the user must evaluate to guide the search algorithm. It might be possible that an experienced user can evaluate a clustering numerically ("this clustering has a Rand index of about 0.7"), or at least relatively ("Clustering A is probably 10% better than Clustering B"). However, we want to make the weakest possible assumptions about the user: that they can correctly make the binary choice between two clusters.

Without loss of generality, we propose to show K clusterings to the user, where K^2 is a power-of-two integer, and use a classic single-elimination (knockout or sudden-death) tournament to pick the best clustering. Figure 4.6 illustrates the process for K=4. Note that the user is only tasked with making K-1 binary choices to obtain a winner.

K is a parameter in the search algorithm we introduce in Section 4.5.6. If K=0, no human input is used, and the search algorithm only uses its heuristic function. Although `Evaluate` is generally correlated with the Rand index, solely relying on it would likely not produce the best possible clustering. In contrast, if K is arbitrarily large, the user could

[2]In all experiments, the "K" in K-means and K-Shape follow the "the number of clusters is set to the number of classes in the problem" assumption. When discussing K, we refer to the number of feature sets permissible to show to a user in the tournament.

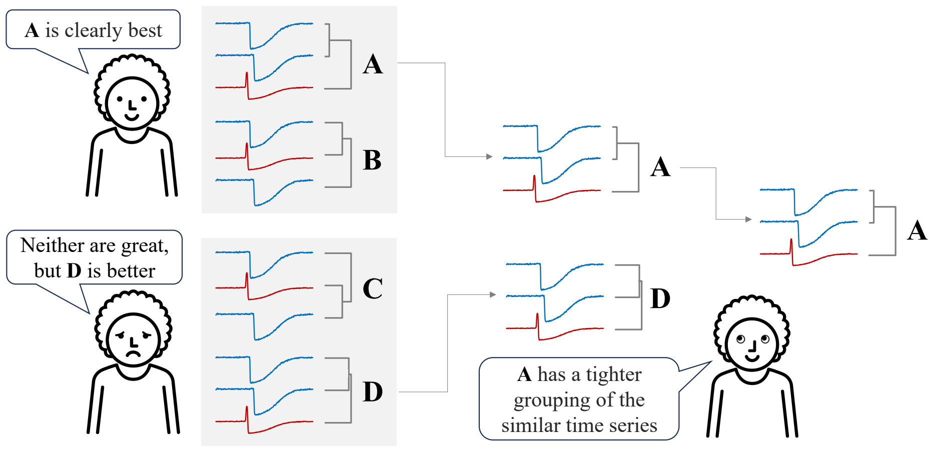

A is clearly best
Neither are great, but **D** is better
A
B
C
D
A
D
A
A has a tighter grouping of the similar time series

Figure 4.6: A single-elimination tournament of four different subsets of features (represented by the clustering they produce). For clarity we show the time series colored by class, however the user sees only a single color.

guide the search algorithm to a high-quality feature subset, but only at the expense of hours of tedious work. However, as we will demonstrate, for relatively small values of K (i.e. as small as 4 or 8), we can obtain high quality clusterings using less than a minute of user time.

This process is outlined in Algorithm[3] 6. It takes in a time series dataset's C24 feature vectors, the current best feature set, a list of the heuristic's top-K feature subsets, and it returns the best feature subset of those K. In general, for evaluation purposes, we assume that this best feature subset is also most likely to have the highest Rand index, but as noted in Section 4.5.4, the user's needs may not always align with the Rand index.

In line 2 of Algorithm 6, we initialize the first candidate feature pair as the *incumbent*, i stores the winning feature pair's index, and j for counting. The loop in line 3 compares it to the other K-1 candidate feature pairs. In line 4, four random time

[3]Because this subroutine gathers human subjective opinion, a pendant may argue it is not an *algorithm*.

series' C24 feature vectors from each class are drawn for plotting in the dendrogram to show to the user. The specificity of *4*N* is just for the purpose of limiting the size of the dendrogram show to the user, and can be changed.[4] The incumbent and challenger feature subsets are set in lines 5-6. In line 7, a dendrogram of the random time series' feature vectors and the challenger and incumbent feature subsets are shown to the user. The user will pick "the best" dendrogram of the two. Their selection is stored in w, and its corresponding index in i. If the match winner is not the *incumbent*, the current challenger is inaugurated to be the *incumbent* in line 9. By the end of the matches, the incumbent is the winner of the single-elimination tournament.

4.5.4 The User's Objective Function

In general, we envision the user giving feedback reflecting the clustering that aligns with their informational needs. However, sometimes that guidance may not align with the Rand index evaluation. For example, consider the classic Gun-Point data [27]. The class labels for the Gun-Point dataset reflect the eponymous condition of the actor processing a firearm (gun, or no gun). However, a user may be interested in the sex of the actors instead, who have very different physiques. Because of this, nearest neighbor classification with the Euclidean distance works very well on this dataset, but clustering algorithms using Euclidean distance struggle to outperform random guessing when evaluated with the class labels that reflect weapon possession, not sex. It is important to recall such possibilities when we evaluate clustering algorithms.

[4]For simplicity, we do not declare N a parameter to Algorithm 6. It is a constant determined by the properties of the dataset, and not a parameter we study in this work.

Algorithm 6 Present and build the single-elimination knockout to the user.

```
1: procedure HumanEval($F, L, fvs$)    ▷ $F$ the current best feature subset, $L$ a list of the $K$ candidate feature pairs, $fvs$ the C24 feature vectors
2:     $incumbent \leftarrow L[0], i \leftarrow 0, j \leftarrow 1$
3:     for $challenger \in L[1:]$ do
4:         $s \leftarrow$ sample($fvs, F, 4 * N$)    ▷ N: # classes
5:         $c \leftarrow F \cup challenger$
6:         $i \leftarrow F \cup incumbent$
7:         $w \leftarrow$ HumanFeedback($fvs[s, c], fvs[s, i]$)
8:         if $w == c$ then
9:             $incumbent \leftarrow challenger$
10:            $i \leftarrow j$
11:        end if
12:        $j \leftarrow j + 1$
13:    end for
14:    return $incumbent$, $i$
15: end procedure
```

4.5.5 Augmented Search Operators

Given that we have a heuristic to evaluate the cluster quality of any subset, the natural question is how best to use it. Clearly testing all $2^{24} - 1 = 16,777,215$ possible subsets is untenable. The normal solution is to use forward greedy search (FGS) [94]. Here, FGS requires 24 evaluations at the beginning of the search, then 23, then 22 etc., for a more tenable 300 evaluations. However, there is a well-known limitation of FGS: it may fail to add features that are good, but only when used in conjunction with each other. For example, consider the "Meat" dataset from the UCR Archive [27]. In Figure 4.7 we show features *f2* and *f21* as scatterplots or distributions. Visually, we can see that in the 2D space, Class 2 is linearly separable from {Class3,Class 1}. We also see that Class3 and Class 1 themselves are are almost linearly separable. However, if we only examine *individual* features, the class overlap is high.

This visual intuition is reflected in the numerical clustering results. Here the RI of just using *f2* or *f21* are 0.73 and 0.71 respectively. However, in combination, the set {*f2* , *f21*} has a dramatically better RI of 0.88. Moreover, there are individual features that are just slightly better than the better of *f2* and *f21.* If we run classic FGS on this dataset, we may fail to find this high performing feature pair. Of course, this *local maximum* problem is not unique to time series features, however it does seem common for C24, perhaps because of the way the C22 features were discovered in the first place [35,57]. We propose to mitigate this problem by augmenting the FGS operators to include adding any combination of two features. Clusterings where one feature is added and clusterings where two features are added are commensurate; we do not have normalize or adjust for the number of features in

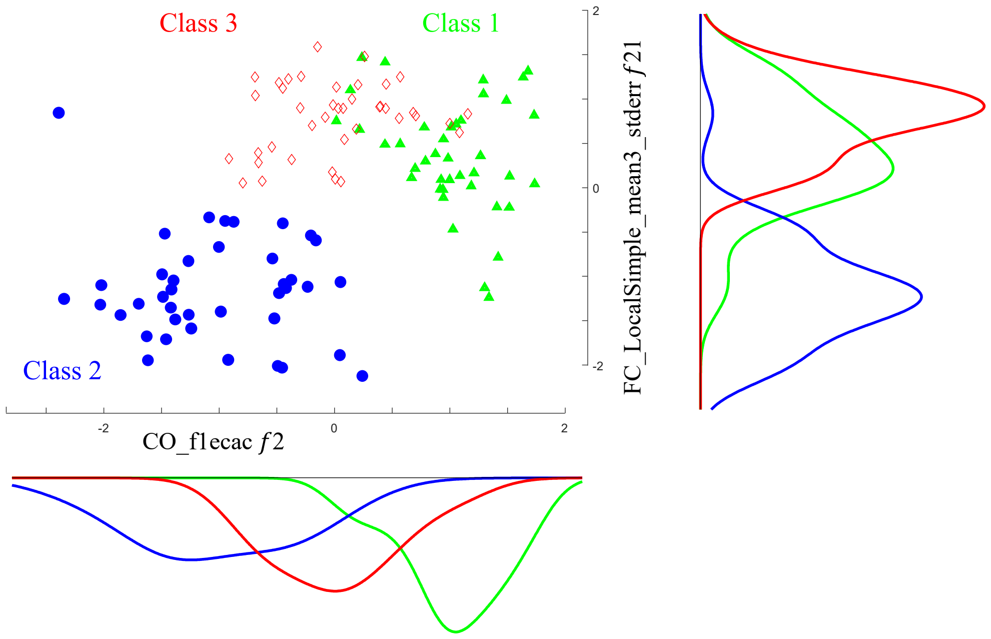


Figure 4.7: The *f2* and *f21* features from the Meat dataset. The classes are almost perfectly linearly separable in 2D, but overlap in individual dimensions.

our search evaluation.

