UNIVERSITY OF CALIFORNIA
RIVERSIDE

Problems with Problems in Data Mining

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

Doctor of Philosophy

in

Computer Science

by

Renjie Wu

June 2024

Dissertation Committee:

Dr. Eamonn Keogh, Chairperson
Dr. Evangelos Papalexakis
Dr. Tao Jiang
Dr. Tamar Shinar

The Dissertation of Renjie Wu is approved:

Committee Chairperson

University of California, Riverside

Acknowledgments

I would like to express my gratitude to my brilliant advisor, Dr. Eamonn Keogh, for his invaluable mentorship and support throughout my PhD journey. Dr. Keogh's commitment to my personal and professional growth has left a long-lasting impact on me and will continue inspiring me to pursue excellence in my future career endeavors.

I would like to thank my committee members, Dr. Evangelos Papalexakis, Dr. Tao Jiang, and Dr. Tamar Shinar, for their time and effort, as well as Dr. Gareth Funning for serving as the oversight committee member in my qualifying exam.

I would like to thank my wonderful colleagues in the UCR Data Mining Lab: Dr. Alireza Abdoli, Audrey Der, Dr. Sadaf Tafazoli, Dr. Ryan Mercer, Maryam Shahcheraghi, Dr. Yan Zhu, Dr. Chin-Chia Michael Yeh, Dr. Zachary Zimmerman, Frank Madrid (R.I.P.), Kaveh Kamgar, Dr. Shima Imani, Dr. Shaghayegh Gharghabi, Dr. Sara Alaee, Dr. Hoang Anh Dau, Dr. Yue Lu, and Dr. Shailendra Singh, as well as lab alumni Dr. Abdullah Mueen.

I would like to thank Dr. Yucheng Zheng for being my closest friend throughout my entire education at UCR, starting from the GPP program, and all the way through the PhD program.

I would like to thank many other friends I met at UCR for being there with me during challenging times throughout my academic journey.

Finally, I would like to express my deepest gratitude to my parents and my sister for their selfless love, support, and encouragement at all times. Their belief in me has been a constant source of motivation for all the achievements I have made so far.

To my parents and my sister,

for all the love, support and encouragement.

To Hatsune Miku,

for all the music, delights and companionship.

ABSTRACT OF THE DISSERTATION

Problems with Problems in Data Mining

by

Renjie Wu

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

Although the term "data mining" did not appear until the 1990s, the process of digging data to discover correlations, patterns and knowledge has a long history. As the rapid growth of data in size and complexity, data mining has augmented manual data processing with automated data analysis assisted by intertwined scientific fields, such as statistics, database systems, and machine learning. However, we unfortunately discovered that several highly cited papers in the field of data mining have surprising problems with their proposed algorithm, datasets, or definition.

- **Not so fast algorithm.** For over two decades, Dynamic Time Warping (DTW) has been known as the best measure to use for most tasks, in most domains. Because the classic DTW takes quadratic time, FastDTW purportedly offers a way to quickly approximate it. The FastDTW algorithm has well over two thousand citations and has been explicitly used in several hundred research efforts. However, we show that in any realistic settings, the *approximate* FastDTW is much slower than the *exact* DTW.

- **Not so good datasets.** In recent years, there has been an explosion of interest in time series anomaly detection (TSAD), driven by the success of deep learning in other domains. Most of these papers test on one or more of popular benchmark datasets from Yahoo, Numenta, NASA, etc. However, we show that the majority of the individual exemplars in these datasets suffer from one or more of four flaws. Because of these four flaws, much of the apparent progress in recent years may be illusionary.

- **Not so clear definition.** Early classification of time series (ETSC) generalizes classic time series classification to ask if we can classify a time series subsequence with sufficient accuracy and confidence after seeing only some prefix of a target pattern. The idea is that the earlier classification would allow us to take immediate action when some practical interventions are possible. However, we show that under reasonable assumptions, no current ETSC algorithm is likely to work in a real-world setting.

In addition to demonstrating our findings, we either provide potential solutions to address these problems, e.g., UCR Time Series Anomaly Archive, or offer recommendations to the community, e.g. specifications for the definition of ETSC.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Although the term "data mining" did not appear until the 1990s in the community, the practice of exploring data to discover correlations, patterns and knowledge has a long history. Early methods, such as Bayes' theorem, have origins dating back to 1700s. As the rapid growth of data in both size and complexity over the past two decades, data mining has augmented manual data processing with automated analysis assisted by intertwined scientific fields, such as statistics, database systems, and machine learning [1].

However, we unfortunately discovered that algorithm, dataset, or definition proposed in several highly cited papers [2, 3, 4, 5, 6, 7, 8] in the field of data mining may suffer from surprising problems. As a result, for example, researchers would have been better off using classic algorithm which would have been much faster. Given the widespread acceptance of these works within the community, we raise concerns about the genuine advances made so far, suggesting that much of the apparent progress in the field of data mining in recent years may be illusionary.

1.1 Not So Fast Algorithm

It has long been believed that the Dynamic Time Warping (DTW) distance measure is the best measure to use for many tasks [9, 10, 11, 12, 13]. Examples of such tasks include similarity search, clustering, classification, anomaly detection and segmentation. DTW reports the distance of two time series after optimally aligning them. DTW is computed by finding the minimum cost path in distance matrix D of two time series X and Y, where $D(i,j) = (X[i] - Y[j])^2 + \min\{D(i-1,j-1), D(i,j-1), D(i-1,j)\}$.

Due to the classic DTW algorithm's quadratic time complexity in relevance to the length of the time series, various ideas have been proposed to reduce its amortized time [10], or to quickly approximate it [11]. One widely cited approximate method is FastDTW [2]. FastDTW asks for a parameter radius (r) and approximates classic (full) DTW by computing DTW on a downsampled version of the data, then iteratively projecting the solution discovered onto an upsampled version and refining it.

The FastDTW algorithm has well over two thousand citations and has been explicitly used in several hundred research efforts. At least five papers use the term FastDTW in their *title* [14, 15, 16, 17, 18]. However, in this work, we make a surprising claim. In any realistic data mining application, the *approximate* FastDTW is much slower than the *exact* DTW. This fact clearly has implications for the community that uses this algorithm: allowing it to address much larger datasets, get exact results, and do so in less time.

1.2 Not So Good Datasets

Time series anomaly detection (TSAD) has been a perennially important topic in data science, with papers dating back to the 1950s [19]. However, in recent years, there has been an explosion of interest in this topic, much of it driven by the success of deep learning in other domains and for other time series tasks. Most of these papers test on one or more of a handful of popular benchmark datasets, created by Yahoo [3], Numenta [4], NASA [5] or Pei's Lab (SMD) [6], etc.

In this work, we make a surprising claim. The majority of the individual exemplars in these datasets suffer from one or more of four flaws. These flaws are *triviality*, *unrealistic anomaly density*, *mislabeled ground truth* and *run-to-failure bias*. For example, we say that a dataset suffers from triviality if can be solved with a single line of code. The implicit of this "one-liner" observation suggests that the great complexity of deep learning may not be warranted. More generally, because of these four flaws, we believe that many published comparisons of anomaly detection algorithms may be unreliable, and more importantly, much of the apparent progress in recent years may be illusionary.

In addition to demonstrating these claims, with this work, we introduce the UCR Time Series Anomaly Archive [20]. We believe that this resource will perform a similar role as the UCR Time Series Classification Archive [13], by providing the community with a benchmark that allows meaningful comparisons between approaches and a meaningful gauge of overall progress. However, we do not consider this work to be the final answer on the subject. We hope this work will inspire the community to take action towards the creation of a crowdsourced set of benchmark datasets.

1.3 Not So Clear Definition

Since its introduction two decades ago, there has been increasing interest in the problem of *early classification of time series* (ETSC). This problem generalizes classic time series classification to ask if we can classify a time series subsequence with sufficient accuracy and confidence after seeing only some prefix of a target pattern. The idea is that the earlier classification would allow us to take immediate action, in a domain in which some practical interventions are possible. For example, that intervention might be sounding an alarm or applying the brakes in an automobile.

In this work, we make a surprising claim. In spite of the fact that there are dozens of papers [7, 8, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37] on early classification of time series, it is not clear that any of them could ever work in a real-world setting. The problem is not with the algorithms per se but with the vague and underspecified problem description. Essentially all algorithms make implicit and unwarranted assumptions about the problem that will ensure that they will be plagued by false positives and false negatives even if their results suggested that they could obtain near-perfect results.

One of our astonishing findings is that we are now two decades and many dozens of papers into this area, there still seems no real-world publicly available dataset(s) dedicated to ETSC task. The overreliance on the proxy datasets and synthetic datasets, such as UCR datasets [13], seems to have led the community astray here. Without real-world publicly available dataset(s), we doubt whether any genuine progress in ETSC has been be made. Considering how easy for a student to obtain seismic data recorded on Mars, it is staggering that every publications on ETSC have to rely on proxy and synthetic datasets.

1.4 Reproducibility Statement

We have taken the greatest care to ensure that all experiments in this work are easily reproducible. To that end, all datasets and code use in this work are archived in perpetuity at [38], [39], and [40] respectively. In the event that we ever discover an issue with this work that makes us temper its claims slightly, we will discuss it on corresponding webpage within forty-eight hours. If we ever discover an issue with this work that significantly affects its claim, we will move to retract the corresponding published papers [41, 42, 43, 44, 45, 46], but will leave all materials on the website to document our error in perpetuity.

1.5 Organization of the Dissertation

The rest of this dissertation is organized as follows. In **Chapter 2**, we investigate FastDTW in four possible settings and show FastDTW is slower in 99% of all use cases, then explain when and why FastDTW fails to approximate well. **Chapter 3** discusses four flaws existed in popular TSAD benchmark datasets and introduces UCR Time Series Anomaly Archive as a potential solution that is largely free of these four flaws. **Chapter 4** reveals false assumptions inherently made by almost all ETSC algorithms, which will be overwhelmed by false positives under real-world settings, due to the unclear definition of ETSC. Finally in **Chapter 5**, we conclude our findings and summarize the impact we have made to the community.

Chapter 2

Not So Fast Algorithm

Many time series data mining problems can be solved with repeated use of an appropriate distance measure. Examples of such tasks include similarity search, clustering, classification, anomaly detection, rule discovery, summarization, and segmentation. It has long been believed that the Dynamic Time Warping (DTW) distance measure is the best measure to use in many domains, and recent extensive empirical "bake-offs" have confirmed this [9, 10, 11, 12, 13].

Because the DTW algorithm has time complexity that is quadratic in the length of the sequences, many ideas have been introduced to reduce its amortized time [10], or to quickly approximate it [11]. One of the most cited approximate approaches is FastDTW [2]. FastDTW works by creating an approximation of classic (full) DTW by computing DTW on a downsampled version of the data, then iteratively projecting the solution discovered onto an upsampled version and refining it.

At least *dozens*, but perhaps as many as *hundreds* of research efforts explicitly adopt FastDTW in order to gain scalability. The quotes below come from some representative works:

- "*In order to expedite the algorithm, we adopted fastDTW in our work*" [47].
- "(to) *minimize the computational complexity, we use a method called FastDTW*" [48].
- "*To increase the speed of the process, we employed a faster version of DTW, called FastDTW*" [49].
- "*FastDTW provides an efficient approximation to DTW*" [50].
- "*We used FastDTW to analyze the recorded accelerometer data for a first implementation of the gesture recognition*" [51].
- "*we use the FastDTW algorithm to automatically identify matching segments*" [52].
- "*the distances between the time series are computed using FastDTW*" [53].

To gauge how commonly used this algorithm is, consider the fact that at least five papers use the term FastDTW in their *title* [14, 15, 16, 17, 18].

In this chapter, we make a surprising claim. In any realistic setting, FastDTW is actually *slower* than DTW. Every paper that we are aware of that uses FastDTW would have obtained faster results by using simple DTW. Moreover, these results would have been exact (by definition), not approximate.

Clearly there are many other papers that made claims that did not pan out with the passage of time. However, the FastDTW paper is very unusual in which the proposed

algorithm is commonly used, especially by people outside the data mining community. That is to say, practitioners in medicine, bioengineering, industry, etc.

Moreover, there are many research efforts whose main goal is to improve FastDTW. For example, a recent paper summarizes its research contribution with: "*In the opinion of the authors, the disparity between the alignment inefficiency of the FastDTW algorithm and that of* (our algorithm) *is especially significant*" [54]. This effort also uses FastDTW as benchmark for accuracy: "*While* (our algorithm) *is slightly less accurate than FastDTW*..." [54]. However, the speed-up reported over FastDTW_5 is a factor of about five for time series of length 150. But as we will show, using off-the-shelf DTW would have dwarfed this apparent speedup.

The rest of this chapter is organized as follows. In **Section 2.1**, we briefly review the necessary background material and notation. **Section 2.2** divides the similarity measurement task into four possibilities, which are empirically investigated. In **Section 2.3**, we show that even if FastDTW were faster than classic DTW, it is not clear to most people when it could fail to give a high-quality approximation. **Section 2.4** explains the assumption made by FastDTW, which leads to its failure in providing good approximation. In **Section 2.5**, we present third party observations that confirm our claims are correct. **Section 2.6** summarizes our claims before we offer conclusions in **Section 2.7**.

2.1 Background and Notation

This review of DTW will be succinct, we encourage the interested reader to consult [9, 10] and the references therein for more information.

DTW reports the distance between two time series after optimally aligning them. DTW is computed by finding the minimum cost path in distance matrix D of two time series X and Y, where $D(i,j) = (X[i]-Y[j])^2 + \min\{D(i-1,j-1), D(i,j-1), D(i-1,j)\}$.

Since at least the 1970s, many practitioners have added *constraints* to the allowable warping paths. These constraints are normally denoted as w, which limits the number of cells (warping window) explored by DTW and the warping path allowed to deviate at most w cells from the diagonal in computing D. Note that most papers report the warping constraint as a percentage of the length of time series, a practice we follow here. In this chapter, we denote DTW with the constraint of w as cDTW_w. Two special cases are worth noting: the case cDTW_0 is equivalent to the Euclidean distance, and the case cDTW_{100} is equivalent to "unconstrained" or "Full" DTW.

It is important to correct a common misunderstanding here, even though a widely cited paper corrected it twenty years ago [9]. Many people still believe that the purpose of using cDTW is to speed up the computation of DTW. However, this is only a happy side effect of using constraints on the warping path. The real purpose of using cDTW is that it is almost always more accurate because it prevents pathological warpings (see [9, 12]). An example of a pathological warping is when, say, a single heartbeat maps onto a dozen heartbeats. The use of cDTW with a suitable value of w, allows a short heartbeat to align to a longer heartbeat, but prevents this meaningless one-to-a-dozen alignment.

FastDTW is an approximation to Full DTW. A full exposition of FastDTW can be found in [2]. Briefly, FastDTW performs three steps recursively with a parameter radius (r). At each level of recursion, FastDTW downsamples the two time series being compared

to half their length. Then FastDTW invokes itself to find the warping path of the two smaller (lower resolution) time series. Finally, a limited DTW is computed at the higher resolution. The warping window is the *neighborhood* of the projected warping path from lower resolution. The size of the neighborhood (i.e. the number of cells away from the projected warping path) is determined by r. Since FastDTW approximates Full DTW, r can be seen as the tradeoff between precision and time: to achieve better accuracy of approximation, a larger r is required; to reduce the running time of the algorithm, a smaller r is necessary.

It is important to restate that w and r are not the same thing. The former is the parameter to give different maximum warping constraints, the latter is a parameter to control the tradeoff between accuracy and speed for FastDTW.

To be clear, we use FastDTW$_r$ for FastDTW with the radius of r and cDTW$_w$ to denote cDTW with the warping window width of w.

In this chapter, we use N to refer to the length of the time series being compared, r for the radius of FastDTW, and w for the user-specified warping constraint, given as a percentage of N. We use W to refer to the natural amount of warping needed to align two random examples in a domain, also given as a percentage of N. This value can be difficult to know exactly, however there are often strong domain hints. For example, when aligning classical music performances, it is clear that there can be differences in timing between performances, however Kwon *et al.* [55] estimated that this is not more than 0.2 seconds. Thus, for a two-minute music performance this would suggest $W = 0.16\%$.

2.2 Four Cases in Similarity Measurement

With our notation established, in **Table 2.1**, we can consider the exhaustive and exclusive matrix of possible settings in which DTW can be used.

Table 2.1: Four settings in which DTW can be used

N gets larger $\rightarrow$		
	Case B Music performance, classical dance performance, seismic data	**Case D** <no obvious applications>
	Case A Heartbeats, gestures, signatures, golf swings, gene expressions, gait cycles, star-light-curves, sign language words or phrases, bird song	**Case C** Residential electrical power demand

W gets larger $\rightarrow$

The boundaries between these four cases are somewhat subjective. For our purposes we will say that N transitions from short to long somewhere around 1,000, and that W transitions from narrow to wide somewhere around $W = 20\%$.

We can now consider the utility of FastDTW for each of these cases.

2.2.1 Case A: Short N and Narrow W

For this case, cDTW is unambiguously faster. Moreover, the original authors echo this point, writing in 2020 that "*If* (W) *is known beforehand to be* (small)*, I recommend cDTW and do not recommend fastDTW.*" [56].

To show how slow FastDTW can be compared to a vanilla iterative implementation of cDTW, consider **Figure 2.1**. Here we consider the UWaveGestureLibraryAll dataset from the UCR archive [13], which has exemplars of length 945, towards the long end of Case A. We consider all values of w from 0 to 20%.

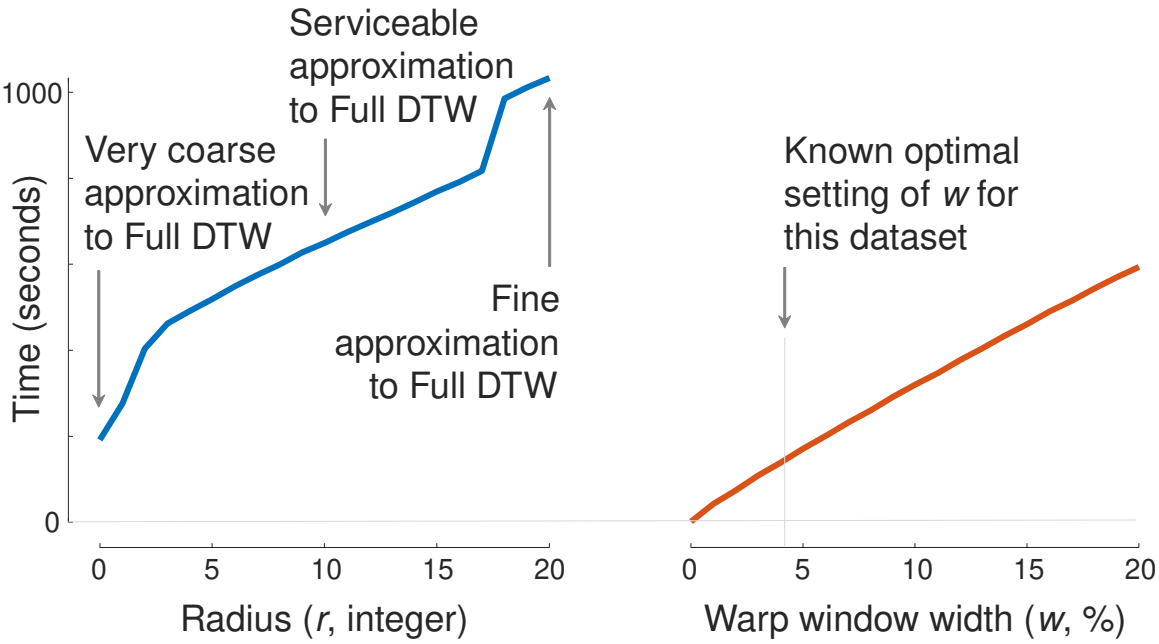

Figure 2.1: A comparison of the time needed to compute all pairwise distances of the 896 training examples in UWaveGestureLibraryAll, (which requires $(896\times895)\div2 = 400{,}960$ comparisons) for $r = 0$ to 20 for FastDTW (*left*) and for $w = 0$ to 20% for cDTW (*right*).

In this case, we know the best value of W for this dataset, at least in the context of classification. The UCR archive notes that the error rate of cDTW_0 (i.e. Euclidean distance) is 0.052, that cDTW_4 minimizes the error to 0.034, and that cDTW_{100} (i.e. Full DTW or unconstrained DTW) has a much higher error rate of 0.108 [13].

It is worth discussing those results. The classification error rate of Full DTW is much higher than the error rate of constrained DTW. This continues to surprise people, but it has been known since at least 2004 [9]. It is sometimes referred to as the Ratanama-

hatana's observation that "*a little warping is a good thing, but too much warping* (can be) *a bad thing*" [9].

It is important to recall that the two algorithms being compared are both implemented in the same language, running on the same hardware, performing the same task.

It is also important to state that we did not use any optimizations for cDTW. It is well known that when doing repeated measurements of DTW, say to find an object's nearest neighbor or to do nearest neighbor classification, one can avail of both lower bounding and early abandoning [9, 57]. Moreover, these ideas have be carefully optimized by the community for DTW. Using these ideas would have shaved at least two further orders of magnitude off the time for cDTW.

Note that our **Figure 2.1** (*left*) annotations of how well FastDTW approximates Full DTW are taken from the original paper. We do not make any comment on the quality of approximation here, other than to say that we assume the original claims are true. Thus **Figure 2.1** shows that for the optimal setting of w for this dataset, cDTW_4 is faster than the coarsest and fastest version of FastDTW. Moreover, even if we insisted on setting a larger value of w, up to 20, we can still exactly compute cDTW_{20} as fast as we can compute a serviceable approximation to Full DTW, by using FastDTW_{10}. Thus, this experiment provides forceful evidence that at least for Case A, FastDTW is slower than using cDTW.

We believe that at least 99% of all uses of DTW in the literature fall into this case. One way to see this is to consider the distribution of N and W for the 128 datasets in the UCR archive [13]. This archive is clearly not a perfect representation of all datasets, all domains, and all problems. However, it is the largest such collection of labeled time series

data in the world, and the optimal setting of w (which is our proxy for W) that maximizes classification accuracy, was computed by brute-force search [13]. **Figure 2.2** summarizes the data.

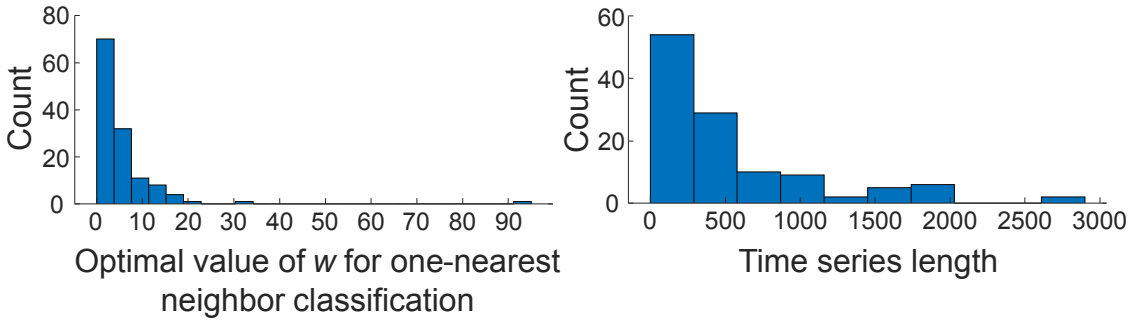

Figure 2.2: (*left*) The distribution of optimal values for w, for the task of one-nearest neighbor classification, for 128 datasets. (*right*) The distribution of the lengths of these datasets.

These histograms show that majority of time series subsequences considered are less than 1,000 datapoints, and more importantly, the best value for w is rarely above 10%. Almost all uses of FastDTW fall into Case A [14, 15, 16, 17, 18, 47, 48, 49, 50, 51, 52, 53], and in every case the researchers using FastDTW would have been better off using classic cDTW, which would have been much faster, and *exact*.

Thus, the vast majority of readers of this work, who are reading this chapter to decide if they should use cDTW or FastDTW, can stop reading here. Both the current author, and the original authors of FastDTW [56], are recommending that you use cDTW. There is no disagreement or ambiguity for this case.

2.2.2 Case B: Long N and Narrow W

Case B considers the possibility of long time series with a low value for W. We have already hinted at one such possibility, musical performances, where the task is sometimes called score following or score alignment. Is cDTW faster here?

Let us perform an experiment. Imagine we align the exactly four-minute long song "Let It Be" with a live version. For classical music, various papers have suggested values such as $W = 0.16\%$ [55]. Let us be much more liberal and assume that the live version can be up to two seconds ahead or behind at some point[1]. Thus, we set $w = 0.83\%$. Music processing typically uses Chroma Features, which are normally sampled at 100Hz, thus we have a times series of length 24,000. To obtain a robust estimate we measured the time required for each algorithm one thousand times reporting the average. We find that:

- $\text{cDTW}_{0.83}$ takes 45.6 milliseconds.
- FastDTW_{10} takes 238.2 milliseconds.
- FastDTW_{40} takes 350.9 milliseconds.

Thus, for Case B we find no evidence of the utility of FastDTW.

2.2.3 Case C: Short N and Wide W

Case C considers the case where the N is short (say <1,000 data points), but W is large. There are no examples in the UCR archive, and a search of the literature does not suggest examples. However, we have a large collection of datasets, and after a significant effort, we managed to create a somewhat contrived situation/dataset.

[1] With apologies to Sir Paul McCartney, who has superb timing.

Imagine that a researcher decides to compare the first hour of electrical power demand each day in a residence (i.e. from midnight to 1am). Most of the time these would not be very similar under any measure. However, as shown in **Figure 2.3**, we occasionally encounter a pattern that is similar, but only under the assumptions of Case C, where N is reasonably short (here 450 datapoints) but W is a large fraction of this value. Note that if we just use one pattern as a query on a sliding window of the entire year-long trace, the value of W would dramatically decrease, and we would be back in Case A.

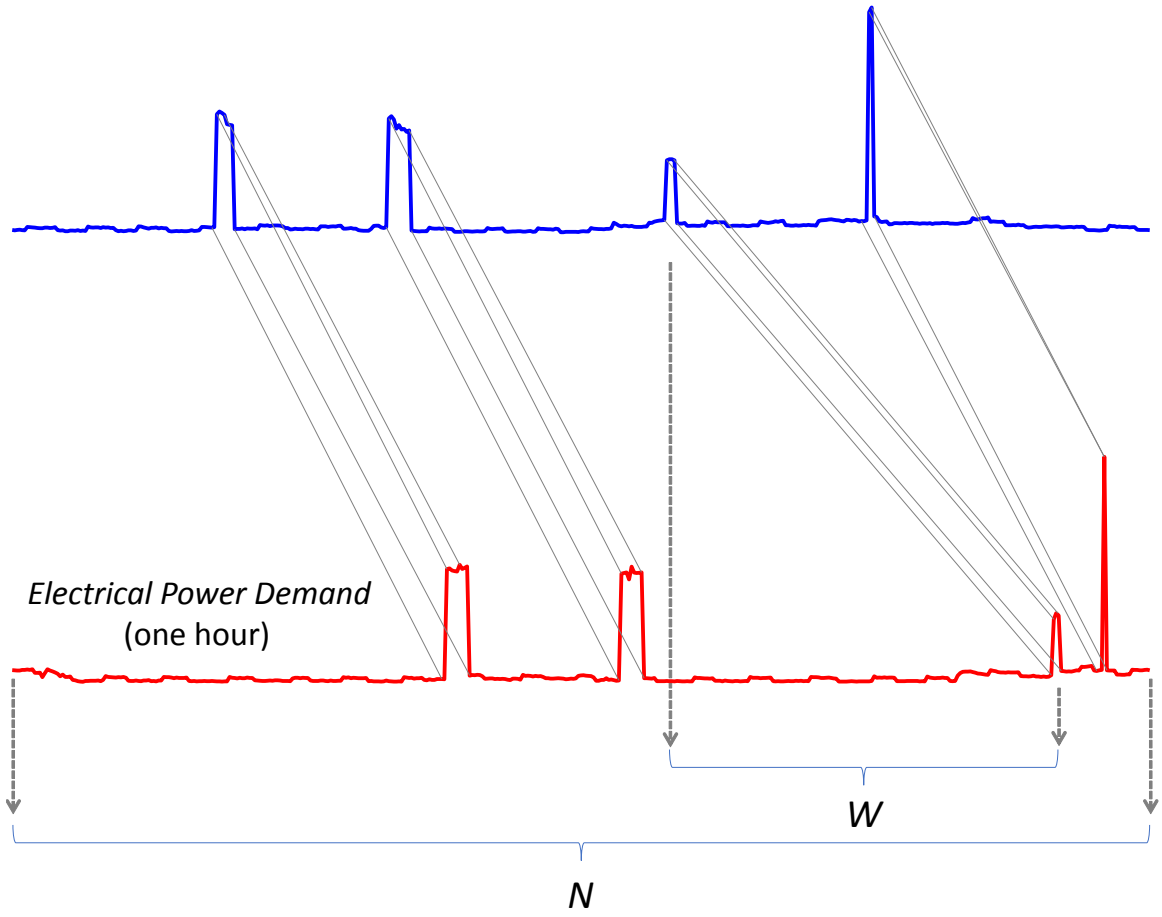

Figure 2.3: An example of a case where W is a large faction of N. Two examples of the electrical power demand from midnight to 1:00 AM, sampled once per eight seconds. This conserved pattern reflects the program of a dishwasher. The owner may have programmed it to run after midnight, when the electrical power costs are cheaper in the UK.

Let us use the electrical power demand to motivate Case C. The natural value of W here is estimated by looking at the *maximum* difference in timing between corresponding pairs of peaks. This happens for the third pair, which differ by 153 datapoints. Given that time series are of length 450, that gives us an estimate of $W = 34\%$, which to be conservative, we will round up to 40%. Thus, in **Figure 2.4**, we repeat the type of experiment shown in **Figure 2.1**, but consider time series of length 450, and w from 0 to 40%.

Since the timing for both algorithms does not depend on the data itself, we use random walk datasets.

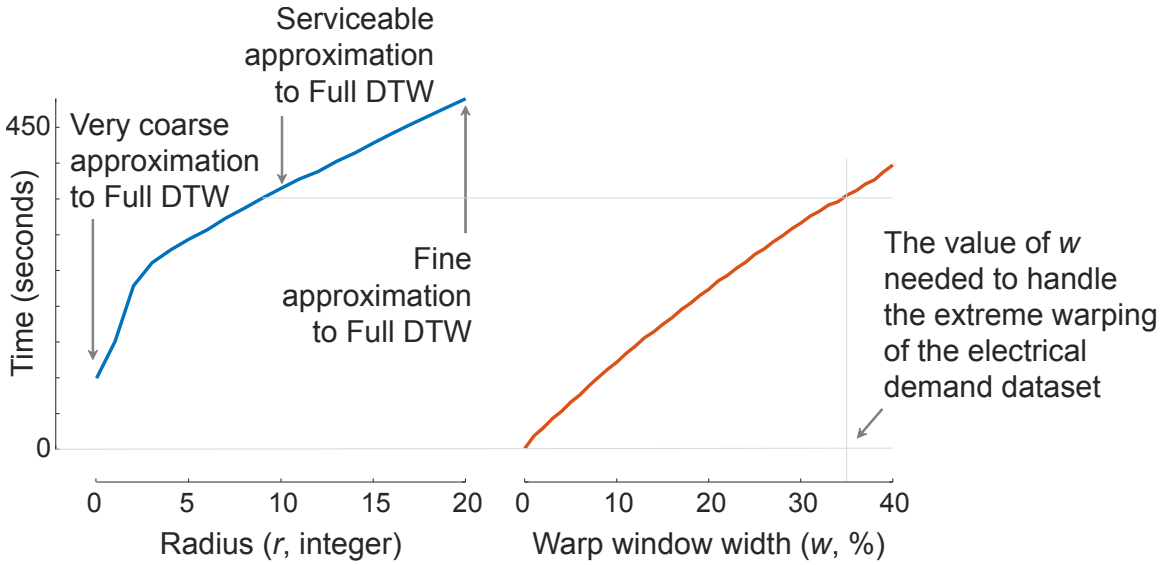

Figure 2.4: The experiment shown in **Figure 2.1** generalized to consider warping window width up to 40%. The time is the cumulative time needed for all pairwise comparisons in a dataset of 1,000 examples (499,500 comparisons).

A recent paper also observed that FastDTW does not achieve expected time optimization in Case C. The authors conclude "*this is because* (our time series) *are usually short, which makes the cost of* (FastDTW) *exceed the cost of calculating DTW directly*" [58].

Note that as with Case A, we resisted the temptation to do any of the optimizations available only to cDTW when doing multiple comparisons. This is a straightforward head-to-head comparison. Thus, for Case C we find no evidence of the utility of FastDTW.

2.2.4 Case D: Long N and Wide W

Case D is the case emphasized by the original authors of the FastDTW paper as the best case for their algorithms. However, they did not show any real-world examples of such datasets, and a (admittedly incomplete) survey of the papers that refence FastDTW does not show any examples in the literature [14, 15, 16, 17, 18, 47, 48, 49, 50, 51, 52, 53].