This strategy increases the number of evaluations needed, but only modestly so. It requires at most 300 evaluations at the beginning of the search, which comes from considering adding $\binom{24}{2} = 276$ pairs of features and 24 single features at this time. As we descend the search tree, the number of combinations decreases, giving us a worst case of 2600 evaluations. Recall that the vast majority of these evaluations will be done with a heuristic, and only a handful will require human attention.

Of course, it may be argued that there may be circumstances where there are *triples*, *quadruples*, etc. of features that are independently poor, but collectively useful. However, broaching the "overcoming local minima with combinations of features" expansion should begin with two features, and we demonstrate in Section 4.6 this is sufficient for the problems we tackle in this work. We leave the study of larger feature combination sizes for future work.

We do this in Algorithm 7, where we create the list of candidate feature pairs to add in the next iteration of CEREBRIC. In line 3, we consider the remaining C24 features that have not been added to our best feature set yet, and in line 4 we create these pairwise combination of features based on that remaining set of C24 features. To give CEREBRIC the choice to add just *one* feature at a time, we also consider identity feature pairs for these remaining features in this set.

Algorithm 7 Return a list of candidate feature sets given the current feature set.

```
1: procedure EXPAND($fset$)
2:     $F \leftarrow \{1 : 24\}$                                  ▷ Specific to C24.
3:     $R \leftarrow F - fset$
4:     $C \leftarrow \{\texttt{combination}(R, 2) \bigcup (i, i), i \in R\}$
5:     return $C$
6: end procedure
```

4.5.6 Search Algorithm

We present the main subroutines in our methodology.

The Main Search Algorithm

The main search algorithm is outlined in Algorithm 8. It begins with initializing the current best feature set F and their corresponding heuristic scores P to empty. This algorithm adds either one or two features until all 24 features have been added, or the search terminates early. In line 5, we to create all candidate feature pairs to add to F at this iteration; this is done using Algorithm 7. CEREBRIC opts to add up to two features at a time, and the possibility of adding single features is accommodated by including identity feature pairs in the list of candidate feature pairs.

In line 6, we call Algorithm 9, which computes the heuristic scores for all candidate feature pairs (Algorithm 5). Algorithm 9 returns `None` for both S and $cpairs'$ when there is no reasonable feature pair left to add to the best feature subset, and the feature search terminates early. In line 11, the top-K heuristic candidate feature pairs are shown in the form of dendrograms to the user in `HumanEval` (outlined in Section 4.5.3). We rely on human judgement to return the true best subset.

The feature search returns the ordered best feature set list F, and its corresponding heuristic scores P. For example, the C24 feature subset F`[:3]` will have its corresponding heuristic score stored in P`[3]`.

Evaluating the Candidate Feature Pairs

In Algorithm 9, we initialize a list of heuristic scores S for each candidate feature pair in $cpairs'$, and an early termination flag in line 2. For every pair of features $cpairs$ (line 3), determine if it is suitable for evaluation in lines 5-6. In line 5, we consider adding them to the current best feature subset (line 5), and remove any exemplars whose feature vectors contain NaNs as a preprocessing step (line 6). In line 7, we check if at least 30% of the original dataset can be evaluated in the clustering after this preprocessing step. If this check is failed, we skip considering this feature pair. If this check passes, we progress.

Algorithm 8 Conduct the C24 Feature Search.

```
1: procedure FeatureSearch($D, K$)    ▷ Time series dataset $D$ and Human Effort $K$
2:     $F \leftarrow []$, $P \leftarrow []$
3:     $fvs \leftarrow catch24(D)$    ▷ C24 feat. vec. of each ts ∈ D
4:     while $|F| < 24$ do
5:         $cpairs \leftarrow$ Expand($F$)    ▷ Alg. 7
6:         $S, cpairs' \leftarrow$ EvalExpanded($F, fvs, cpairs$)    ▷ Alg. 9
7:         if $S$ is None and $cpairs'$ is None then
8:             break
9:         end if
10:        $sort \leftarrow$ argsort($S$)[: $K$]
11:        $w, i \leftarrow$ HumanEval($F, cpairs'[sort], fvs$)    ▷ Alg. 6
12:        $F$.append($w$)
13:        $P$.append($sort[i]$)
14:    end while
15:    return $F, P$
16: end procedure
```

Should none of the remaining feature pairs in *cpairs* pass this check, the entire C24 feature search terminates early (lines 14-15).

If a feature pair is suitable for evaluation, we record cluster predictions of K-Means on our feature vectors, where the number of clusters is set to the number of classes in the dataset N (line 9).[5] This decision introduces the assumption of one cluster per class label. `Evaluate` uses these predictions and valid feature vectors to compute our heuristic score. In line 11, we keep a list of candidate feature sets corresponding to the list of heuristic scores maintained in line 10. At this point, we can sort this list of candidate feature sets by their estimated utility (which we do in line 10 of Algorithm 8).

While we have confidence in the *specificity* of this ranking, we are less confident of its *sensitivity*. In other words, while a feature set designated as poor by `Evaluate` is (often) truly weak, we are less sure that a feature set designated as strong by `Evaluate` is truly strong, as demonstrated in Section 4.6.

4.5.7 A Preview of CEREBRIC's Effectiveness

In Figure 4.8, we offer a brief preview the results we can expect in our formal, full evaluation by revisiting the example shown in Figure 4.2 with CEREBRIC to compare. Like CEREBRIC, the Greedy Catch24 Search adds up to two features at iterations of the search. The results are remarkable, with CEREBRIC achieving nearly the same improvement achieved by the optimal Oracle Search algorithm.

[5]For the same reasons as those for Algorithm 6, we do not declare N a subroutine parameter.

Algorithm 9 For each remaining candidate pair of features to add, evaluate the performance of their addition using our heuristic function, `Evaluate` (Algorithm 5).

```
1: procedure EvalExpanded($F, fvs, cpairs$)    ▷ $F$ the current best feat. set, $fvs$ the C24 feat. vectors, $cpairs$ the candidate feature pairs to add at this iteration.
2:     $S \leftarrow []$, $cpairs' \leftarrow []$, $quit, \leftarrow True$
3:     while $cpairs.empty()$ do
4:         $cpair \leftarrow cpairs.pop()$
5:         $cset \leftarrow F \bigcup cpair$
6:         $fvs' \leftarrow$ preprocess($fvs[cset]$)
7:         if $|fvs'| >= |fv| * 0.3$ then
8:             $quit \leftarrow False$
9:             $pred \leftarrow$ KMeans($fvs', N$)    ▷ $N$: # classes
10:            $S$.append(Evaluate($fvs', pred$))    ▷ Alg. 5
11:            $cpairs'$.append($cpair$)
12:        end if
13:    end while
14:    if $quit$ then
15:        return None, None
16:    end if
17:    return $S$, $cpairs'$
18: end procedure
```

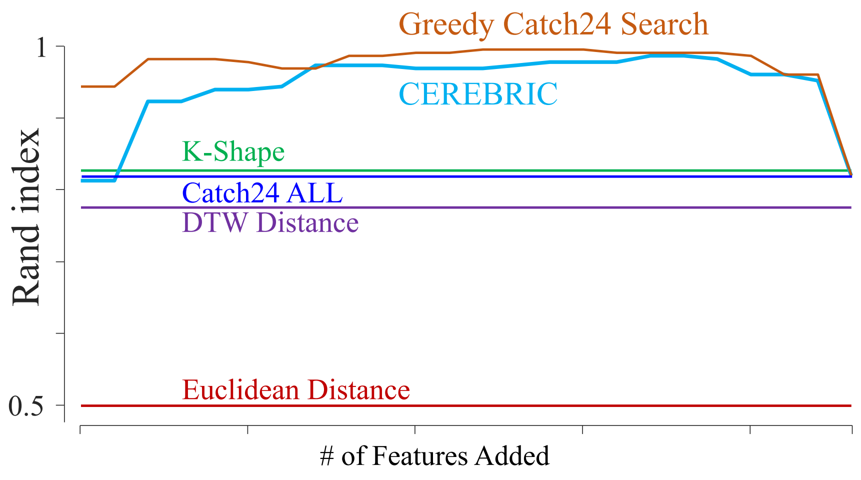


Figure 4.8: The clustering performance of CEREBRIC with $K = 8$ (see Figure 4.2) against three shape-based baselines and the Oracle Search.