We claim that there are no practical applications of such comparisons. There are simply no problems for which we need to compare time series of this length with a large value for w. Of course, it is hard to prove a negative, but consider:

- Modern electrocardiograms can record data at rates of up to 25,000Hz [59]. However, there have been numerous studies that ask, "*what is the minimum sampling rate we need for* (some cardiological problem)?" The answer is typically around 250Hz [60]. This means that to compare two heartbeats, we need to compare about 120 to 200 datapoints. Does it ever make sense to compare longer regions of ECGs? No. To see why, imagine comparing two one-minute long ECGs. Such traces would have about 100 beats, but it is very unlikely that they would have the exact same number. While DTW is forgiving of misalignments, it must explain all the data. It is never meaningful to compare say 98 heartbeats to 103 heartbeats. Thus, we believe that all uses of DTW for cardiology are in Case A.

- A recent exhaustive empirical study asked a similar question to the above ECG study in the context of gesture recognition. It was discovered that "*recognition rates for $N = 32$*, (are) *not significantly different than those delivered by higher rates*" [61]. This reflects a complex twenty-five gesture, user-independent study. Perhaps there is some circumstance in which we need a greater N for gestures. Perhaps Asians have more nuanced gestures than the tested Europeans. However, this study (and many similar studies) strongly suggest that there is little utility in comparing more than a few hundred data points for human gestures, gait cycles, sport performances, etc.

- As noted above, the UCR archive has 128 datasets, many culled from real-world problems. The longest of these is 2,844. However, even for the handful of long time series, the long length is typically just an artifact of how it was recorded. We can downsample most of this time series by a factor of eight or more, and get an accuracy that is statistically significantly the same.

In summary, to the best of our knowledge, there is no evidence that it is ever useful to compare time series with lengths exceeding (conservatively) 1,000. Of course, absence of evidence is not evidence of absence. However, it is clear that at a minimum, this is a very rare case.

Nevertheless, for completeness we *do* test this case.

We consider a contrived "fall" dataset. Suppose that a researcher was investigating falls by having actors wearing a motion capture suit fall over in a safe environment. Further imagine she instructs actors to "*Fall over anytime within two seconds of hearing the beep*". Assume she does not crop and clean the data, but simply measures the distance between

two-second snippets, which were recorded at 100Hz. Knowing this, we can assume that in this domain, $W \approx 100\%$.

Instead of two seconds, let us generalize to L seconds. As shown in **Figure 2.5**, we created a data generator that creates pairs of time series of length L seconds at 100Hz. One time series has an immediate fall, then the actor is near motionless for the rest of the time. For the other time series, the actor is near motionless until just before L seconds are up, then he falls.

It is clear that for cDTW to align the two falls, we must use cDTW_{100}.

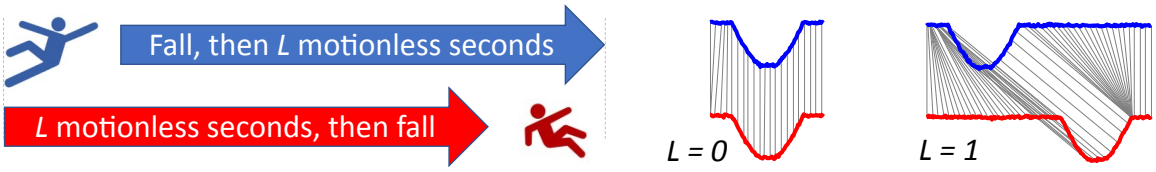

Figure 2.5: (*left*) We model the task of aligning early and late falls in a L-second long interval. (*right*) The cDTW alignment for $L = 0$ and $L = 1$, are both examples of Case C, but for large enough values of L, we begin to move to Case D.

Note that we do not test to see if FastDTW_{40} actually aligns the two falls, we simply assume it does.

We can now create pairs of time series of increasing values of L and discover at what point FastDTW_{40} becomes as fast as cDTW_{100}. **Figure 2.6** shows the results.

Thus, we have finally found a circumstance where FastDTW_{40} is faster than cDTW_{100}. Note that at this breakeven point FastDTW_{40} is an *approximation* to cDTW_{100}, so cDTW_{100} is still preferable. However, as L grows well beyond the transition point, each user needs to consider the tradeoff between the time taken vs. utility of approximation.

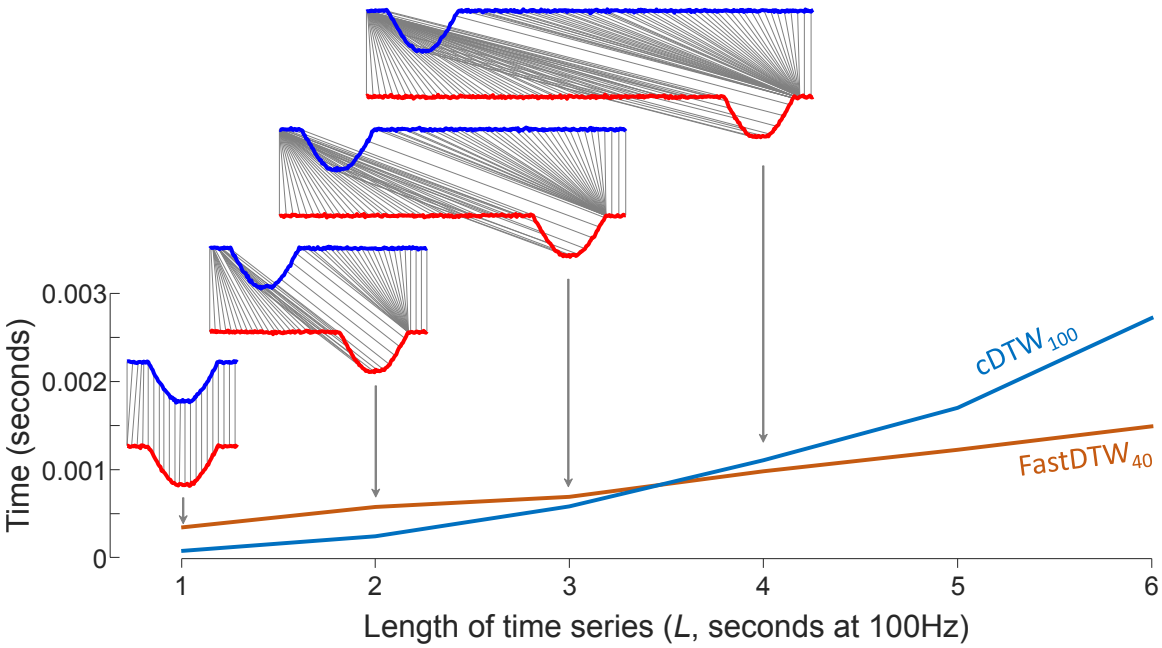

Figure 2.6: As we make L longer and longer, we find that when $L = 4$ ($N = 400$), FastDTW finally becomes faster than unconstrained cDTW (or cDTW_{100}). For each L, the time is measured by running each algorithm 1,000 times, and reporting the average.

The full answer to that question is beyond the scope of this chapter, and in any case depends upon the domain, the analytic task and the cost of an error. For example, if we were comparing some data that we were confident was really in Case D, and we worked out that for the values of N, w and r, FastDTW_r would be ten times faster, and there was little consequence of using an approximation, we might well decide to use FastDTW_r. However, suppose the data in question came from Mars, or from an intrusive, expensive and time-consuming medical biopsy. In these cases, it would be more difficult to justify an approximation, even if it gives you a tenfold speedup.

Another issue is the *magnitude* at which the hypothetical tenfold speedup occurs. There is a real tangible difference between one day and ten days. However, for most practical purposes there is simply no difference between 0.01 seconds and 0.1 seconds. The reader

might imagine that *repeated* use of comparisons could allow these short amounts of time to add up to the one/ten days situation: for example, for similarity search or classification. However, for repeated uses of DTW, there are several ideas that can *only* be applied to cDTW, including lower bounding, early abandoning, just-in-time normalization, etc. These ideas accelerate cDTW by a further two to five orders of magnitude [10, 12]. For example, for similarity search of a $cDTW_5$ query of length 128, using a 2012 machine, Rakthanmanon *et al.* [10] searched a time series of length one trillion in 1.4 days, while using a modern machine, $FastDTW_{10}$ would take 5.8 years[2]. Likewise, to create the UCR archive [13], Hoang Anh Dau computed cDTW 61,041,100,000,000 times, all on an off-the-shelf desktop, something that would be simply inconceivable with FastDTW.

2.3 When Does FastDTW Fail to Approximate Well?

In this chapter, we have mostly refrained from measuring the *accuracy* of the FastDTW approximation. Partly this is because we have shown that in almost all cases, it is a moot point. In addition, this question opens a pandora's box of what the appropriate measure of quality of approximation is?

Nevertheless, it is instructive to consider one example. We created three time series, and as shown in **Table 2.2**, we measured their pairwise distances, using these distance matrices to create the dendrograms shown in **Figure 2.7**.

[2] Averaged over a million comparisons, we found $FastDTW_{10}$ takes 0.1845 milliseconds for $N = 128$, and $10^{12} \times 0.1845$ milliseconds = 5.8 years.

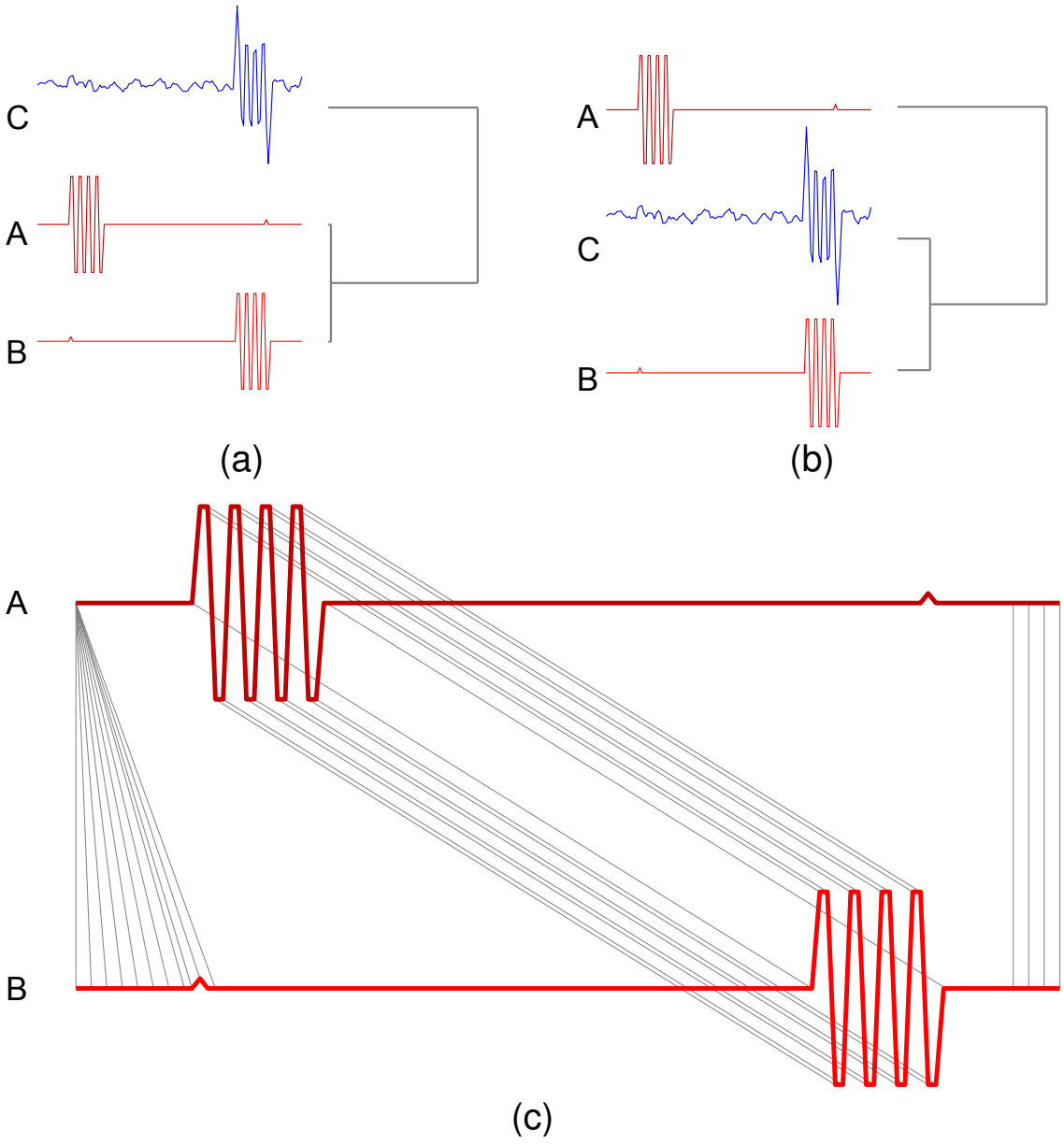

Figure 2.7: A clustering of three time series under Full DTW (a) and under FastDTW$_{20}$ (b). (c) The Full DTW alignment between A and B (only selected hatch lines are shown for clarity) shows that they are almost identical if we allow unconstrained warping.

Table 2.2: The distance matrices for the three time series shown in **Figure 2.7** under Full DTW and FastDTW$_{20}$

Full DTW

	A	B	C
A	0	0.02	6.822
B		0	6.848
C			0

FastDTW$_{20}$

	A	B	C
A	0	31.24	6.822
B		0	6.848
C			0

The two time series A and B clearly require significant warping. However, as **Figure 2.7** (c) shows, given unconstrained freedom to warp, they are almost identical, differing only by 0.02. However, FastDTW$_{20}$ finds them to be 31.24 apart. Using the error metric proposed in the original FastDTW paper [2], this is an error of 156,100%.

In a sense, this example is unfair to FastDTW$_{20}$. For any distance measure or approximation or upper/lower bound to a distance measure, if you understand how the technique works, you can create synthetic examples that will defeat it. This typically says little or nothing about how likely you are to encounter such adversarial examples in the real world. However, we show this example to make the following point. There appears to be no literature that considers under what conditions FastDTW can fail (and therefore, when to avoid using it). Without such an understanding, practitioners (who are in Case D) may wish to step back and reexamine the trade-offs they are making.

2.4 Why FastDTW Fails?

The result shown in **Figure 2.7** struck some early readers of this chapter as so extraordinary they assumed it was an error on our part. Thus, for completeness, we show how we made this example.

FastDTW assumes that the low dimensionality version of a time series has the same basic shape as the raw data under Piecewise Aggregate Approximation (PAA). This is a reasonable assumption, but as with any dimensionality reduction technique, the pigeonhole principle tells us there must be examples of objects that are poorly represented in lower dimensionality representation. Suppose that poor approximation for a pair of objects has the property that DTW warps it in the opposite direction to the original data. As **Figure 2.8** shows, our pair of time series have exactly that property.

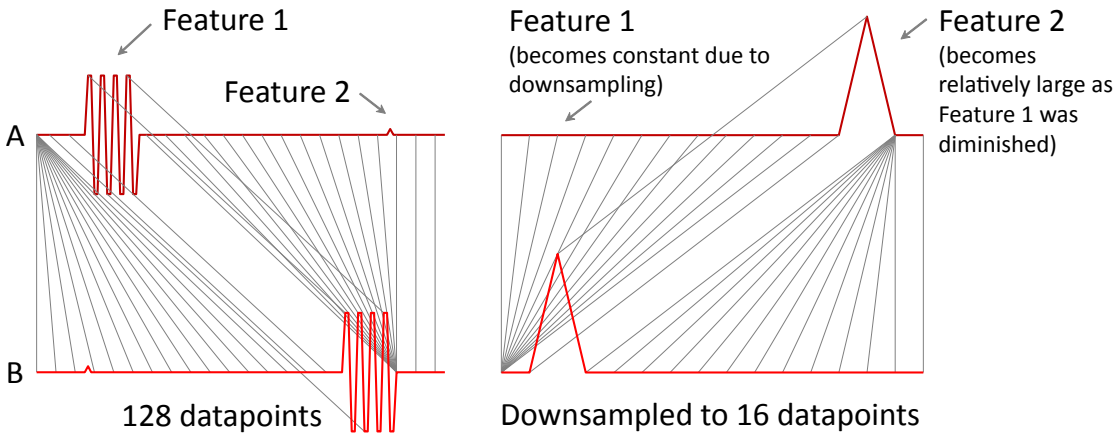

Figure 2.8: (*left*) The two time series shown in **Figure 2.7** optimally warped by DTW. (*right*) The eight-to-one PAA downsampled version of the time series depresses the important features and (relatively) magnifies a tiny feature that warps in the opposite direction to the original time series. It is this "wrong way" warping that is passed up to a finer resolution for refinement.