To review: it was demonstrated that using a subset of these C24 features can be better than using all of them, and CEREBRIC is a feature search method designed to prioritize which features to add first. We propose a human guide the search in an active learning setting through the use of dendrograms, and they can choose to stop adding features once they are satisfied with the results. The parameter K represents the number of candidate pairs of features a user must evaluate in a single-elimination knockout tournament at each iteration of the algorithm.

However, because we conduct no human study in our experiments, we proxy `HumanEval` in our reported results. For each top-K heuristic candidate pairs p, we compute the Rand index between the dataset labels and the K-means cluster assignments of $fvs'[F \cup p]$, where F is the current best feature subset. The "winning" candidate feature pair will be the pair that contributed to the highest overall Rand index. This our proxy for human feedback. This highest RI of the top-K pairs is the value presented in our figures and results in Section 4.6.

Time Complexity. When comparing two time series, the time complexity of the previously discussed three rival methods and CEREBRIC are:

- **ED**: O(W^2N), where W is the number of time series in the dataset, $(W^2 - W)$ the number of pairs of time series, and N is the length of both time series.

- **DTW**: $O(W^2NM)$, where W is the number of time series in the dataset, $(W^2 - W)$ the number of *pairs* of time series; N and M are the length of the time series in those pairs.

- **K-Shape** [65]: $O(\texttt{max}\{KMlog(M), M^2, KM^3\})$ time per iteration, where K is the number of clusters (and classes, in our experiments), and M is the length of both time series.

- **CEREBRIC**: The overall time complexity of CEREBRIC is $O(WN^{1.6} + C(W^2))$, where $C \leq 2600$ and denotes the number of feature set evaluations are conducted, W is the number of time series in the dataset (and also the number of C24 feature vectors). Converting time series to C24 feature vectors using catch22 *empirically scales* in $O(WN^1.6)$ [57], where N is any given time series' length (C24 feature vector computation for time series is done on a series-by-series basis).

4.6 Experimental Evaluation

To ensure our experiments are reproducible, we have built a website [7] containing our data and code used in this work.

For simplicity and consistency, we use K-means for all experiments, for all approaches (human feedback subroutine uses single-linkage hierarchical clustering). However, note that nothing in our proposed method precludes other clustering algorithms, such as density peaks or DBSCAN. For the purpose of empirical verification, we assume one cluster per class label, and use a clustering algorithm that enforces this.

4.6.1 Experimental Details

All experiments were conducted on an Intel® Xeon® CPU E5-2680 v3 @ 2.50GHz, unless otherwise stated. For all clustering algorithms, we use scikit-learn (sklearn) [66] or tslearn [91], a package built around sklearn specifically for time series. For loading the time series classification datasets [61], we write a wrapper around the data loader implemented by aeon.

Seeding. There are various points within our method that have elements of variance or randomness. Wherever there is a seed accepted, we set the seed to 666 (e.g. sklearn's `random_state` parameter). All other significant parameter values are either noted in the main paper or this section on reproducibility. Any other parameter uses default values. We normalize the C24 feature vectors after computing the feature values. In possible cases where every exemplar has a NaN value for the same feature, we simply replace those NaNs with 0s.

Stratified sampling. CEREBRIC is relatively fast, however both K-Shape and DTW can be much slower for the handful of UCR datasets with high numerosity and/or high dimensionality. Consequently, we conducted stratified sampling on datasets with more than 636 time series[6] in them and sampled a stratified sample of 636 time series instead. Also note that sixteen of the UCR datasets have different length exemplars.

Handling time series of different lengths. For the purposes of evaluation against the other baselines, we resample such time series to the dataset's mean length, supplying the resampled time series to all methods. However, note that CEREBRIC itself does not have problems comparing time series of (even vastly) different lengths, as all such time series map to 24-dimensional vectors.

4.6.2 A Sanity Check of Our Assumptions

As noted above, the utility of our ideas depends on two assumptions: that our heuristic is at least correlated with the true RI, and that human inspection of small dendrograms can generally distinguish the better of two options (relative to the RI). The success of our experiments in subsequent sections implicitly confirms these assumptions are plausible, but in Figure 4.9 we show some direct explicit evidence.

A user can clearly see that the dendrogram in Figure 4.9.*left* is poor, for example by noting there appears to be two types of time series, symmetric and asymmetric, that are seemingly randomly placed in subtrees. Figure 4.9.*center* does a better job at organizing the objects, makes a single mistake in placing a symmetric time series in a subtree that is otherwise all asymmetric. Finally, Figure 4.9.*right* shows a perfect clustering.

At the top of our search tree, we begin by testing all 300 combinations of one or two features with our heuristic. In Figure 4.10 we show the Pearson correlation between

[6]The median number of time series in the 128 univariate UCR Archive datasets is 636 (rounded to the nearest whole number).

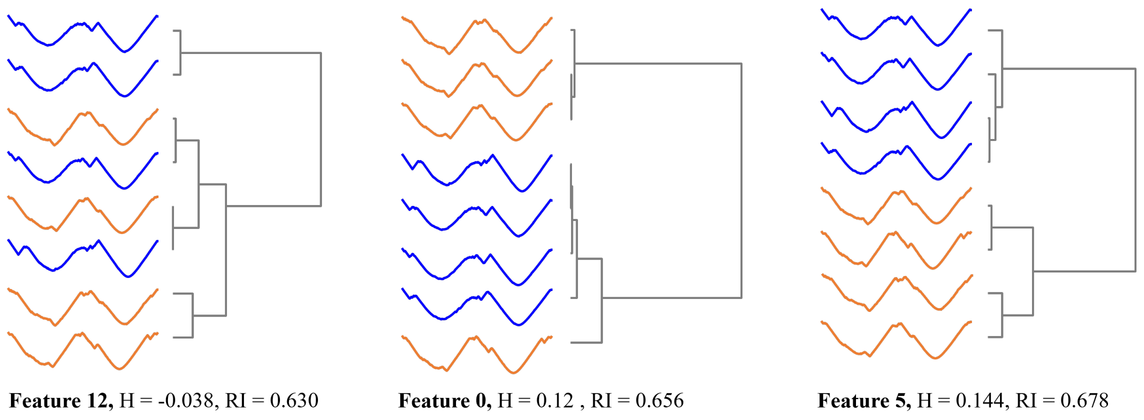


Figure 4.9: Three clusterings of eight examples from the UCR-Car dataset, using three different C24 features. The labels are ‘sedan’ and ‘pickup’. On all metrics, they are: *left-to-right*) poor, good, better. Shown in color for clarity; users will not see color when evaluating.

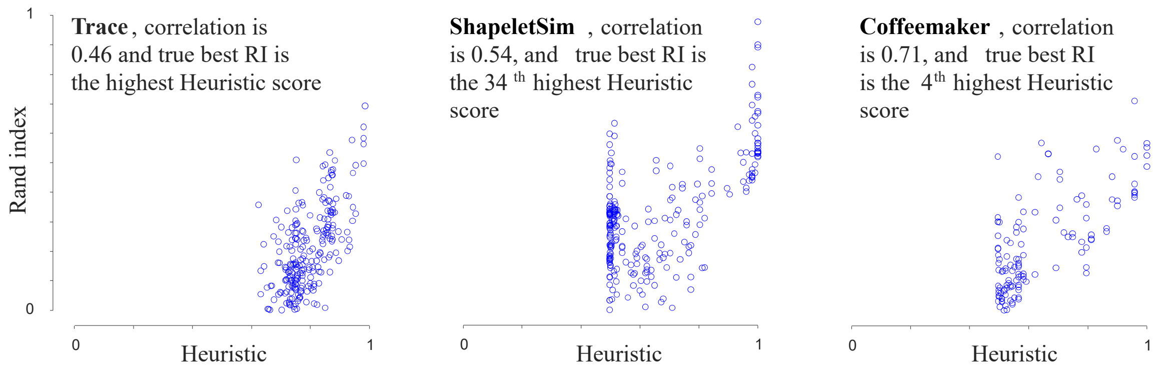


Figure 4.10: Scatterplots showing the correlation between our heuristic score and the true Rand index.

our heuristic score and the true RI for some randomly chosen datasets. The correlation is high, but that is a weak measure of utility. We only really care about the correlation in the upper right quadrant. Thus, we also ask, how does the true best feature set rank in our heuristic ranking?

For Trace [27], this is perfect. For Coffeemaker [7], it is fourth place, meaning that as long as K=4 or greater, the human annotation subroutine in Algorithm 5 should be able to extract the true best performing feature set. Only in ShapeletSim [27] do we see a poorer result. However, note that even here our heuristic is suggesting a feature set that is in the top 16.5% of all features. Moreover, it is still possible (and likely), that as the greedy search progresses, that the true best performing feature set will be included in the K best feature sets recommended by our heuristic. In general, we have between 11 and 23 more chances to include this feature set.

We argue in Section 4.5.2 that a human can estimate the Rand index well, and that this can be used to guide the search. We present Figure 4.11, where we compare the performance of variations of CEREBRIC to consider different qualities of human eye and judgement for dendrograms to explore this. The CEREBRIC variants are:

- **CEREBRIC K=8**: A user with a "good eye". This assumes the user chooses the best of the K choices. This provides the upper bound for the effectiveness of our ideas, independent of the two subroutines that could be improved in future work.
- **CEREBRIC K=0**: A user with an eye as good as `Evaluate` (Algorithm 5). No labels are examined to compute this.
- **Worst CEREBRIC K=8**: A user that is "unlucky". They consistently pick the worst of the K clusterings.
- **True Oracle**: Cheats by looking at class labels of all candidate clusterings, computing the RI, using this to guide the search. It provides an absolute upper bound as to the effectiveness of C24 for clustering, independent of the use of active learning.

For clarity of the plot, we do not visually show the baseline method performances, but present them here: ED (0.519), DTW (0.499), and K-Shape (0.533). CEREBRIC K=8

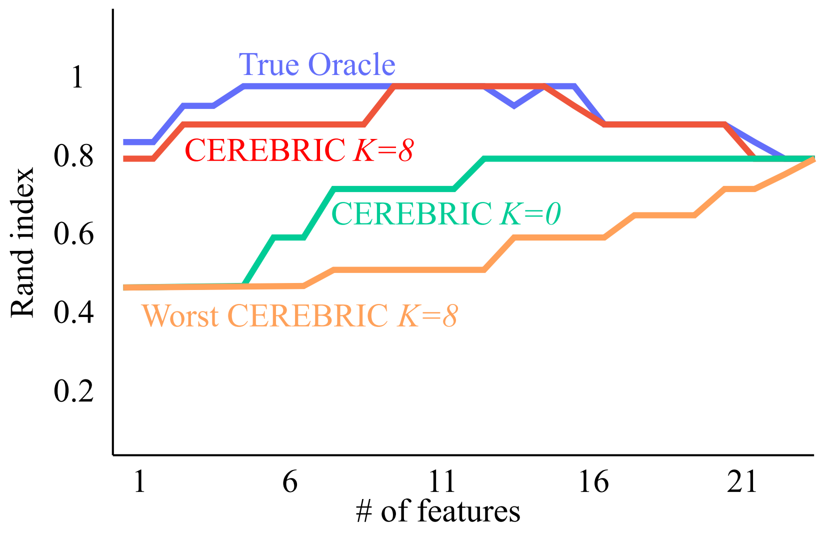


Figure 4.11: The True Oracle and three variants of CEREBRIC on the BirdChicken [27] dataset.

eventually achieves the performance the True Oracle did. Neither CEREBRIC K=0 nor Worst CEREBRIC K=8 find a subset of features that performs better than using all C24 features. The results for these robustness experiments for all other 128 univariate UCR Archive datasets are available at [7].

4.6.3 The UCR Archive

We begin by considering the UCR Archive [27]. We compare the performance of **CEREBRIC *K*=8**, **CEREBRIC *K*=0**, and the **True Oracle** from our experiments described in Figure 4.11. We present the results of the True Oracle and two CEREBRIC variants against ED, DTW, and K-Shape in Table 4.1. We can summarize the results by asking two questions: 1) Does our method beat or equal at least one of the shape-base rival methods? 2) Does our method beat or equal all of the shape-based rival methods?

	Method	**Bests *one* rival**	**Bests *all* rivals**
RI	True Oracle	127/128 (99.2%)	104/128 (81.3%)
	CEREBRIC K=8	124/128 (96.9%)	85/128 (66.4%)
	CEREBRIC K=0	117/128 (91.4%)	66/128 (51.6%)
ARI	True Oracle	127/128 (99.2%)	104/128 (81.3%)
	CEREBRIC K=8	124/128 (96.9%)	74/128 (57.8%)
	CEREBRIC K=0	113/128 (88.3%)	59/128 (46.1%)
NMI	True Oracle	126/128 (98.4%)	97/128 (75.8%)
	CEREBRIC K=8	121/128 (94.5%)	73/128 (57.0%)
	CEREBRIC K=0	109/128 (85.2%)	57/128 (44.5%)
AMI	True Oracle	125/128 (97.7%)	97/128 (75.8%)
	CEREBRIC K=8	122/128 (95.3%)	75/128 (58.6%)
	CEREBRIC K=0	108/128 (84.4%)	58/128 (45.3%)

Table 4.1: The performance (RI, ARI, NMI, AMI) of feature-based methods (CEREBRIC) vs. shape-Based methods (ED, DTW, K-Shape) on the UCR Archive [27]

We present Table 4.1 to demonstrate how often CEREBRIC succeeds at its intended goals over its rivals, but it does not take into account *how well* it does. In Figure 4.12 we present a critical difference diagram computed [45] for each similarity measure (RI, ARI, NMI, AMI). First the Friedman test is conducted, then a post-hoc analysis is performed using the Wilcoxon-Holm method [45].

For all four clustering similarity scores we examine in this benchmark, CEREBRIC K=8's performance is surprisingly good in a setting where datasets may be biased against our approach and biased towards shape-based approaches. CEREBRIC K=0 often can best a single shape-based rival, but is dramatically worse at besting *all* rivals in comparison to CEREBRIC K=8. This suggests that a good human eye will be beneficial in an active learning setting.

The original archive curator confirmed to us [48], many of the UCR Archive datasets were originally edited by using the Euclidean Distance. Therefore, it is tauto-

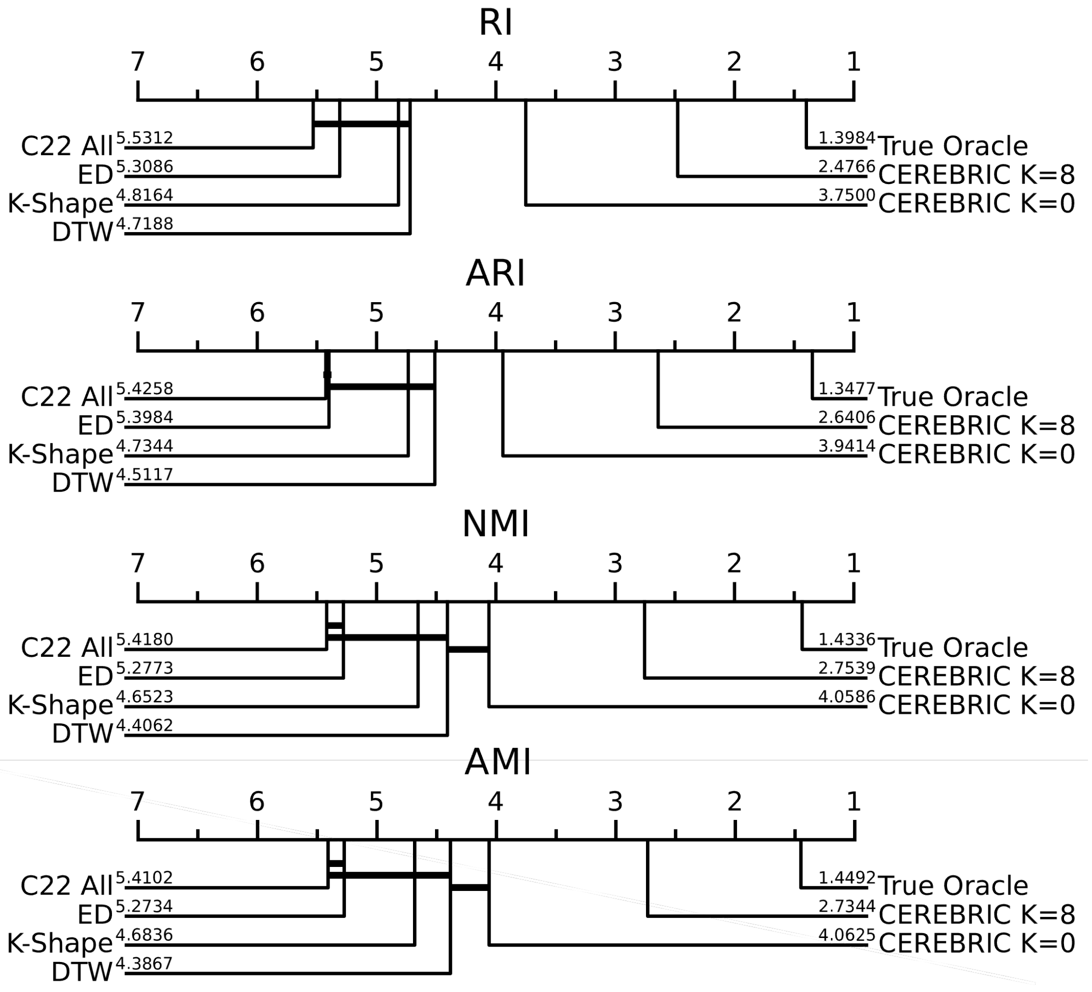


Figure 4.12: The critical difference diagrams of CEREBRIC variants against three shape-based rivals under four different similarity measures.