Once the low resolution approximation of FastDTW has committed to warping in the wrong direction, it cannot recover in the higher resolutions, because the parameter r excludes reaching the correct warping path.

2.5 Independent Confirmations of Our Claims on FastDTW

We wrote to several authors that had recently used FastDTW and asked them if they would be willing to re-run their experiments using cDTW. At the time of going to press, we received one reply, from the authors of [62]. Below is the reply, edited for brevity (full text at [63]).

> *I reran the main experiment from our paper with the newest re-leased version of the "fastdtw" package for Python (version 0.3.4) as well as with the implementation you provided me (using radius=30 for both).*
>
> *- Using FastDTW reproduced the same results we originally published (77.38% of gestures correctly classified)*
>
> *- Using your version improved the results of our classifier by about 5% (82.14% of gestures correctly classified)*
>
> *I also compared the runtime of the two DTW implementations during the experiment. I ran the whole experiment twice, which amounts to a total of $2 \times 5.851 = 11.702$ runs of each DTW implementation for which I compared the run times.*
>
> *Result:*
>
> *- Your implementation was approx. 24x faster than FastDTW on average (mean: 23.7059, std: 3.587)*
>
> *- In the "slowest" case, your implementation was still approx. 5.8x faster than FastDTW*
>
> *I'd conclude that these tests suggest that your implementation is indeed superior in terms of speed as well as for usage in time-series classification.*

Several recent papers [58, 64, 65, 66, 67, 68, 69] also observed that FastDTW does not achieve expected performance. Thus, we have at least eight confirmations from a third party that our claims are correct.

2.6 Summary

We have shown that the vast majority of the researchers that used FastDTW would have been better off simply using cDTW. They would have found simple cDTW to be both faster and to produce exact results.

This chapter was not written as part of a game of one-upmanship. It really is the case that there are lost opportunities here, and the community should be aware of this issue. Consider the recent paper [62] which notes "*We employ the FastDTW method by Salvador and Chan...*". The paper shows promising results in gesture recognition, but then ends on the pessimistic question, "*how* (can) *this method can be sped up, desirably up to the point where it reaches real-time capability*". However, for at least the last decade, it was already possible to achieve at least ten thousand times faster real-time performance on their task[3].

2.7 Conclusion

We have shown that a commonly used tool to accelerate time series data analytics does not actually achieve speed-up in any realistic setting. We discovered this issue be-cause Salvador and Chan took enormous efforts to make their code available, to clearly explain their approach in their paper, and because they were incredibly responsive to the many questions we asked them. We are extremely appreciative of their assistance.

[3] Schneider *et al.* [62] conclude by bemoaning the inability to do real time gesture monitoring with FastDTW. They have 36 channels corresponding to different body parts sampled at 30Hz. Let us assume the longest gesture takes two seconds. Can we monitor thirty-six 60-datapoint queries in real time under DTW? Twelve years ago, Rakthanmanon *et al.* [10] showed they could monitor a 128-datapoint query at about 6 million Hz. To demonstrate this visually, they produced a video [57] that shows they could monitor a query heartbeat of length 421 at about thirty thousand times faster than real time.

Chapter 3

Not So Good Datasets

Time series anomaly detection (TSAD) has been a perennially important topic in data science, with papers dating back to the dawn of computer science [19]. However, in the last eight years, there has been an explosion of interest in this topic, with at least one or two papers on the topic appearing each year in virtually every database, data mining and machine learning conference, including SIGKDD [5, 6], ICDM [70], ICDE [71], VLDB [72], etc.

A large fraction of this increase in interest seems to be largely driven by researchers anxious to transfer the considerable success of deep learning in other domains and from other time series tasks such as classification.

Most of these papers test on one or more of a handful of popular benchmark datasets, created by Yahoo [3], Numenta [4], NASA [5] or Pei's Lab (SMD) [6], etc. In this chapter, we make a surprising claim. The majority of the individual exemplars in these datasets suffer from one or more of four flaws. These flaws are *triviality*, *unrealistic*

anomaly density, *mislabeled ground truth* and *run-to-failure bias*. Because of these four flaws, we believe that most published comparisons of anomaly detection algorithms may be unreliable. More importantly, we believe that much of the apparent progress in recent years may be illusionary.

For example, Qiu *et al.* [73] introduce a "*novel anomaly detector for time-series KPIs based on supervised deep-learning models with convolution and long short-term memory (LSTM) neural networks, and a variational auto-encoder (VAE) oversampling model.*" This description sounds like it has many "moving parts", and indeed, the dozen or so explicitly listed parameters include: convolution filter, activation, kernel size, strides, padding, LSTM input size, dense input size, softmax loss function, window size, learning rate and batch size. All of this is to demonstrate "*accuracy exceeding 0.90* (on a subset of the Yahoo's anomaly detection benchmark datasets)." However, as we will show, much of the results of this complex approach can be duplicated with a single line of code and a few minutes of effort.

This "one-line-of-code" argument is so unusual that it is worth previewing it before we formally demonstrate it in **Section 3.1.2** below. Almost daily, the popular press vaunts a new achievement of deep learning. Picking one at random, in a recent paper [74], we learn that deep learning can be used to classify mosquitos' species. In particular, the proposed algorithm had an accuracy of 97.8% when distinguishing *Aedes vexans* from *Culex triaeniorhynchus*. Should we be impressed? Dr. Keogh has significant computational experience working with mosquitos, and he *is* impressed.

Suppose however that someone downloaded the original 1,185 images from the study and showed that they could classify them with 100% accuracy using a single line of code[4]. If that happened, there are two things we can confidently say:

- We would not for one moment imagine that the one line of code had any particular value as a classifier. We would assume that this was some kind of "trick". Perhaps the *Aedes* images were in JPEG format and the *Culex* images were in GIF format. Or perhaps one species was recorded in color, and the other in B/W. Something interesting is clearly happening, but it is surely not the case that a useful entomological image classification algorithm takes a single line of code.

- We would have lost some confidence in the original paper's results. It is still likely that the paper is genuinely doing something useful. However, we would all be a lot happier trusting the paper's contribution if the authors released a statement to the effect of "*we converted all files to JPEG format, and all images to 16-bit B/W, and reran the experiments getting similarly good results. Moreover, we are confident that our new publicly released dataset will now not yield to a single line of code algorithm*".

This is a perfect analogy of our one-line-of-code argument. Our ability to produce "one-liners" for most datasets does not mean that the original papers that tested on these datasets are not making a contribution. However, at a minimum, it does *strongly* suggest that the community needs to regroup, and test on new datasets that would generally stump trivial one-line solutions.

[4] To be clear, we choose this example because it was the first hit for a Google search for "novel deep learning applications". We have no reason to doubt the claims of this paper, which we only skimmed.

Before continuing, it is important to note that our discussion of some issues with the benchmark datasets should in no way be interpreted as criticism of the original introducer of these datasets.

These groups have spent tremendous time and effort to make a resource available to the entire community and should rightly be commended. It is simply the case that the community must be aware of the severe limitations of these datasets, and the limitations of any research efforts that rely upon them.

3.1 A Taxonomy of Benchmark Flaws

Before discussing the four major flaws found in many public archives, we will briefly discuss related work, to put our observations into context.

3.1.1 Related Work

The literature on anomaly detection is vast [75, 76], with a particular increase in works in just the last five to ten years [3, 4, 5, 70, 73, 77, 78, 79, 80, 81]. We refer the interested reader to [72] and [76] which offer the reader a detailed review and taxonomy.

Almost all these works test on one or more public datasets created by a handful of groups, including Yahoo [3], Numenta [4], NASA [5] or SMD [6], etc. Some papers test on these public datasets in addition to a private dataset. In many cases, the authors do not even show a plot of any data from the private datasets. Thus, here we can clearly make no claims about such private datasets, other than to note that the use of private datasets thwarts the community's laudable move to reproducibility. In addition, the use of private

datasets will always be accompanied by the possibility of unconscious cherry-picking that the reader or the reviewer will never know about.

There is a strong implicit assumption that doing well on one of the public datasets is a sufficient condition to declare an anomaly detection algorithm useful (and therefore warrant publication or patenting). Indeed, this assumption is stated explicitly in many works, for example Huang [82] notes "(The Yahoo) *A1Benchmark is undoubtedly a good time-series dataset for testing the general effectiveness of an anomaly detection method*", and Gao *et al.* [83] gush that "*Yahoo data set has a good coverage of different varieties of anomalies in time series, such as seasonality, level change, variance change and their combinations.*" However, we are not aware of any work that has treated this assumption critically.

In the following four sections, we will introduce four issues with these public datasets that we believe throws doubt on the assumption that they are suitable for comparing algorithms or gauging progress in time series anomaly detection.

3.1.2 Triviality

A large fraction of the problems in the benchmark datasets are so simple to solve that reporting success in solving them seems pointless or even absurd. Of course, *trivial* is not a well-defined word, so, to firm up our claim we will make a practical testable definition:

Definition 1. A time series anomaly detection problem is *trivial* if it can be solved with a single line of standard library MATLAB code. We cannot "cheat" by calling a high-level built-in function such as *kmeans* or *ClassificationKNN* or calling custom written functions. We must limit ourselves to basic vectorized primitive operations, such as *mean*, *max*, *std*, *diff*, etc.

This definition is clearly not perfect. MATLAB allows nested expressions, and thus we can create a "one-liner" that might be more elegantly written as two or three lines. Moreover, we can use unexplained "magic numbers" in the code, that we would presumably have to learn from training data. Finally, the point of anomaly detectors is to produce purely automatic algorithms to solve a problem. However, the "one-liner" challenge requires some human creativity (although most of our examples took only a few seconds and did not tax our ingenuity in the slightest).

Nevertheless, our simple definition gets at the heart of our point. If we can quickly create a simple expression to separate out anomalies, it strongly suggests that it was not necessary to use several thousands of lines of code and tune up to a dozen parameters to do it.

Perhaps the best way to see this is to imagine that we give the same challenge to create a "one-liner" for differentiating protein-coding and noncoding RNA [80], or we had the challenge of separating positive vs negative Yelp reviews. Both of these are also one-dimensional problems on which deep learning appears to have made significant progress in recent years [80]. However, it seems inconceivable that the bioinformatic or text datasets considered in the literature could be teased apart with a single line of code, no matter how contrived. These are intrinsically hard problems, and the communities working on them are using intrinsically challenging datasets.

To illustrate our point, consider **Figure 3.1**, which shows an example from the SMD dataset [6]. The example is a multiple-dimensional dataset, here we consider only dimension 19.

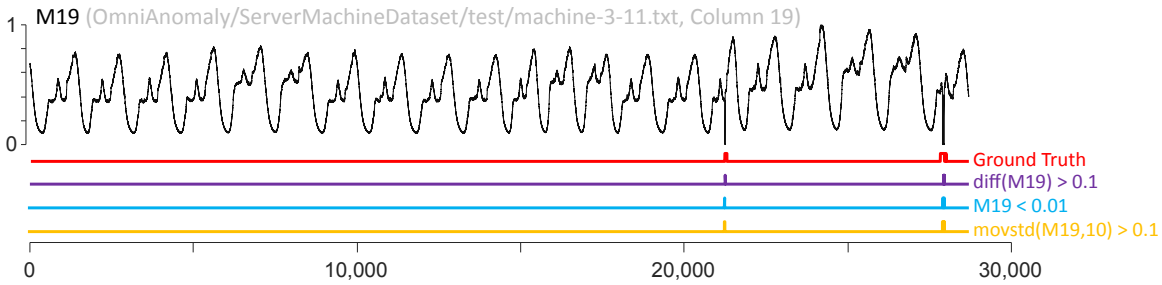

Figure 3.1: (*top to bottom*) Dimension 19 from SMD3-11 dataset. A binary vector (red) showing the ground truth anomaly labels. Three examples of "one-liners" that can solve this problem.

There are dozens of simple one-liners that solve this problem. In the figure, we show three representative examples.

Let us take the time to preempt some possible objections to this demonstration:

- *All the one-liners have a parameter.* True, but recall that most anomaly detection algorithms, especially ones based on deep learning, have *ten* or more parameters. Moreover, the results here are not particularly sensitive to the parameter we set.

- *The choice of dimension was cherry-picked.* We deliberately chose one of the *harder* of the 38 dimensions here. Most of the rest are even *easier* to solve.

- *The choice of problem was cherry-picked.* Of the twenty-eight example problems in this data archive, the majority are this easy to solve with one-liners.

- *The fact that you can solve this problem in one line, does not mean that other algorithms that are successful in this dataset are not useful.* True, we have acknowledged that point in multiple places in this chapter and are happy to do so again here.

The second most cited benchmark is Numenta [4]. The Numenta archive is commendably diverse, however most of the examples, like the one shown in **Figure 3.2**, readily yield to a single line of code.

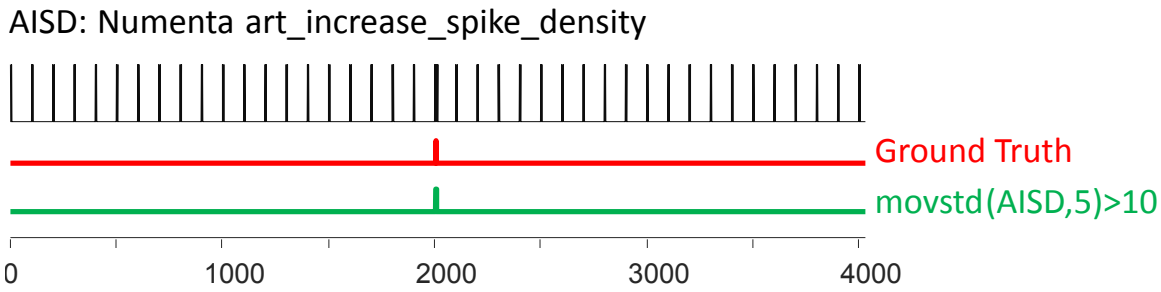

Figure 3.2: (*top to bottom*) The Numenta Art Increase Spike Density datasets. A binary vector (red) showing the ground truth anomaly labels. A "one-liner" (green) that can solve this problem.

We will not even bother to show any examples from the NASA dataset (the interested reader can view many examples on [39]). In about half the cases, the anomaly is manifest in many orders of magnitude difference in the value of the time series. Such examples are well beyond trivial.

Other NASA examples consist of a dynamic time series suddenly becoming exactly constant (see in **Figure 3.9**). For those examples, we can flag an anomaly if, say, three consecutive values are the same, with something such as `diff(diff(TS)) == 0`.

Having said that, perhaps 10% of the examples in the NASA archive are mildly challenging, although even those examples do not need to avail of the power of deep learning, as the yield to decade-old simple ideas [84, 85].

The Yahoo archive [3] is by far the most cited in the literature. It contains a mixture of real and synthetic datasets. Let us consider the first real dataset, which happens

to be one of the more challenging examples (at least to the human eye). However, as **Figure 3.3** shows, it readily yields to a one-liner.

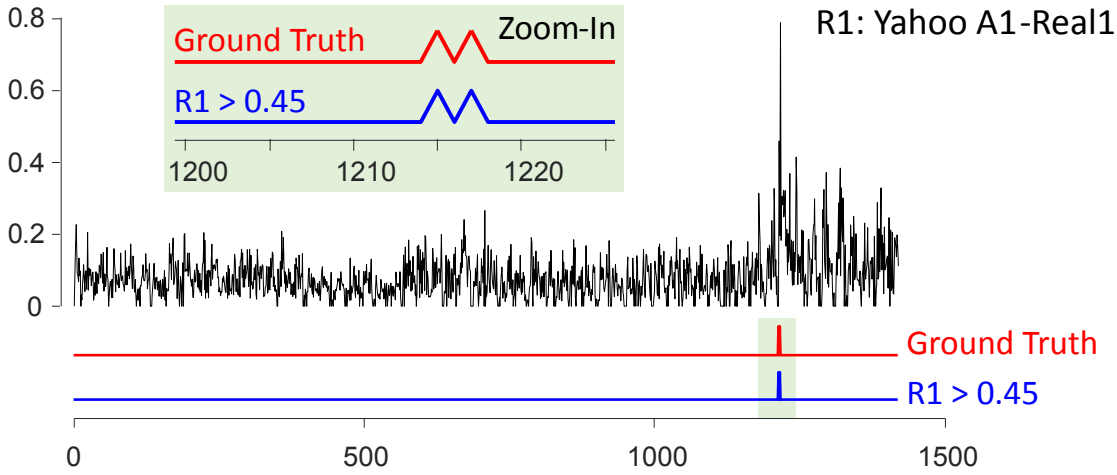

Figure 3.3: Yahoo A1-Real1. A binary vector (red) showing the ground truth anomaly labels. An example of a "one-liner" (blue) that can solve this problem. A zoom-in shows how precisely the simple one-linear can match the ground truth.