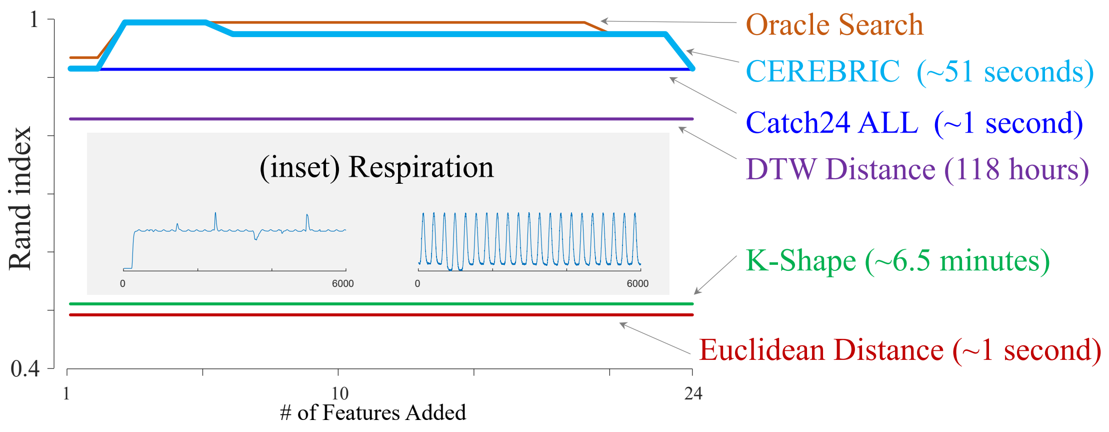


Figure 4.13: The clustering performance of CEREBRIC with K=8 on the Non-Contact Respiration dataset.

logical to say that shape-based measures will work well here. Nevertheless, as this resource is something of a "Rosetta Stone" for time series algorithms, we will make the comparison. A spreadsheet of all *individual dataset results* can be found at our website [7]. Because of this weakness of the UCR Archive, below we consider more challenging real world case studies.

4.6.4 Case Study: Non-Contact Respiration

An ongoing research effort suggests that it is possible to use Wireless lightwave sensing to discreetly monitor human breathing from a distance [44]. The researchers created a dataset containing examples of medical conditions and data containing faults caused by "*environmental noise/internal malfunction*". In Figure 4.13 we apply CEREBRIC K=8 to this dataset, and see that CEREBRIC closely follows the Oracle Search algorithm, which "cheats" by looking at the ground truth labels. The nearest competitor is DTW, which is 8,000 times slower.

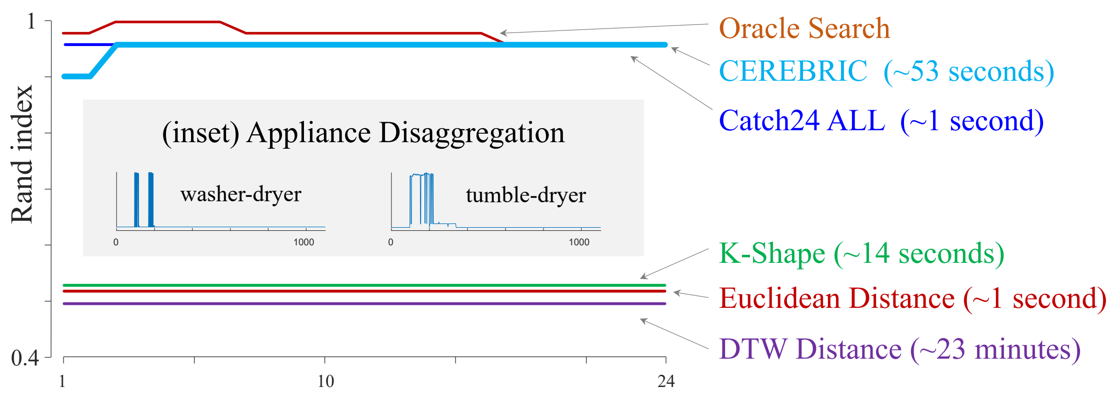


Figure 4.14: The clustering performance of CEREBRIC K=8 on the Appliance Disaggregation dataset [25].

4.6.5 Case Study: Appliance Disaggregation

Appliance disaggregation is the task of examining an aggregate electrical signature and parsing out the signatures of individual appliances [25]. In Figure 4.14 we consider a benchmark normally used for classification in this domain. Here CEREBRIC K=8 does not quite match the performance of the Oracle, but it is significantly better than all the shape-based methods, which perform at close to chance levels (i.e. the default rate). Note the CEREBRIC K=8 performs as well as using all C24 features. However, by using fewer features (here, just one) it allows the possibility of interpretability of the clustering.

4.6.6 A Case Study in Robustness

One of the properties of the UCR Archive that is most unrealistic is that every instance belongs to a well-defined class. However, most real-world datasets also have some noisy/outlier instances. For example, an industrial dataset may have occasional sensor faults [44], or a dataset of gait cycles may have instances that reflect a user tying their

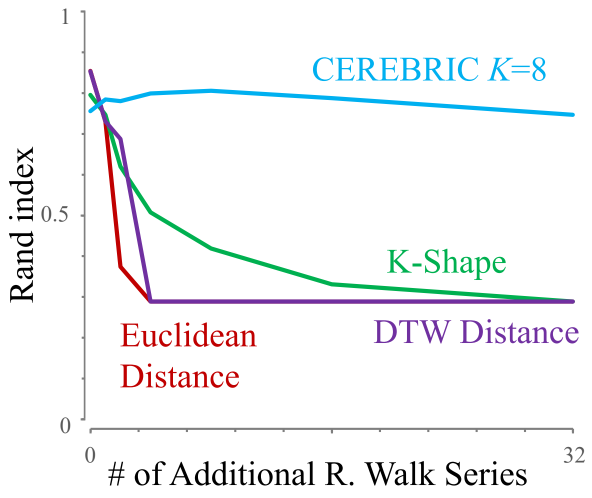


Figure 4.15: The robustness of four algorithms presence of extraneous Random Walk samples on the OliveOil [27] dataset. For each method, the reported performance is averaged over ten runs.

shoelace mid-exercise. We claim that CEREBRIC is more robust to such “imperfect” datasets.

To demonstrate this, we consider the UCR OliveOil [27] dataset, which consists of four classes (and thus, we set the number of clusters in K-means to 4) and 60 time series. We incrementally added Random Walk outlier instances to this dataset, reevaluating the RI for each method, once again comparing the cluster predictions against the ground truth class labels. Note that we assessed the RI of clustering using only the original dataset’s time series, ignoring the clustering predictions of the random walk. In Figure 4.15 we summarize the results. Initially, all algorithms perform similarly. Upon adding the outlier samples (up to $>$ 50% the size of the original dataset), CEREBRIC remains robust to the noisy samples, whereas the performance of the other algorithms rapidly degrades. K-Shape degrades more gracefully than ED and DTW, but still is outperformed by CEREBRIC.

Dataset	# classes	Rand index of Methods		
		Classic FGS	CEREBRIC	Difference
ECGFiveDays	2	0.545	0.604	0.059
Strawberry	2	0.543	0.600	0.056
BME	3	0.794	0.943	0.149
Rock	4	0.720	0.774	0.053

Table 4.2: Comparing the Rand indexes of four UCR Archive datasets on single-feature and pair-of-features feature search.

4.6.7 Effectiveness of Augmented Operators

In Section 4.5.4 we motivated the use of augmented search operators. It is natural to ask if they help. On many datasets they make little difference, but in a handful of cases they really do improve performance. We can see that on the UCR BME dataset [27], the maximum RI achieved by classic one-feature-at-a-time search was 0.79, whereas our augmented one-or-two-feature-at-a-time search achieved a maximum of 0.94. To show this in a reproducible manner (independent of human choices), let us consider only the Oracle versions of search (similar to the search shown in Figure 4.2). We show in Table 4.2 the Oracle version of our pair-at-a-time search and the full classic FGS (single-feature-at-a-time) on a sample of the UCR Archive tasks. Results for other UCR Archive tasks are available on our website [7].

4.7 Conclusion

We introduce CEREBRIC, a novel feature search algorithm for clustering, and show that its combination of heuristic filtering, user input, and augmented search operators allow it to outperform classic clustering algorithms. For clarity, we confined our attention to classic K-means, in future work we will apply our methodology to more advanced algorithms such as DBSCAN and density peaks [69]. We made our code/data available in [7], to allow the community to confirm and exploit our findings.

Chapter 5

Conclusions

In this dissertation, I have introduced three novel time series data mining methods. All code for each of these projects will remain available on their websites in perpetuity.