Lest the reader think that we cherry-picked here, let us consider the *entire* Yahoo Benchmark [3]. There are 367 time series in the Yahoo Benchmark; most of them can be solved with a *universal* one-liner (3.1) or (3.2):

$$\begin{aligned} abs(\text{diff}(\mathbf{TS})) > \; & \mathbf{u} \times \text{movmean}(abs(\text{diff}(\mathbf{TS}), \mathbf{k}) \\ & + \mathbf{c} \times \text{movstd}(abs(\text{diff}(\mathbf{TS})), \mathbf{k}) \\ & + \mathbf{b} \end{aligned} \tag{3.1}$$

$$\begin{aligned} \text{diff}(\mathbf{TS}) > \; & \mathbf{u} \times \text{movmean}(\text{diff}(\mathbf{TS}), \mathbf{k}) \\ & + \mathbf{c} \times \text{movstd}(\text{diff}(\mathbf{TS}), \mathbf{k}) \\ & + \mathbf{b} \end{aligned} \tag{3.2}$$

where **TS** is the time series, **u** is either 0 or 1 to determine whether `movmean` is used, **k** is the window size to compute **k**-points mean values and standard deviations, **c** is

the coefficient applied to `movstd`, and **b** is the offset to adjust the center of the right-hand side of (3.1) or (3.2).

The only difference between (3.1) and (3.2) is to use either `diff(TS)` or *`abs`*`(diff(TS))`. From (3.1) and (3.2), we can derive the following simplified one-liners:

$$abs(\text{diff}(\mathbf{TS})) > \mathbf{b} \tag{3.3}$$

$$\begin{aligned} abs(\text{diff}(\mathbf{TS})) > {} & \text{movmean}(abs(\text{diff}(\mathbf{TS}), \mathbf{k}) \\ & + \mathbf{c} \times \text{movstd}(abs(\text{diff}(\mathbf{TS})), \mathbf{k}) \\ & + \mathbf{b} \end{aligned} \tag{3.4}$$

$$\text{diff}(\mathbf{TS}) > \mathbf{b} \tag{3.5}$$

$$\begin{aligned} \text{diff}(\mathbf{TS}) > {} & \text{movmean}(\text{diff}(\mathbf{TS}, \mathbf{k}) \\ & + \mathbf{c} \times \text{movstd}(\text{diff}(\mathbf{TS}), \mathbf{k}) \\ & + \mathbf{b} \end{aligned} \tag{3.6}$$

We did a simple bruteforce search to compute individual **k**, **c** and **b** which solve anomaly detection problems on all 367 time series. As the results show in **Table 3.1**, we are surprised by the triviality of the Yahoo Benchmark: 316 out of 367 (86.1%) can be easily solved with a one-liner.

Surprisingly, 193 out of 367, that is more than half, time series in Yahoo Benchmark can be solved with individual *magic numbers* **b** in (3.3) or (3.5). Even for those fourteen time series solvable with (3.6) in A3 dataset, they share a common property of $\mathbf{k} = 5$ and $\mathbf{c} = 0$, while **b** varies case by case.

The overall 86.1% number seems competitive with most papers that have examined this dataset [73, 82, 83] (it is difficult to be more precise than that because of the vagaries of scoring functions). Moreover, as we will show in **Section 3.1.4**, because of some labeling errors, this is probably as close to perfect as can be achieved on this dataset.

Table 3.1: Bruteforce results on Yahoo Benchmark

Dataset	Solvable with	# Time Series Solved	# Time Series in Dataset	Percent
A1	(3.3)	30	67	44.8%
	(3.4)	14		20.9%
	Subtotal	44	67	65.7%
A2	(3.3)	40	100	40.0%
	(3.4)	57		57.0%
	Subtotal	97	100	97.0%
A3	(3.5)	84	100	84.0%
	(3.6)	14		14.0%
	Subtotal	98	100	98.0%
A4	(3.5)	39	100	39.0%
	(3.6)	38		38.0%
	Subtotal	77	100	77.0%
	Total	316	367	86.1%

In [39], we show a gallery of dozens of additional examples from Yahoo [3], Numenta [4], NASA [5] and Pei's Lab (SMD) [6] that yield to one line solutions.

3.1.3 Unrealistic Anomaly Density

This issue comes in three flavors:

- For some examples, more than half the test data exemplars consist of a contiguous region marked as anomalies. For example, NASA datasets D-2, M-1 and M-2. Another dozen or so have at least 1/3 of their length consist of a contiguous region marked as anomalies [5].

- For some examples, there are *many* regions marked as anomalies. For example, SMD exemplar machine-2-5 has 21 separate anomalies marked in a short region.

- In some datasets, the annotated anomalies are very close to each other. For example, consider **Figure 3.3**, it shows two anomalies sandwiching a *single* normal datapoint.

There are many issues with such an unrealistic anomaly density. First, it seems to blur the line between *classification* and *anomaly detection*. In most real-world settings, the prior probability of an anomaly is expected to be only slightly greater than zero. Having half the data consist of anomalies seems to violate the most fundamental assumption of the task. Moreover, many algorithms are very sensitive to the priors.

Another issue is that this unrealistic density greatly confuses the task of scoring and comparing algorithms. Suppose we have a dataset with ten anomalies, one at about midnight for ten days, reflecting an increasingly weakening pump filling a tank at the start of a batch process. We could imagine two rival algorithms, each of which managed to detect a single anomaly. However, one algorithm finds the first anomaly, and the other algorithm finds the last. These outcomes correspond to very different practical results when deployed. The former saves ten bad batches being created, the latter only one. We might imagine rewarding more for earlier detection, and in fact the Numenta team [4] (among others) have suggested that. However, the resulting scoring function is exceedingly difficult to interpret, and almost no one uses this [86].

We believe that the ideal number of anomalies in a single testing time series is exactly *one*. Moreover, this number should be communicated with the dataset. This makes the users task a little easier. Instead of trying to predict *if* there is an anomaly in the dataset,

the algorithm should just return the most likely *location* of the anomaly. However, for this slight simplification which ignore specificity (which can and should be evaluated separately), we gain the fact that the evaluation is now *binary*. By testing on multiple datasets, we can report the aggregate results as simple *accuracy*, which is intuitively interpretable.

3.1.4 Mislabeled Ground Truth

All of the benchmark datasets appear to have mislabeled data, both false positives and false negatives. Of course, it seems presumptuous of us to make that claim, as the original creators of the datasets may have had access to out-of-band data they used to produce the labels. Nevertheless, we believe that many of the examples we claim are compelling enough to be unambiguous.

For example, consider the snippet of Yahoo A1-Real32 shown in **Figure 3.4**. Any algorithm that points to location **B** will be penalized as having a false positive, but a true positive region **A**, is part of the same constant line. Since literally nothing has changed from **A** to **B**, it is hard to see how this labeling makes sense[5].

In **Figure 3.5**, we see another Yahoo time series A1-Real46. There is a point anomaly (or "dropout") marked with **C**. However, at location 360, there is an almost identical dropout **D** that is not labeled as having an anomaly.

[5] If the rest of the data had many short constant regions, say of length 12, then you could imagine that a good algorithm might consider the 13^{th} constant datapoint in a row an anomaly. However, this is the only constant region in this dataset.

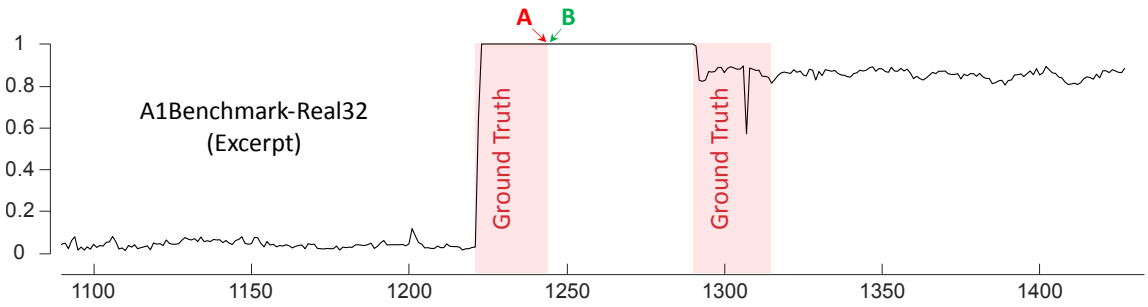

Figure 3.4: An excerpt from Yahoo A1-Real32. An algorithm that points to **A** will be marked as a true positive. An algorithm that points to **B** will be marked as a false positive.

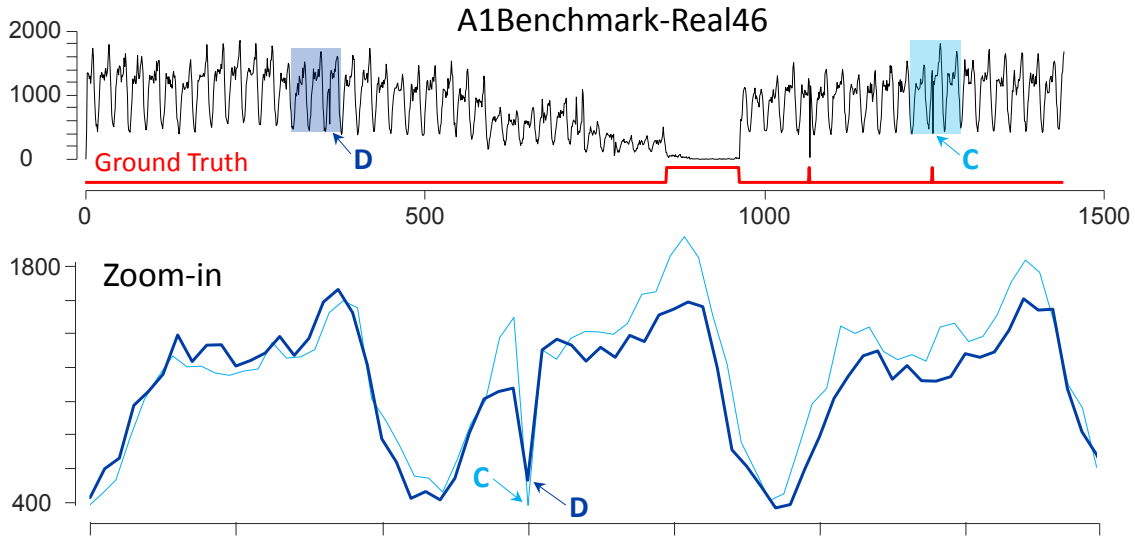

Figure 3.5: (*top*) The Yahoo A1-Real46 dataset with its class labels (red). (*bottom*) Overlaying two snippets allows a one-to-one comparison between the region of **C** and **D**. The single point marked **C** is a true positive, but surprisingly, the point marked **D** is not.

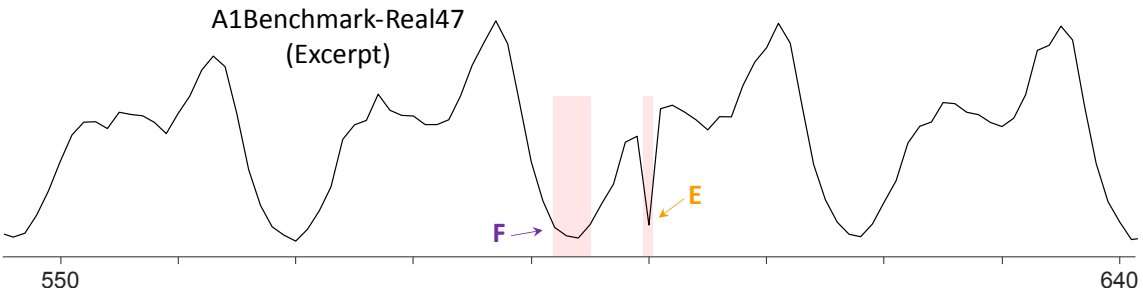

Figure 3.6: An excerpt from Yahoo A1-Real47. Both **E** and **F** are marked as anomalies, but it is hard to see that **F** is truly an anomaly.

In **Figure 3.6**, we see a snippet of Yahoo A1-Real47 with two labeled anomalies. The one pointed to by **E** seems like a dropout, but **F** is a puzzle. Its rounded bottom visually looks like a dozen other regions in this example.

If we measure **F**'s mean, min, max, variance, autocorrelation, complexity, Euclidean distance to the nearest neighbor, etc. and compare these numbers to other rounded bottom regions (**Figure 3.6** shows two others, of the about 48), there is simply nothing remarkable about it.

Beyond these issues, there are other labeling issues in the Yahoo datasets. For example, two datasets seem to be essentially duplicates (A1-Real13 and A1-Real15). An additional issue is more subjective, but some of the datasets seem to have unreasonably precise labels. Consider the labels for A1-Real67 shown in **Figure 3.7** (*top*).

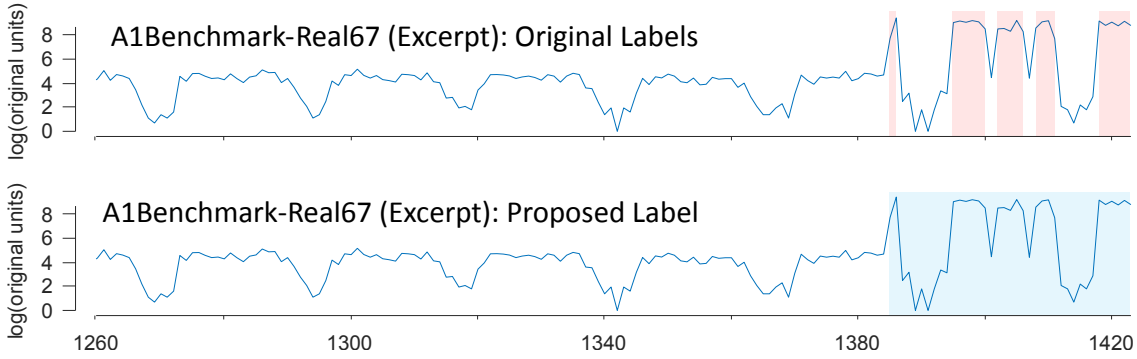

Figure 3.7: (*top*) An excerpt from the Yahoo A1-Real67 dataset with its class labels (red). (*bottom*) Our proposed label (blue) for this dataset.

By analogy, some modern automobiles have anomaly detection sensors to detect violent crashes. Imagine a fast-moving car is involved in a crash and goes thumbing end-over-end down the highway. At some points in the rotation, the car will momentarily have a normal orientation. However, it would be bizarre to label those regions as "`normal`".

Similarly, in A1-Real67, after about 50 almost identically repeated cycles, at time 1,384, the system has clearly dramatically changed, warranting flagging an anomaly. However, the subsequent rapid toggling between "`anomaly`" and "`normal`" seems unreasonably precise.

There are several reasons why this matters. Most anomaly detectors effectively work by computing statistics for each subsequence of some length. However, they may place their computed label at the beginning, the end or the middle of the subsequence. If care is not taken, an algorithm may be penalized because it reports a positive *just* to the left (or *just* to the right) of a labeled region. This is always a possible concern, but it becomes much more of an issue with rapid toggling of states.

One of the most referenced datasets is Numenta's NYC Taxi dataset, which records the taxi demand in New York City from 2014/07/01 to 2015/01/31 [4]. According to the original labels, there are five anomalies, corresponding to the NYC marathon, Thanksgiving, Christmas, New Year's Day, and a blizzard.

However, as shown in **Figure 3.8**, this ground truth labeling seems to have issues.

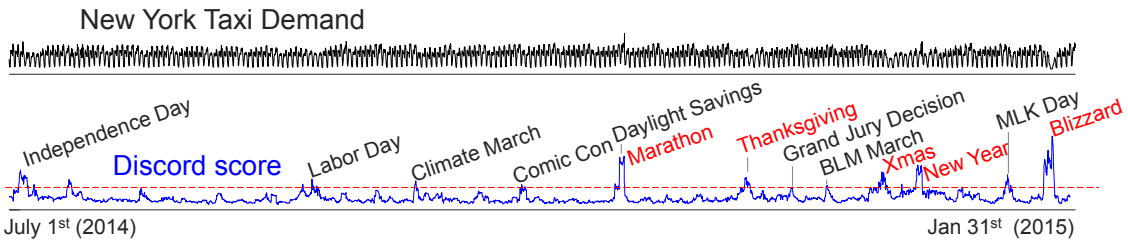

Figure 3.8: (*top*) Numenta's NYC Taxi dataset. (*bottom*) The *time series discord* score of the dataset [85, 84], with peaks annotated. The red text denotes the ground truth labels.