The first is PRCIS, which is discussed in Chapter 2. PRCIS is a novel distance measure for long time series, also a novel distinction made in contrast to short time series. This distinction lies akin to the amount of semantic information in the time series. A time series of a home's power demand over a day might be considered a *short time series*, and a time series of the same home's power demand over the course of a month or years might be considered a *long time series.* We present compelling arguments demonstrating that directly comparing these time series using Euclidean Distance or Dynamic Time Warping are often unfruitful. It proposes to leverage time series summarization techniques to represent a time series by a set of subsequences considered representative of it by the user's chosen algorithm. The novel distance measure formulates a distance measure considering all subsequences in these sets. I demonstrate the utility of PRCIS by presenting several reasonable clusterings produced by PRCIS. For the sake of benchmarking, we outfit two datasets into a classification task and use this task to measure PRCIS' classification performance against other rivals. PRCIS is flexible, not imposing common constraints on the source time series, length or preprocessing missing data points in some way.

The second of these methods is PUPAE, and is introduced in Chapter 3. PUPAE is a simple, novel, proposed framework for time series anomaly explanations using natural language counterfactual examples, and was intended to be used by domain-practitioners mining for insights through large amounts of data, and for the benefit of human examination. To our knowledge, work in anomaly explanations include both counterfactual examples or natural language formats, but not natural language counterfactuals. Given a user's preferred time series anomaly detector, an anomalous time series T_A, and a "normal" sample without the anomaly T_N, PUPAE will select a function $f(\cdot)$ from a bank of operators $\mathcal{F}$ that best minimizes the Euclidean distance between $f(T_A)$ and T_N. PUPAE uses the selected $f(\cdot)$ as the explanation for the anomaly. A single natural language might be formed in template from this result: T_A `would be like` T_N `except for` $f(\theta)$. Here, θ is/are the parameters that fulfill the equation 3.4.12. This work proposes a starting set of eight operators that has been selected ad hoc based on the authors' experience with time series. Our framework is flexible and amenable to changes by domain-informed users, who might remove or add operators to $\mathcal{F}$ as they deem fit for their needs. Future work can be taken in multiple directions. An anomalous time series may have two (or more) sources of anomalies. Another direction lies in learning operators, suggesting automated downstream tasks, and there will be a natural resulting trade-off with method explainability. Nevertheless, learning operators may be an interesting way to utilize the PUPAE framework for other use cases: namely quick, automated classification of anomalies. If we were to try to automate this at scale for use cases without human involvement, an exhaustive grid-search method without some kind of early termination or other heuristic involvement probably cannot be justified. The PUPAE framework of using natural language counterfactual operators will stand, but the design of the method is likely to need fundamental changes.

The third of these works is CEREBRIC, and is introduced in Chapter 4. CEREBRIC is a novel clustering algorithm building off the work Catch22 [57]. While Catch22 was presented as an all-purpose feature set for time series classification, previous work [88] has demonstrated subsets of the Catch22 subset may outperform using the entire set, still. We formally apply this well-known principle to the task of clustering on the UCR Archive [27] and through case studies. To avoid the complexities of a combinatorial-optimization problem, we outline and propose a simple active-learning framework by which to integrate user expertise. We propose this is done through a single-elimination knockout tournament, positing that a human's clustering needs align with similarity measures such as Rousseeuw's Silhouette Score. We propose that, given a visual representation of the clustered elements under some feature subset of Catch22, the human eye can estimate when some clusterings are clearly better than others. We demonstrate CEREBRIC's consistent performance across four different similarity measures for clustering: RI, ARI, NMI, and AMI. Future work includes formally demonstrating our claims about active learning and human perception w.r.t. the clustering task.

Bibliography

[1] Case school of engineering: Bearing data center, normal baseline data.

[2] Cats and dogs for adoption: Petsmart saves lives.

[3] Why everything in austria is closed on sundays.

[4] Milling data set, nasa ames prognostics data repository, 2007.

[5] Prcis companion website (contains a release of all code, data, and will exist in perpetuity.), 2022.

[6] Pupae companion website. (contains a release of all code, data, and will exist in perpetuity.), 2023-2024.

[7] Cerberic companion website. (contains a release of all code, data, and will exist in perpetuity.), 2024.

[8] Kayode S Adewole and Vicenç Torra. Dftmicroagg: a dual-level anonymization algorithm for smart grid data. *International Journal of Information Security*, 21(6):1299–1321, 2022.

[9] Charu C Aggarwal et al. *Data mining: the textbook*, volume 1. Springer, 2015.

[10] Marco Ancona, Enea Ceolini, Cengiz Öztireli, and Markus Gross. Towards better understanding of gradient-based attribution methods for deep neural networks. *arXiv preprint arXiv:1711.06104*, 2017.

[11] Arthur Asuncion and David Newman. Uci machine learning repository, 2007.

[12] Emre Ates, Burak Aksar, Vitus J Leung, and Ayse K Coskun. Counterfactual explanations for multivariate time series. In *2021 international conference on applied artificial intelligence (ICAPAI)*, pages 1–8. IEEE, 2021.

[13] Julien Audibert, Sébastien Marti, Frédéric Guyard, and Maria A Zuluaga. From univariate to multivariate time series anomaly detection with non-local information. In *Advanced Analytics and Learning on Temporal Data: 6th ECML PKDD Workshop, AALTD 2021, Bilbao, Spain, September 13, 2021, Revised Selected Papers 6*, pages 186–194. Springer, 2021.

[14] Boris Babic, Sara Gerke, Theodoros Evgeniou, and I Glenn Cohen. Beware explanations from ai in health care. *Science*, 373(6552):284–286, 2021.

[15] Anthony Bagnall, Jason Lines, Aaron Bostrom, James Large, and Eamonn Keogh. The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *Data mining and knowledge discovery*, 31:606–660, 2017.

[16] Omar Bahri, Peiyu Li, Soukaina Filali Boubrahimi, and Shah Muhammad Hamdi. Temporal rule-based counterfactual explanations for multivariate time series. In *2022 21st IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 1244–1249. IEEE, 2022.

[17] Kasun Bandara, Christoph Bergmeir, and Slawek Smyl. Forecasting across time series databases using recurrent neural networks on groups of similar series: A clustering approach. *Expert systems with applications*, 140:112896, 2020.

[18] Syed Khairul Bashar, Eric Ding, Allan J Walkey, David D McManus, and Ki H Chon. Noise detection in electrocardiogram signals for intensive care unit patients. *IEEE Access*, 7:88357–88368, 2019.

[19] Gustavo EAPA Batista, Eamonn J Keogh, Oben Moses Tataw, and Vinicius MA De Souza. Cid: an efficient complexity-invariant distance for time series. *Data Mining and Knowledge Discovery*, 28:634–669, 2014.

[20] Alberto Blanco-Justicia and Josep Domingo-Ferrer. Machine learning explainability through comprehensible decision trees. In *Machine Learning and Knowledge Extraction: Third IFIP TC 5, TC 12, WG 8.4, WG 8.9, WG 12.9 International Cross-Domain Conference, CD-MAKE 2019, Canterbury, UK, August 26–29, 2019, Proceedings 3*, pages 15–26. Springer, 2019.

[21] Alexander Bronstein, Michael Bronstein, Alfred Bruckstein, and Ron Kimmel. Partial similarity of objects, or how to compare a centaur to a horse. *International Journal of Computer Vision*, 84:163–183, 08 2009.

[22] Ruth MJ Byrne. Counterfactuals in explainable artificial intelligence (xai): Evidence from human reasoning. In *IJCAI*, pages 6276–6282, 2019.

[23] Wen-kuei Chang and Tianzhen Hong. Statistical analysis and modeling of occupancy patterns in open-plan offices using measured lighting-switch data. In *Building Simulation*, volume 6, pages 23–32. Springer, 2013.

[24] Pham Minh Chau, Bui Minh Duc, and Duong Tuan Anh. Discord discovery in streaming time series based on an improved hot sax algorithm. In *Proceedings of the 9th International Symposium on Information and Communication Technology*, pages 24–30, 2018.

[25] Dong Chen, David Irwin, and Prashant Shenoy. Smartsim: A device-accurate smart home simulator for energy analytics. In *2016 IEEE International Conference on Smart Grid Communications (SmartGridComm)*, pages 686–692. IEEE, 2016.

[26] Roberto Confalonieri, Ludovik Coba, Benedikt Wagner, and Tarek R Besold. A historical perspective of explainable artificial intelligence. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 11(1):e1391, 2021.

[27] Hoang Anh Dau, Anthony Bagnall, Kaveh Kamgar, Chin-Chia Michael Yeh, Yan Zhu, Shaghayegh Gharghabi, Chotirat Ann Ratanamahatana, and Eamonn Keogh. The ucr time series archive. *IEEE/CAA Journal of Automatica Sinica*, 6(6):1293–1305, 2019.

[28] Sergio Martin del Campo Barraza, William Lindskog, Davide Badalotti, Oskar Liew, and Arash Toyser. Active learning framework for time-series classification of vibration and industrial process data. In *Annual Conference of the PHM Society*, volume 13, 2021.

[29] Eoin Delaney, Derek Greene, and Mark T Keane. Instance-based counterfactual explanations for time series classification. In *International conference on case-based reasoning*, pages 32–47. Springer, 2021.

[30] Angus Dempster, François Petitjean, and Geoffrey I Webb. Rocket: exceptionally fast and accurate time series classification using random convolutional kernels. *Data Mining and Knowledge Discovery*, 34(5):1454–1495, 2020.

[31] Angus Dempster, Daniel F Schmidt, and Geoffrey I Webb. Minirocket: A very fast (almost) deterministic transform for time series classification. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*, pages 248–257, 2021.

[32] Berkeley J Dietvorst, Joseph P Simmons, and Cade Massey. Overcoming algorithm aversion: People will use imperfect algorithms if they can (even slightly) modify them. *Management science*, 64(3):1155–1170, 2018.

[33] German H Flores and Roberto Manduchi. Weallwalk: An annotated dataset of inertial sensor time series from blind walkers. *ACM Transactions on Accessible Computing (TACCESS)*, 11(1):1–28, 2018.

[34] B Fulcher. *Highly comparative time-series analysis.* PhD thesis, Oxford University, UK, 2012.

[35] Ben D Fulcher, Max A Little, and Nick S Jones. Highly comparative time-series analysis: the empirical structure of time series and their methods. *Journal of the Royal Society Interface*, 10(83):20130048, 2013.

[36] Mononito Goswami, Cristian Challu, Laurent Callot, Lenon Minorics, and Andrey Kan. Unsupervised model selection for time-series anomaly detection. *arXiv preprint arXiv:2210.01078*, 2022.

[37] Thomas Grote. The allure of simplicity: On interpretable machine learning models in healthcare. *Philosophy of Medicine*, 4(1), 2023.

[38] Lisa Anne Hendricks, Ronghang Hu, Trevor Darrell, and Zeynep Akata. Generating counterfactual explanations with natural language. *arXiv preprint arXiv:1806.09809*, 2018.

[39] Jacqueline Höllig, Cedric Kulbach, and Steffen Thoma. Tsevo: Evolutionary counterfactual explanations for time series classification. In *2022 21st IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 29–36. IEEE, 2022.

[40] Bryan Hooi, Shenghua Liu, Asim Smailagic, and Christos Faloutsos. Beatlex: Summarizing and forecasting time series with patterns. In Michelangelo Ceci, Jaakko Hollmén, Ljupčo Todorovski, Celine Vens, and Sašo Džeroski, editors, *Machine Learning and Knowledge Discovery in Databases*, pages 3–19, Cham, 2017. Springer International Publishing.

[41] Kyle Hundman, Valentino Constantinou, Christopher Laporte, Ian Colwell, and Tom Soderstrom. Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 387–395, 2018.

[42] Shima Imani, Frank Madrid, Wei Ding, Scott E Crouter, and Eamonn Keogh. Introducing time series snippets: a new primitive for summarizing long time series. *Data Mining and Knowledge Discovery*, 34:1713–1743, 2020.

[43] Abu Dhabi IRENA, International Renewable Energy Agency. Innovation landscape brief: Time-of-use tariffs, 2019.

[44] Md Zobaer Islam, Brenden Martin, Carly Gotcher, Tyler Martinez, John F O'Hara, and Sabit Ekin. Noncontact respiratory anomaly detection using infrared light-wave sensing. *IEEE Transactions on Human-Machine Systems*, 2024.

[45] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. Deep learning for time series classification: a review. *Data Mining and Knowledge Discovery*, 33(4):917–963, 2019.

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

[47] Han Jiawei and Kamber Micheline. *Data mining: concepts and techniques*. Morgan kaufmann, 2006.

[48] Eamonn Keogh. (personal correspondence over several emails), December 2023.

[49] Eamonn Keogh, Jessica Lin, and Ada Fu. Hot sax: Efficiently finding the most unusual time series subsequence. In *Fifth IEEE International Conference on Data Mining (ICDM'05)*, pages 8–pp. Ieee, 2005.

[50] Sina Khani and Michael L Waite. Buoyancy scale effects in large-eddy simulations of stratified turbulence. *Journal of fluid mechanics*, 754:75–97, 2014.

[51] Ahmed Khiari. Analysing neural firing with the fitz-hugh nagumo model. 2015.

[52] Guiling Li, Olli Bräysy, Liangxiao Jiang, Zongda Wu, and Yuanzhen Wang. Finding time series discord based on bit representation clustering. *Knowledge-Based Systems*, 54:243–254, 2013.

[53] Jason Lines and George Oastler. Ts-quad: A smaller elastic ensemble for time series classification with no reduction in accuracy. In *International Conference on Pattern Recognition and Artificial Intelligence*, pages 221–232. Springer, 2022.

[54] Jason Lines, Sarah Taylor, and Anthony Bagnall. Hive-cote: The hierarchical vote collective of transformation-based ensembles for time series classification. In *2016 IEEE 16th international conference on data mining (ICDM)*, pages 1041–1046. IEEE, 2016.

[55] Yue Lu, Renjie Wu, Abdullah Mueen, Maria A Zuluaga, and Eamonn 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, 2022.

[56] Zhengdong Lu. Semi-supervised clustering with pairwise constraints: A discriminative approach. In *Artificial Intelligence and Statistics*, pages 299–306. PMLR, 2007.

[57] Carl H Lubba, Sarab S Sethi, Philip Knaute, Simon R Schultz, Ben D Fulcher, and Nick S Jones. catch22: Canonical time-series characteristics: Selected through highly comparative time-series analysis. *Data Mining and Knowledge Discovery*, 33(6):1821–1852, 2019.

[58] Paul McCartney. Ballroom dancing (song). track 6 on tug of war, 1982.

[59] Elisa Mejia-Mejia, John Allen, Karthik Budidha, Chadi El-Hajj, Panicos A Kyriacou, and Peter H Charlton. Photoplethysmography signal processing and synthesis. In *Photoplethysmography*, pages 69–146. Elsevier, 2022.

[60] Matthew Middlehurst, James Large, Michael Flynn, Jason Lines, Aaron Bostrom, and Anthony Bagnall. Hive-cote 2.0: a new meta ensemble for time series classification. *Machine Learning*, 110(11):3211–3243, 2021.

[61] Matthew Middlehurst, Patrick Schäfer, and Anthony Bagnall. Bake off redux: a review and experimental evaluation of recent time series classification algorithms. *Data Mining and Knowledge Discovery*, pages 1–74, 2024.

[62] George B Moody and Roger G Mark. The impact of the mit-bih arrhythmia database. *IEEE engineering in medicine and biology magazine*, 20(3):45–50, 2001.

[63] Mohsin Munir, Shoaib Ahmed Siddiqui, Ferdinand Küsters, Dominique Mercier, Andreas Dengel, and Sheraz Ahmed. Tsxplain: Demystification of dnn decisions for time-series using natural language and statistical features. In *Artificial Neural Networks and Machine Learning–ICANN 2019: Workshop and Special Sessions: 28th International Conference on Artificial Neural Networks, Munich, Germany, September 17–19, 2019, Proceedings 28*, pages 426–439. Springer, 2019.

[64] R Nair Nimmi, Mathew Basil, and K Mohan Kumar. *Spatial and Temporal Variability of some Meteorological Parameters over Tropical Indian Ocean.* PhD thesis, Naval Physical and Oceanographic Laboratory, 2004.

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

[66] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[67] Thanawin Rakthanmanon, Bilson Campana, Abdullah Mueen, Gustavo Batista, Brandon Westover, Qiang Zhu, Jesin Zakaria, and Eamonn 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, 2012.

[68] Marcel Robeer, Floris Bex, and Ad Feelders. Generating realistic natural language counterfactuals. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 3611–3625, 2021.

[69] Alex Rodriguez and Alessandro Laio. Clustering by fast search and find of density peaks. *science*, 344(6191):1492–1496, 2014.

[70] Simone Romano, Nguyen Xuan Vinh, James Bailey, and Karin Verspoor. Adjusting for chance clustering comparison measures. *Journal of Machine Learning Research*, 17(134):1–32, 2016.

[71] Mikhail Ronkin and Dima Bykhovsky. Passive fingerprinting of same-model electrical devices by current consumption. *Sensors*, 23(1):533, 2023.

[72] Peter J Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65, 1987.

[73] Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature machine intelligence*, 1(5):206–215, 2019.

[74] Stefania Russo, Moritz Lürig, Wenjin Hao, Blake Matthews, and Kris Villez. Active learning for anomaly detection in environmental data. *Environmental Modelling & Software*, 134:104869, 2020.

[75] Andrei Sorin Sabau. Survey of clustering based financial fraud detection research. *Informatica Economica*, 16(1):110, 2012.

[76] Seref Sagiroglu and Duygu Sinanc. Big data: A review. In *2013 international conference on collaboration technologies and systems (CTS)*, pages 42–47. IEEE, 2013.