One minor issue is the anomaly attributed to the NYC marathon is really caused by a daylight-saving time adjustment that was made the same day.

However, the main problem with the five labels is that they seem very subjective. After a careful visual analysis, we believe that there are at least seven more events that are equally worthy of being labeled anomalies, including Independence Day, Labor Day and MLK Day. In addition to these USA holidays, we can easily detect the impromptu protests that followed the grand jury decision not to indict officers involved in the death of Eric Garner, "*Large groups shouted and carried signs through Times Square... Protesters temporarily blocked traffic in the Lincoln Tunnel and on the Brooklyn Bridge*" [87], and the more formal protest march that followed ten days later.

It is difficult to overstate the implications of this finding. At least dozens of papers have compared multiple algorithms on this dataset [4, 77, 78, 88], especially a recent paper [88] claiming that "*The performance of* (our algorithm) *is compared with those of related methods, such as STL, SARIMA, LSTM, LSTM with STL, and ADSaS. The comparison results show that* (our algorithm) *outperforms the others in terms of the precision, recall, and F1-score.*" However, it is possible that an algorithm that was reported as performing very poorly, finding zero true positives and multiple false positives, actually performed *very* well, discovering Grand Jury, BLM March, Comic Con, Labor Day and Climate March, etc. The perfect straight 1.0 on precision, recall and F1 scores claimed in [88], just happens to agree with significant mislabeling in NYC Taxi dataset, and strongly suggests *overfitting*.

Finally, let us consider an example from the NASA archive [5]. In **Figure 3.9**, we show three snippets from a test set of Mars Science Laboratory: G-1. One of the snippets is

labeled with the only anomaly acknowledged in this dataset. The anomaly corresponds to a dynamic behavior, becoming "frozen" for a period of time. However, the two other snippets also have this strange neighbor, but are *not* marked as anomalies. As always, it is possible that the creators of this archive have access to some out-of-band information that justifies this (none of the metadata or reports that accompany the data discuss this). However, in this case, it is particularly hard to believe these labels. In any case, suppose we compare two algorithms on this dataset. Imagine that one finds just the first true anomaly, and the other finds all three events highlighted in **Figure 3.9**. Should we really report the former algorithm as being vastly superior?

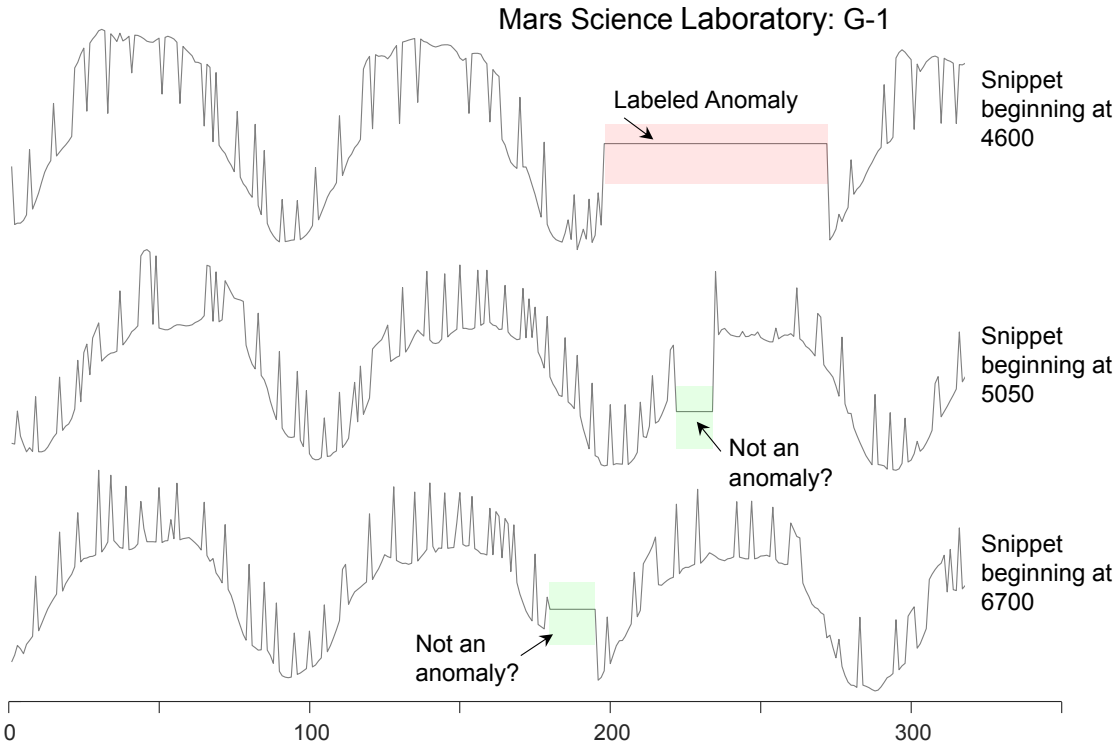

Figure 3.9: (*top to bottom*) Three snippets from Mars Science Laboratory: G-1. The topmost one has the only labeled anomaly in this dataset. However, the bottom two snippets have essentially identical behaviors as the anomaly, but are not identified as such.

3.1.5 Run-to-failure Bias

There is an additional issue with at least the Yahoo (and NASA) datasets. As shown in **Figure 3.10**, many of the anomalies appear towards the end of the test datasets.

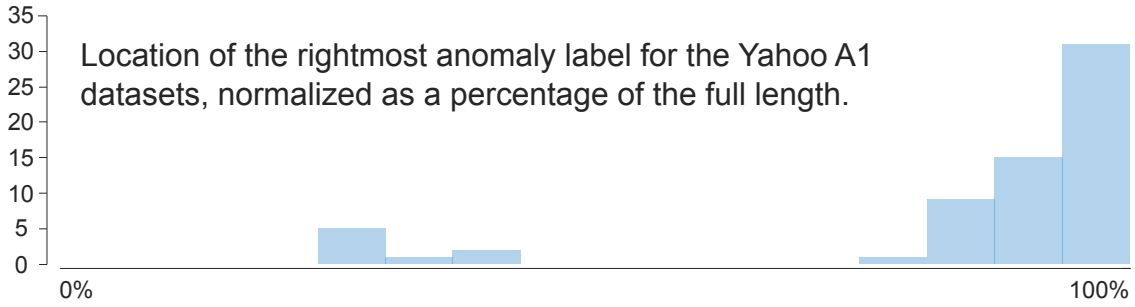

Figure 3.10: The locations of the Yahoo A1 anomalies (rightmost, if there are more than one) are clearly not randomly distributed.

It is easy to see why this could be true. Many real-world systems are run-to-failure, so in many cases, there *is* no data to the right of the last anomaly. However, it is also easy to see why this could be a problem, as it drastically affects the default rate. A naïve algorithm that simply labels the last point as an anomaly has an excellent chance of being correct.

3.1.6 Summary of Benchmark Flaws

We believe that we have demonstrated that the classic time series anomaly detection archives are irretrievably flawed. For example, if we were told that algorithm A could achieve an F1 score of 1.0 on one of these datasets [88], should we be impressed? Given what we know about the amount of mislabeling on these datasets, we should not be impressed, instead we should have to suspect fraud or (much more plausibly) error.

However, suppose instead that we were told that algorithm B could achieve an F1 score of 0.9 on one of these datasets. Given what we know about the triviality of these datasets, this seems like something we could match or beat with decades-old algorithms. Thus, there is simply no level of performance that would suggest the utility of a proposed algorithm.

Similarly, if we were told that algorithm C was compared to algorithm D on these datasets, and algorithm C emerged as being an average of 20% better, could we now assume that algorithm C really is a better algorithm in general? Again, given what we know about these datasets, even a claimed 20% improvement (larger than the typically claimed margin of improvement) would not imbue confidence. Recall just **Figure 3.8**, on that dataset, if algorithm C scored a perfect score, relative to the *claimed* labels, we should regard it as a poor algorithm with *low* sensitivity.

3.2 Introducing the UCR Anomaly Archive

Having observed the faults of many existing anomaly detection benchmarks, we have used the lessons learned to create a new benchmark dataset, The UCR Time Series Anomaly Archive [20]. As we explain below, we have endeavored to make our resource free of the issues we have noted, with one exception. A small fraction of our datasets may be solvable with a one-liner. There are two reasons for this. First, we wanted to have a spectrum of problems ranging from easy to very hard. Second, there *are* occasionally real-word anomalies that manifest themselves in a way that is amenable to a one-liner, and their inclusion will allow researchers to make claims about the generality of their ideas. For

example, AspenTech, an oil and gas digital historian, encodes missing data as -9999. If the data is ported to another system and normalized, the exact value of -9999 may change, but such a rapid decrease in value should rightly trigger an anomaly. Such dropouts are generally easy to discover with a one-liner.

To prevent the datasets in the archive reflecting our biases and interests too much, we broadcasted a call for datasets on social media platforms read by data scientists, and we wrote to hundreds of research groups that had published a paper with "anomaly detection" in the title in the last five years. Alas, this did not yield a single contribution. Nevertheless, the datasets span many domains, including medicine, sports, entomology, industry, space science, robotics, etc.

As we discussed in **Section 3.1.3**, we believe that the ideal number of anomalies in a test dataset is one. The reader will be curious as to how we ensured this for our datasets. Clearly, we do not have space to explain this for *each* dataset (although the archive does have detailed provenance and metadata for each dataset [20]). Below we show two representative examples to explain how we created single anomaly datasets.

3.2.1 Natural Anomalies Confirmed Out-of-Band

Consider **Figure 3.11** which shows an example of one of the datasets in our archive. The first 2,500 datapoints (the '2500' in the file's name) are designed to be used as training data, and the anomaly itself is located between datapoints 5,400 and 5,600 (the '5400_5600' in the file's name) indicate the location of the anomaly.

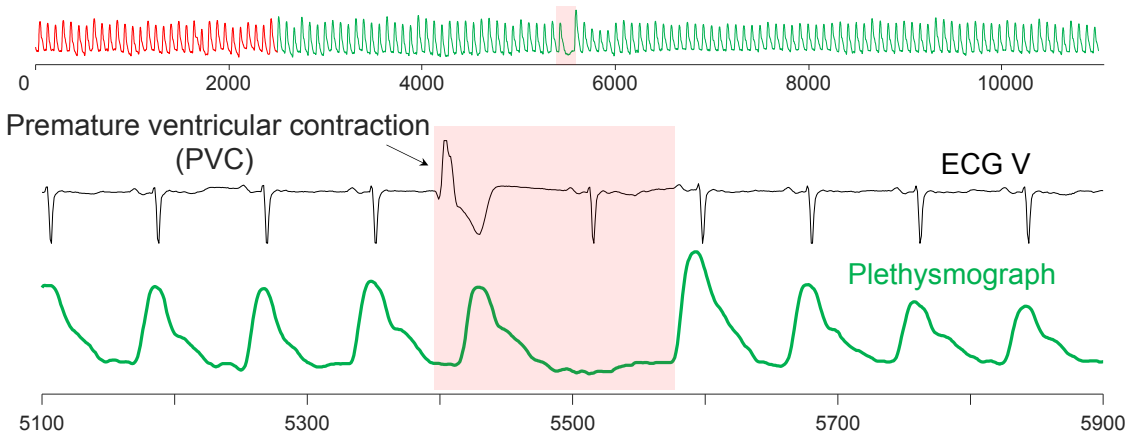

Figure 3.11: (*top*) `UCR_Anomaly_BIDMC1_2500_5400_5600`, a dataset from our archive. (*bottom*) A zoom-in of the region containing the anomaly. A PVC observed in an ECG that was recorded in parallel offers out-of-band evidence that this is a true anomaly.

Here the anomaly is a little subtle. How can we be so confident that it is semantically an anomaly? We can make this assertion because we examined the electrocardiogram that was recorded in parallel. This was the only region that had an abnormal heartbeat, a PVC. Note that there is a slight lag in the timing, as an ECG is an *electrical* signal, and the pleth signal is *mechanical* (pressure). However, the scoring functions typically have a little "play" to avoid the brittleness of requiring spurious precision.

Note that we did not directly create an ECG benchmark here because it is too simple (although we do have a handful of equally simple examples in the archive). We used this general technique, of using obvious out-of-band data to annotate subtle data, to create many of our benchmark datasets.

3.2.2 Synthetic, but Highly Plausible Anomalies

We can also *create* single anomaly datasets in the following way. We find a dataset that is free of anomalies, then insert an anomaly into a random location. However, we want

to do this in a way such that the resulting dataset is completely plausible and natural. **Figure 3.12** shows an example of how we can achieve this.

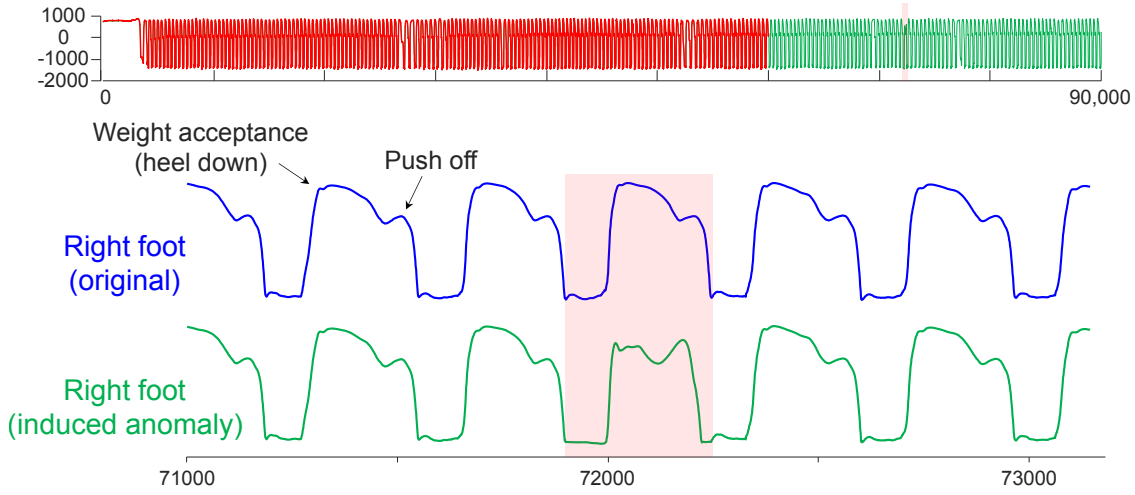

Figure 3.12: (*top*) `UCR_Anomaly_park3m_60000_72150_72495`, a dataset from our archive. (*bottom*) This individual had a highly asymmetric gait, so we created an anomaly by swapping in a single *left* foot cycle in a time series that otherwise records the *right* foot.

Here we started with a two-dimensional time series, containing the left and right foot telemetry on a force plate. The data came from an individual with an antalgic gait, with a near normal right foot cycle (RFC), but a tentative and weak left foot cycle (LFC). Here we replaced a single, randomly chosen RFC with the corresponding LFC (shifting it by a half cycle length). The resulting dataset looks comply natural, modeling a normal gait, where for one cycle the individual felt a sudden spasm in the leg.

This dataset has another source of viability that happens three or four times. Because the force plate apparatus is of finite length, the gait speed changes as the user circles around at the end of the device. However, we took pains to ensure that both the train and test data have examples of this behavior, so it should not be flagged as an anomaly.

When creating such datasets, we attempted to thread the needle between being too easy, and too difficult. Here we are confident that this example is not impossibly cryptic, as nine out of ten volunteers we asked could identify this anomaly after careful *visual* inspection.

3.3 Recommendations

We conclude with some recommendations for the community.

3.3.1 Existing Datasets should be Abandoned

The community should abandon the Yahoo [3], Numenta [4], NASA [5] and SMD [6] benchmark datasets. As we have demonstrated, they are irretrievably flawed, and almost certainly impossible to fix, now that we are several years past their creation. Moreover, existing papers that evaluate or compare algorithms primarily or exclusively on these datasets should be discounted (or, ideally *reevaluated* on new challenging datasets).

3.3.2 Algorithms should be Explained with Reference to their Invariances

We would argue that the task of time series *classification* has seen more progress in recent years. In that community, it is understood that it is often useful to discuss novel algorithms in terms of the *invariances* they support [89]. These invariances can include amplitude scaling, offset, occlusion, noise, linear trend, warping, uniform scaling, etc. [89]. This can be a very useful lens for a practitioner to view both domains and algorithms. For example, suppose we wish to classify mosquitoes sex using a Single-Sided Amplitude

Spectrum of their flight sounds (as was done in [90]). With a little introspection about entomology and signal processing, we can see that we want any algorithm in this domain to be invariant to the amplitude of the signal. We also want some *limited* warping invariance to compensate for the fact that insect's wingbeat frequency has a dependence of temperature, but not too much warping, which might warp a sluggish female (about 400 Hz) with a much faster male (about 500 Hz). This immediately suggests using a nearest neighbor classifier, with area-under-the-curve constrained DTW (cDTW) as the distance measure. Here, seeing the problem as choosing the right invariances is a very helpful way to both communicate the problem and search the literature for the right solution.

In contrast, one thing that is striking about many recent papers in anomaly detection is that the authors do not clearly communicate under what circumstances the proposed algorithms should work for practitioners that might want to use them. (The work of [91] is a notable exception.) For example, would the ideas in [6] work if my data was similar, but had a wandering baseline that was not relevant to the normal/anomaly distinction?

We suggest that authors could communicate the important invariances with figures.

Consider **Figure 3.13** (*top*) which shows a one-minute long electrocardiogram that contains a single anomaly (a premature ventricular contraction). The figure also shows the anomaly score from two methods, Telemanom [5] and Discord [84, 85].

Here we are only interested in the relative values, so we omitted the Y-axis, in both cases, the higher values are considered more anomalous. In this example the anomaly is very obvious, and gratifyingly, both methods peak at the location of the anomaly. Visually,

we might claim that Discords offer more *discrimination* (informally, the difference between the highest value and the mean values).

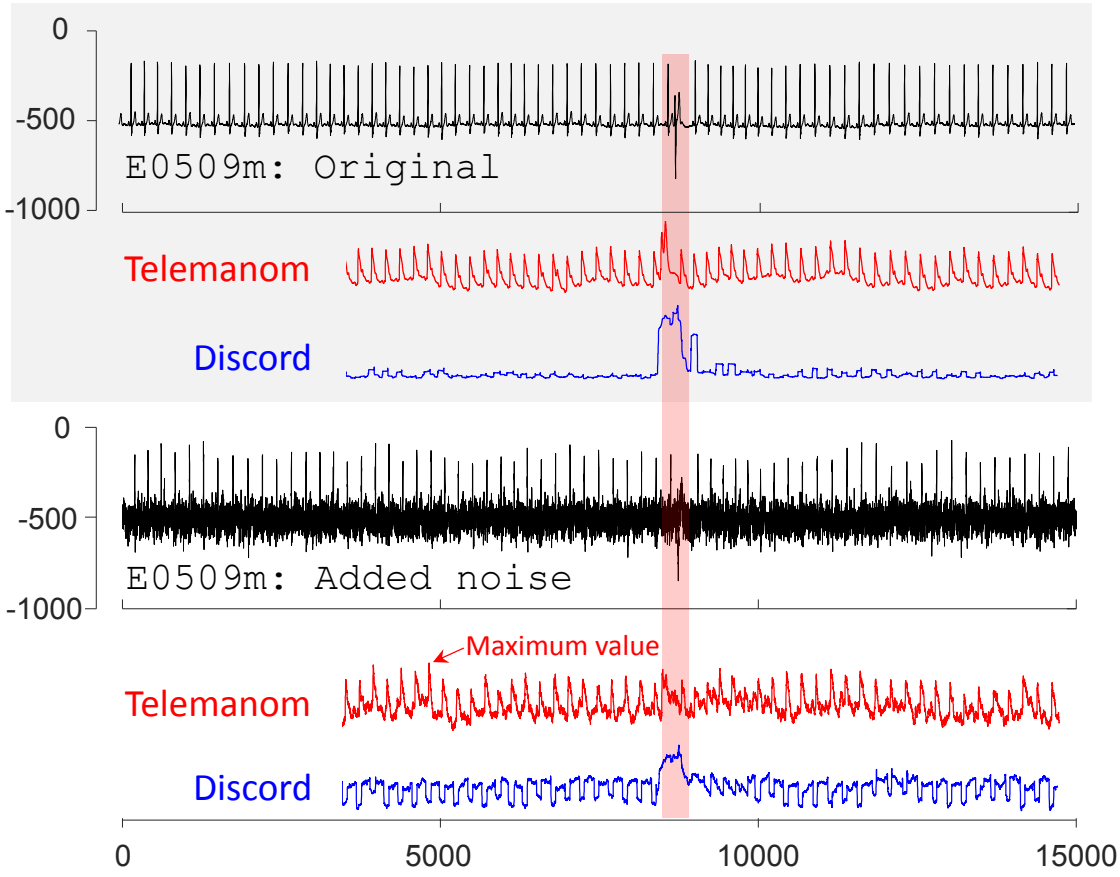

Figure 3.13: (*top*) One minute of an electrocardiogram with an obvious anomaly that is correctly identified by two very different methods. Telemanom uses the first 3,000 datapoints from training, using the original authors suggested settings. Discord uses no training data. (*bottom*) The same electrocardiogram with noise added confuses one of the algorithms more that than the other.

In **Figure 3.13** (*bottom*), we show the same time series, after we added a significant amount of Gaussian noise. The Discord approach now provides less *discrimination*, but still peaks in the right place. In contrast, Telemanom now peaks in the wrong location.

This example suggests that one approach might be better than the other if we

expect to encounter noisy data. We are not suggesting that such visualizations *replace* the reporting of metrics such as precision, recall and F1 score, etc. However, for the datasets we consider in this chapter, those metrics often summarize an algorithm's predictions at just two or three locations. In contrast, the plots shown in **Figure 3.13** visually summarize the algorithm's predictions at 12,000 locations, and give us a much richer intuition as to the algorithms invariances.

3.3.3 Visualize the Data and Algorithms Output

The point is partly subsumed by the previous point, but worth explicitly stating.

It is very surprising to note that many papers that study time series anomaly detection plot few (as few as *zero*) examples of the time series themselves, in spite of the fact that time series analytics (unlike say protein strings) is inherently a visual domain.

This is more than just a presentation issue; it informs how we should do research. We suspect that some researchers rarely view the time series, they simply pass objects to a black box and look at the F1 scores, etc. One reason we believe this is that the four issues we note in this chapter are readily visually apparent, they do not need any tools to discover, other than a way to plot the data. For example, the issues with Numenta's NYC Taxi dataset discussed in **Section 3.1.4** simply "jump out" of the screen if you plot the data, and the entire data can be comfortably examined on a desktop screen, without even the need for zoom or pan [4]. Yet to our knowledge, no one has noted these problems before.

3.3.4 A Possible Issue with Scoring Functions

In this chapter, we have mostly confined our interest to problems with the current *datasets*. Others have considered problems with current *scoring* functions [86]. However, it would be remiss of us not to note a simple potential issue with scoring functions, especially when comparing rival algorithms. As we noted above, algorithms can place their computed anomaly score at the beginning, the end or the middle of the subsequence. **Figure 3.13** (*top*) nicely illustrates this. Both approaches can find the obvious anomaly, but Telemanom places its peak earlier than Discords[6]. It is easy to see that unless we are careful to build some "slop" into what we accept as a correct answer, we run the risk of a systemic bias against an algorithm that simply formats its output differently to its rival. As before, *visualization* of the algorithms, together with *visualization* of the acceptable answer range (the red bar in **Figure 3.13**) would go a long way to boost a reader's confidence that the evaluation is fair.

3.4 Conclusion

We have shown that the most commonly used benchmarks for anomaly detection have flaws that make them unsuitable for evaluating or comparing anomaly detection algorithms. On a more positive note, we have introduced a new set of benchmark datasets that is largely free of the current benchmark's flaws [20].

However, we do not regard this chapter as the last word on the matter. Ideally, a committee or a workshop at a conference should gather many diverse viewpoints on these

[6] This should not be confused with the claim that Telemanom *discovers* the anomaly earlier, which may or may not be true. This is only a minor claim about *formatting* of a particular implementation's output.

issues, and draft recommendations for the creation of a crowdsourced set of benchmark datasets. We hope this chapter will go some way to prod the community into action.

Chapter 4

Not So Clear Definition

Since its introduction two decades ago, there has been increasing interest in the problem of *early classification of time series* (ETSC). The problem is expressed differently by different researchers, but it generally reduced to asking if we can classify a time series subsequence with sufficient accuracy and confidence after seeing only some prefix of a target pattern. Using text as an analogy for time series, if someone typed `albuquer...`, we could be very confident that they planned to type the name of the most populous city in New Mexico.

The key claim is that classification without waiting for the entire pattern to appear would allow us to take immediate action in a domain in which some interventions are possible. For example, that intervention might be pre-tightening the seatbelts in an automobile that the classifier predicts may be about to crash.

While the idea of ETSC is interesting and socially noble, in this chapter, we make a somewhat surprising claim. In spite of the fact that there are many research efforts on

ETSC, it is not clear that any of them could ever work in a real-world setting. The problem is not with the algorithms per se but with the vague and underspecified problem description. Most of the issues stem from a mismatch between the data format used to train and test ETSC models and the data format that must be used in the real world. Most ETSC papers consider only data in the UCR format, which as shown **Figure 4.1**, assuming that all exemplars are of the same length and at least approximately aligned in time [13].

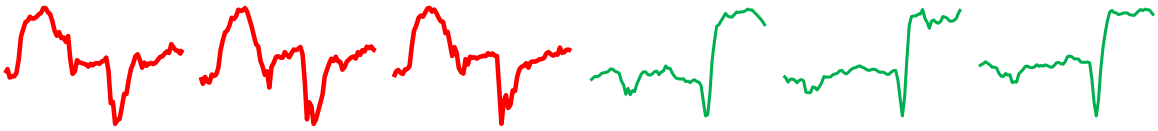

Figure 4.1: Samples of data in the UCR format. Note that exemplars are all of the same length and carefully aligned. The exemplars are utterances of the words cat and dog, spoken by a female in Standard American English, represented in MFCC Coefficient 2.

Given data formatted in this way, the ETSC community has produced dozens of models that can predict the class of an incoming subsequence, after only seeing a fraction of the data [7, 8, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37]. This sounds impressive, but as shown in **Figure 4.2**, consider what would happen when we test on the utterance "*It was said that Cathy's dogmatic catechism dogmatized catholic doggery*'.

This sentence will produce six false positives: three in each class. Note that we cannot brush the problem aside by saying that we can simply recant the classifications *after* we see the rest of the longer word. The whole point of ETSC is to take some actions. The action might be "just" sounding an alarm, but even just false alarm fatigue is known to have a huge cost [92]. If 99.9% of all alarms are false positives, it seems inconceivable that the system would be used.

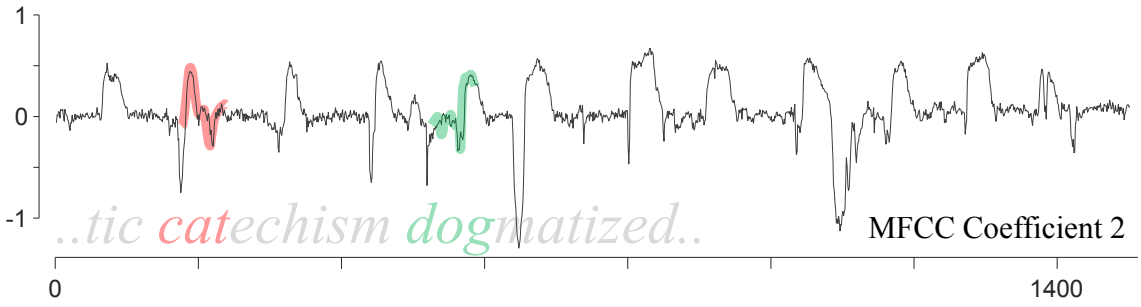

Figure 4.2: A snippet of the phrase "*It was said that Cathy's dogmatic catechism dogmatized catholic doggery*". This short sentence will allow any ETSC method to make confident and early predictions, all of which will later have to be recanted.

It is also important to recall that by the explicit definition of the ETSC problem, the action must be *immediate*. If we wait "to make sure", then in no sense are we doing *early* classification – we are just doing classification.

This issue of false positives would be damning even if we had no false negatives. However, as we will show in **Section 4.3**, most ETSC methods have a misunderstanding about the normalization of the data that will condemn them to produce many false negatives.

We call the "cat" vs "cat*alog*" problem the *prefix* problem. We will later show two other issues, the *inclusion* and *homophone* problems that offer even greater stumbling blocks to any ETSC models.

The absolute weakest interpretation of our findings is that the ETSC community has failed to communicate or appreciate the many assumptions that must be true for their models to be useful in the real world. However, we will argue a stronger interpretation. The ETSC problem is underspecified to the point of being meaningless, and the entire area needs to be "rebooted" with greater rigor.

4.1 Background

4.1.1 How ETSC Algorithms Work

The idea of early classification in time series seems to have originated in an obscure paper in 2001 [93], however the problem framework that is most commonly understood appears in a sequence of papers by Xing *et al.* [7, 8]. These works define the challenge as finding the best compromise between accuracy of prediction and earliness in the face of incrementally arriving data. This can be framed in several ways, and different papers use slightly different terminology. However, **Figure 4.3** shows the two most common interpretations of this idea.

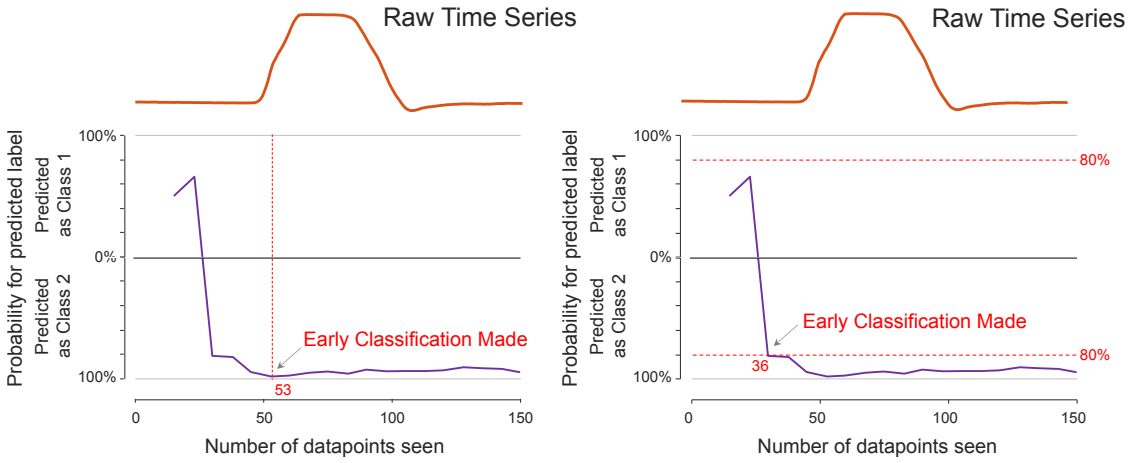

Figure 4.3: (*left*) The TEASER model [21] correctly predicts the class of an exemplar from GunPoint after seeing only 53 data points. (*right*) Other models predict only when a user-specified confidence threshold is met.

In **Figure 4.3** (*left*), we used the method in [5], working on the ETSC community's favorite dataset, GunPoint [13]. As the data arrives, some models predict the probability that we are seeing the prefix of any of the classes we have trained on. At some point, an

internal model decides it has seen enough to trigger a classification. Different papers use different internal models, and a handful incorporates some awareness of misclassification costs [29, 36]. In **Figure 4.3** (*right*), we see another common framing of the problem. Here the ETSC algorithm simply predicts the probability of being in each class, and if that probability exceeds some user-specified threshold. In this case, the user's threshold of 80% allowed classification after seeing only 36 datapoints. In a sense, the two models are equivalent, and the slight distinctions do not concern us here.

4.1.2 Disconnect to the Real World

The motivation for early time series classification is plausible, although to our knowledge, there has never been an ETSC algorithm deployed in the real world. As we will see, this seems to be a telling fact. In contrast, while classic time series classification is perhaps an overstudied problem, it is still easy to point to hundreds of commercial and scientific applications that actually use it.

One issue seems to be that there is a disconnect between the models and the claimed uses for them. Consider [37], which motivates ETSC with "*in the early diagnosis of heart disease, abnormal ECG signals may indicate a specific heart disease that needs immediate treatment. If a classification model that can make early diagnosis as soon as early of ECG time series is available, the patient with the heart disease can get early treatment.*" As shown in **Figure 4.4**, the authors of [37] do indeed test on ECGs.