[77] Anna Saranti, Miroslav Hudec, Erika Mináriková, Zdenko Takáč, Udo Großschedl, Christoph Koch, Bastian Pfeifer, Alessa Angerschmid, and Andreas Holzinger. Actionable explainable ai (axai): a practical example with aggregation functions for adaptive classification and textual explanations for interpretable machine learning. *Machine Learning and Knowledge Extraction*, 4(4):924–953, 2022.

[78] Gustavo Scalabrini Sampaio, Arnaldo Rabello de Aguiar Vallim Filho, Leilton Santos da Silva, and Leandro Augusto da Silva. Prediction of motor failure time using an artificial neural network. *Sensors*, 19(19):4342, 2019.

[79] Patrick Schäfer. Scalable time series classification. *Data Mining and Knowledge Discovery*, 30(5):1273–1298, 2016.

[80] Sue Sendelbach and Marjorie Funk. Alarm fatigue: a patient safety concern. *AACN advanced critical care*, 24(4):378–386, 2013.

[81] Leila Seyedhossein and Mahmoud Reza Hashemi. Mining information from credit card time series for timelier fraud detection. In *2010 5th International Symposium on Telecommunications*, pages 619–624. IEEE, 2010.

[82] Athanasios Sfetsos and Costas Siriopoulos. Time series forecasting with a hybrid clustering scheme and pattern recognition. *IEEE transactions on systems, man, and cybernetics-part A: systems and humans*, 34(3):399–405, 2004.

[83] Ahmed Shifaz, Charlotte Pelletier, François Petitjean, and Geoffrey I Webb. Ts-chief: a scalable and accurate forest algorithm for time series classification. *Data Mining and Knowledge Discovery*, 34(3):742–775, 2020.

[84] Rohit Singh, Nathan Palmer, David Gifford, Bonnie Berger, and Ziv Bar-Joseph. Active learning for sampling in time-series experiments with application to gene expression analysis. In *Proceedings of the 22nd international conference on Machine learning*, pages 832–839, 2005.

[85] Reuters Staff. England brews up sufficient power for world cup tea-time surge.

[86] Martin Steiger, Jürgen Bernard, Sebastian Mittelstädt, Hendrik Lücke-Tieke, Daniel Keim, Thorsten May, and Jörn Kohlhammer. Visual analysis of time-series similarities for anomaly detection in sensor networks. In *Computer graphics forum*, volume 33, pages 401–410. Wiley Online Library, 2014.

[87] Adith Swaminathan and Thorsten Joachims. Counterfactual risk minimization: Learning from logged bandit feedback. In *International Conference on Machine Learning*, pages 814–823. PMLR, 2015.

[88] Sadaf Tafazoli, Yue Lu, Renjie Wu, Thirumalai Vinjamoor Akhil Srinivas, Hannah Dela Cruz, Ryan Mercer, and Eamonn Keogh. Matrix profile xxix: C 22 mp, fusing catch 22 and the matrix profile to produce an efficient and interpretable anomaly detector. In *2023 IEEE International Conference on Data Mining (ICDM)*, pages 568–577. IEEE, 2023.

[89] Xuning Tang, Yihua Shi Astle, and Craig Freeman. Deep anomaly detection with ensemble-based active learning. In *2020 IEEE International Conference on Big Data (Big Data)*, pages 1663–1670. IEEE, 2020.

[90] Chakkrit Tantithamthavorn, Jirayus Jiarpakdee, and John Grundy. Actionable analytics: Stop telling me what it is; please tell me what to do. *IEEE Software*, 38(4):115–120, 2021.

[91] Romain Tavenard, Johann Faouzi, Gilles Vandewiele, Felix Divo, Guillaume Androz, Chester Holtz, Marie Payne, Roman Yurchak, Marc Rußwurm, Kushal Kolar, and Eli Woods. Tslearn, a machine learning toolkit for time series data. *Journal of Machine Learning Research*, 21(118):1–6, 2020.

[92] Giulia Vilone and Luca Longo. Notions of explainability and evaluation approaches for explainable artificial intelligence. *Information Fusion*, 76:89–106, 2021.

[93] Fabien Viton, Mahmoud Elbattah, Jean-Luc Guérin, and Gilles Dequen. Heatmaps for visual explainability of cnn-based predictions for multivariate time series with application to healthcare. In *2020 IEEE International Conference on Healthcare Informatics (ICHI)*, pages 1–8. IEEE, 2020.

[94] Xiaozhe Wang, Kate Smith, and Rob Hyndman. Characteristic-based clustering for time series data. *Data mining and knowledge Discovery*, 13:335–364, 2006.

[95] Yao Wang, Zhaowei Wang, Zejun Xie, Nengwen Zhao, Junjie Chen, Wenchi Zhang, Kaixin Sui, and Dan Pei. Practical and white-box anomaly detection through unsupervised and active learning. In *2020 29th international conference on computer communications and networks (ICCCN)*, pages 1–9. IEEE, 2020.

[96] Zhiguang Wang, Weizhong Yan, and Tim Oates. Time series classification from scratch with deep neural networks: A strong baseline. In *2017 International joint conference on neural networks (IJCNN)*, pages 1578–1585. IEEE, 2017.

[97] I. Watanabe. Atm forecast, 2019.

[98] Phillip Wenig, Sebastian Schmidl, and Thorsten Papenbrock. Timeeval: A benchmarking toolkit for time series anomaly detection algorithms. *Proceedings of the VLDB Endowment*, 15(12):3678–3681, 2022.

[99] Frauke Wiese, Ingmar Schlecht, Wolf-Dieter Bunke, Clemens Gerbaulet, Lion Hirth, Martin Jahn, Friedrich Kunz, Casimir Lorenz, Jonathan Mühlenpfordt, Juliane Reimann, et al. Open power system data–frictionless data for electricity system modelling. *Applied Energy*, 236:401–409, 2019.

[100] Renjie Wu and Eamonn J Keogh. Current time series anomaly detection benchmarks are flawed and are creating the illusion of progress. *IEEE transactions on knowledge and data engineering*, 35(3):2421–2429, 2021.

[101] Chin-Chia Michael Yeh, Yan Zheng, Junpeng Wang, Huiyuan Chen, Zhongfang Zhuang, Wei Zhang, and Eamonn Keogh. Error-bounded approximate time series joins using compact dictionary representations of time series. In *Proceedings of the 2022 SIAM International Conference on Data Mining (SDM)*, pages 181–189. SIAM, 2022.

[102] Chin-Chia Michael Yeh, Yan Zhu, Hoang Anh Dau, Amirali Darvishzadeh, Mikhail Noskov, and Eamonn Keogh. Online amnestic dtw to allow real-time golden batch monitoring. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2604–2612, 2019.

[103] Chin-Chia Michael Yeh, Yan Zhu, Liudmila Ulanova, Nurjahan Begum, Yifei Ding, Hoang Anh Dau, Diego Furtado Silva, Abdullah Mueen, and Eamonn Keogh. 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. Ieee, 2016.

[104] Chin-Chia Michael Yeh, Zhongfang Zhuang, Yan Zheng, Liang Wang, Junpeng Wang, and Wei Zhang. Merchant category identification using credit card transactions. In *2020 IEEE International Conference on Big Data (Big Data)*, pages 1736–1744. IEEE, 2020.

[105] Rongwei Yu, Yong Wang, and Wang Wang. Amad: Active learning-based multivariate time series anomaly detection for large-scale it systems. *Computers & Security*, 137:103603, 2024.

[106] Mi Zhang and Alexander A Sawchuk. Usc-had: A daily activity dataset for ubiquitous activity recognition using wearable sensors. In *Proceedings of the 2012 ACM conference on ubiquitous computing*, pages 1036–1043, 2012.

[107] Lixia Zheng, Qianqian Xu, Gu Gong, Yonglin Liao, Min Yu, Sergey Shabala, Wensheng Chen, and Weijian Wu. Nicotiana tabacum as a dead-end trap for adult diaphorina citri: A potential biological tactic for protecting citrus orchards. *Frontiers in Plant Science*, 13:1081663, 2023.

[108] Yan Zhu, Shaghayegh Gharghabi, Diego Furtado Silva, Hoang Anh Dau, Chin-Chia Michael Yeh, Nader Shakibay Senobari, Abdulaziz Almaslukh, Kaveh Kamgar, Zachary Zimmerman, Gareth Funning, et al. The swiss army knife of time series data

mining: ten useful things you can do with the matrix profile and ten lines of code. *Data Mining and Knowledge Discovery*, 34:949–979, 2020.

[109] Yan Zhu, Makoto Imamura, Daniel Nikovski, and Eamonn Keogh. Introducing time series chains: a new primitive for time series data mining. *Knowledge and Information Systems*, 60:1135–1161, 2019.

[110] Christoph Zimmer, Mona Meister, and Duy Nguyen-Tuong. Safe active learning for time-series modeling with gaussian processes. *Advances in neural information processing systems*, 31, 2018.