They later also correctly note "*If a person has a myocardial infarction, it is usually observed from the ECG that the ST wave is changed and elevated...*". However, let us step

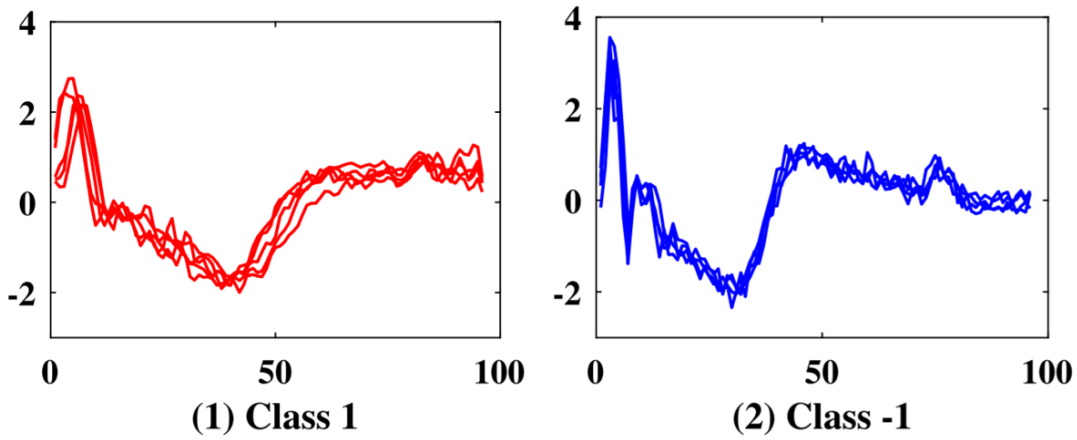

Figure 4.4: A screen dump from [37]. The authors test on an ECG dataset from the UCR archive [13].

back a moment. Yes, it is true that heart disease needs immediate treatment. However, that is typically understood at the scale of "*today*" is better than "*next month*". Maybe the authors meant the case of a patient recovering in an ICU with the plan to page a doctor the moment we see a single myocardial infarction. The full ECG beats in question are about 0.5 seconds long. Suppose, as [37] claims, we could classify the abnormal heartbeats after seeing only 64% of the data. That means that we could alert the doctor 0.18 seconds earlier. This is an inconsequent amount, especially for a warning that comes with a 17% chance of being a false positive [37] (as we will see in **Section 4.3**, the claim is in any case spurious, as it makes a normalization assumption that could not be true).

More generally, there does seem to be a disconnect in the literature between the obvious and true motivation that "earlier is better", and any practical actionable application of ETSC. In any case, this discussion may be largely moot because as we will show in the next two sections, no current ETSC algorithm is likely to work in any real-world settings due to three types of confounding issues that the community has not noticed.

4.2 ETSC is Much Harder Than it Appears

In **Figure 4.2**, we hinted at a problem caused by assuming that data forced into the UCR format represents a real-world problem. As damning as this single issue is, we will now demonstrate that it is only one of the three related issues that cast doubt not only on the solutions proposed for ETSC, but on the very problem definition itself.

4.2.1 The Prefix Issue

The prefix problem is the assumption that the pattern to be early classified is not a prefix of a longer innocuous pattern.

Imagine that we have two classes which are the MFCC representation of the spoken words, *cat* and *dog*. Again, under the UCR formatting assumption, this would be an ideal ETSC problem. However, as illustrated in **Figure 4.2**, we need to consider what will happen when we deploy in a streaming environment. Suppose we encountered the perfectly valid sentence "*... all oxen excel at persistence, strength and doggedness. The use of cattle for draft work in...*" [94]. We would get two early classifications, which must then later be recanted.

The reader might imagine that while we may produce an early classification for "*ca...*", we can later retract that prediction when we subsequently see "*... ttle*". But recall that the whole point of early classification is to give actionable early warning. If it is supposed to be actionable, do we take that action or not? If we need to wait until we are sure that there is no retraction before taking the action, then in what sense are we doing early classification – we are surely just doing classification.

We believe that the prefix problem may be essentially insurmountable in many domains. For example, imagine we wanted to early classify the *vocalization* of {gun, point}. There are eighty-eight English words beginning with gun, including *gunwales*, *gunnel*, *gunnysack*, *gunk*, etc., and twenty-six words that begin with point, including *pointedly*, *pointlessness*, *pointier*, *pointman*, etc.

4.2.2 The Inclusion Issue

The inclusion problem is the assumption that the pattern to be early classified is not comprised of smaller atomic units that are frequently observed on their own.

For example, suppose we learn a model for early classification of the vocalization of {lightweight, paperweight}. We can do very well after seeing the first 10% to 20% of these utterances (which is fortunate, as the final 54% of the signal is identical and offers no additional information). However, suppose the universe contains sentences such as "*In the morning light, I could see that I got a papercut from the paper that the light was wrapped in.*" This sentence would give us two false positives for each class. Moreover, it is clear that the sub-pattern could be vastly more common than the full modeled pattern. For words, this is simply an obvious implication of Zipf's law.

Returning to our vocalization of {gun, point} example, recall that in English, we will encounter words like *disappointing*, *ballpoints*, *appointment*, *burgundy*, *begun*, etc., and also proper names like *Gunderson*, the *Pointer* sisters, etc.

4.2.3 The Homphone Issue

The homophone problem is the assumption that two semantically different events will have different shapes in the time series representation.

Suppose that we learn a model for early classification of the vocalization of {flower, wither}. Moreover, we are fortunate that in this problem space, we are told that *any* word containing the target word is also a true positive, so we should take the same action for *flower*, *flowerpot*, *deflowered*, and for *wither*, *witheringly*, *swithering*, etc. This means we are completely free of the prefix and inclusion problems above. However, what are we to make of the following sentence from Leviticus 2:1 "*Whither anyone presents a grain offering as an offering to the Lord, his offering shall be of fine flour, and...*"? This sentence does not contain either of the target words, but it contains two near-perfect homophones, *flower* vs. *flour* and *wither* vs. *whither*, which would give us false positives.

Just because we know that the semantic meaning of the classes in which we are interested is different, it does not follow that the time series representation we see will also be different. For example, as shown in **Figure 4.6** and **Figure 4.9**, *gun* and *point* are extracted from video by tracking the center of mass of the right hand. They are sufficiently different to be distinguished with high accuracy. However, it is possible that completely different behaviors such as *removing-spectacles*, *looking-at-watch*, or *lighting-a-cigarette* are perfect "homophones" in the time series space. In fact, given the vast space of human actions, the very limited one-dimensional view of 150 datapoints virtually assures us this will be the case.

In order to show that *time series* homophones exist, we conducted the following experiment. We randomly selected two examples from the GunPoint dataset, and for each of them, we searched for its three nearest neighbors. However, rather than searching within a human behavior dataset, we searched within three datasets that do *not* have gestures. **Figure 4.5** shows the results.

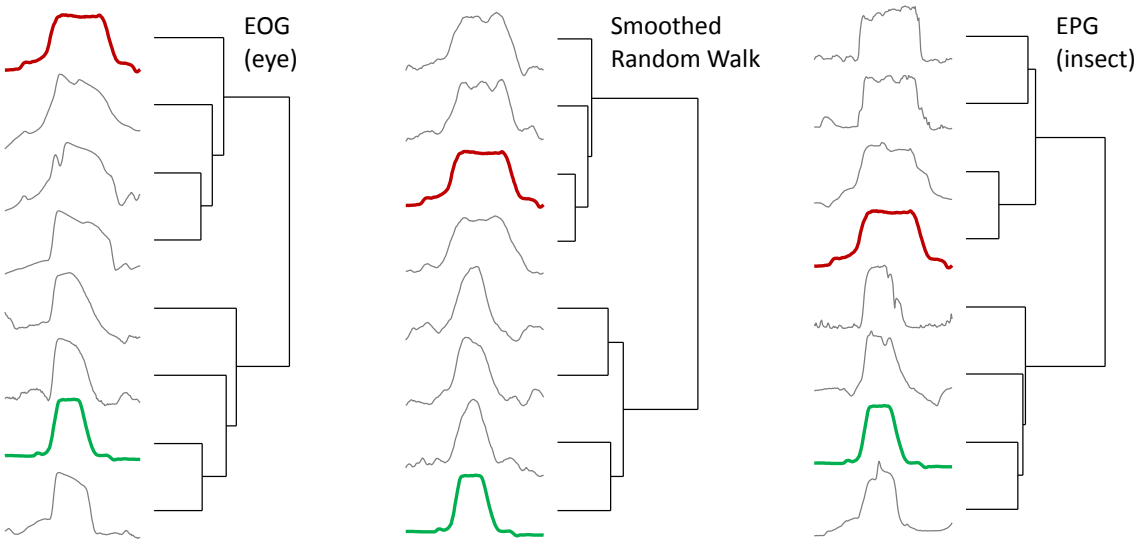

Figure 4.5: Two random examples from the GunPoint dataset (colored), clustered with their nearest neighbors from: (*left*) One hour of eye movement data; (*center*) A smoothed random walk of length 2^{24}; (*right*) Eight hours of insect behavior.

Note that in every case, there is non-gesture data that is much closer to one member of the target class, than the *other* example from the target class. We can repeat this experiment with all datasets from the UCR archive with similar results.

The homophone problem can also show up as part of the inclusion problem. For example, when searching Google for *shapelets*, the time series primitive, most of the hits are true positives, but Google also returns pages with "*Unique puzzle piece shape lets it interlock with...*", "*A simple shape lets the beauty of the faux concrete...*", and "*Its triangular shape*

lets you reach the corners of the pool...". So even though the word *shapelets* does not have a homophone, it does have pseudo-homophones[7]. If we simply searched a large text corpus, we would surely find a lot more of these pseudo-homophones than hits to the obscure data mining primitive.

Returning to our vocalization of {gun, point} example, recall that in English, we will encounter words like *pointe*, *pint*, *Gunn* (proper name), etc.

4.2.4 Summary for this Section

We believe that the prefix, inclusion, and homophone problems imply the space of possible domains where ETSC could be meaningfully applied is vanishingly small. Again, returning to the problem of the vocalization of {gun, point} for a final time. A single English sentence such as "*Amy Gunn thought it pointless to go on pointe before she had begun her appointment to get her burgundy ballet shoes cleaned off all the gunk...*" would produce a plethora of false positives. While most of our examples are contrived for ease of exposition, **Figure 4.5** suggests these problems are common in real-valued time series, as does a more general exploration of the datasets in the UCR archive [13].

It is important to note that while our examples used natural language for simplicity, we have observed these issues in datasets containing gestures, writing, electrical power demand, chicken behavior, insect behavior, bird vocalizations, and in almost everywhere we looked.

There is a data domain *that* might be free of these issues: electrocardiograms (ECGs), photoplethysmograms, and similar time series. However, in the next section, we

[7] Yes, multiple pseudo-homophones: *Our plush ape lets you dress him.*

will show that all ETSC papers that report apparently good results on these datasets are inadvertently "cheating" by peeking into the future.

4.3 Peeking into the Future

Almost all papers on ETSC suffer from a logical flaw that means that their accuracy would plunge if we attempted to use them on streaming data[8]. Once again, the UCR format is the culprit. The UCR datasets are z-normalized. However, when you see the prefix of an oncoming pattern in a streaming environment, you cannot z-normalize it until *after* you have seen all the data, which of course, means that you are not doing *early* classification.

Many researchers seem unaware of just how brittle distance measures are to changes in the mean (and standard deviation) of the exemplars. To show this, let us revisit GunPoint. As shown in **Figure 4.6**, we produced a "denormalized" version of the test data by adding to each instance a random number in the range [-1, 1].

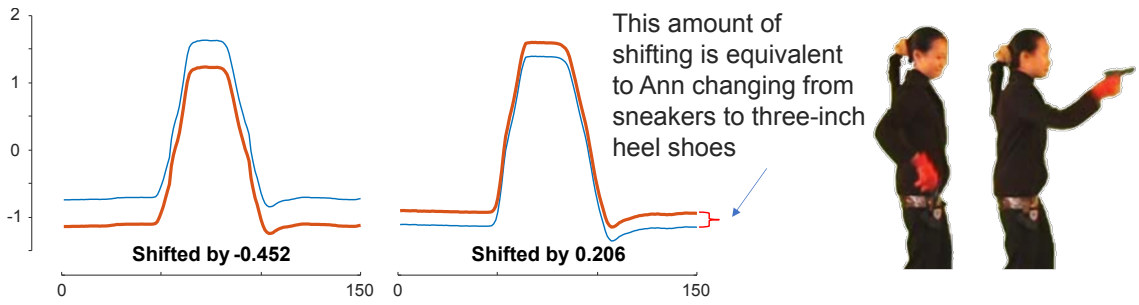

Figure 4.6: Original examples from the GunPoint dataset together with denormalized versions, which have been slightly shifted in the Y-axis.

[8] Paper [21] does *not* have this flaw. We warned them of this issue before [21] was published.

It is important to understand how small of a change this is. It is approximately equivalent to tilting the camera randomly up or down by about 1.9 degrees. Or it is equivalent to replacing Ann with Jessica, a slighter taller grad student.

It is also important to note what effect this would have on normal nearest neighbor classification: *none*. It has long been known that you should z-normalize the data before computing the Euclidean distance or DTW [95]. In **Table 4.1**, we compute the accuracy of six ETSC algorithms on the UCR-normalized data and the denormalized data. We used the authors' recommended settings and/or tested many settings and reported only the *best* results.

Table 4.1: The accuracy of six early classification algorithms

Algorithm	Normalized	DeNormalized
(*min. support* = 0) ECTS [7]	86.7%	68.7%
(*min. support* = 0) RelaxedECTS [7]	86.7%	68.7%
EDSC-CHE [8]	94.7%	62.7%
EDSC-KDE [8]	95.3%	58.7%
($\tau = 0.1$) Rel. Class. [25]	90.0%	70.0%
($\tau = 0.1$) LDG Rel. Class. [25]	91.3%	71.3%

These results show that the algorithms can do apparently very well on GunPoint. However, when we apply the model to streaming data, if the camera zooms in or out, or tilts up or down, or one of the actors decides to go barefoot, or the actor stands a little closer to the camera, etc., the accuracy will plunge.

It is critical not to misunderstand this result. It is not that these algorithms forgot a step, and we can just add it back in. When the algorithms see a value, they are assuming that it is z-normalized based on other values that do not yet exist! As we noted above,

ECGs are a favorite example for ETSC papers [7, 8, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37]. In **Figure 4.7**, we show a tiny snippet of an ECG (recorded from two different chest locations) before it was contrived into the UCR data format.

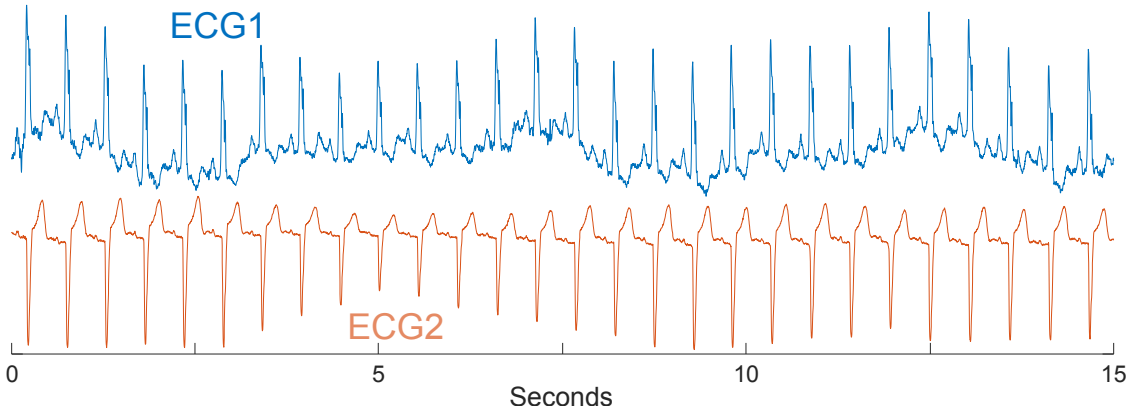

Figure 4.7: An ECG recorded from two locations in the chest. ECG1 shows dramatic but medically meaningless variation in the mean of individual beats. ECG2 shows equally dramatic but also medically meaningless variation in the standard deviation of individual beats.

The practical upshot of this problem is that these algorithms working on medical telemetry will be plagued with false negatives. One might try to get past this issue by saying, "*well, the models will work for domains that don't need z-normalization*". However, Rakthanmanon *et al.* [95] make a forceful case that such domains are very rare or nonexistent.

4.4 Does Early Classification *Ever* Make Sense?

In our long search for a dataset that might work under ETSC assumptions, our best match was a dataset that consists of more than 12.5 billion datapoints of chicken behavior,

measured using a "backpack" accelerometer, as shown in **Figure 4.8** (*right*). Consider the time series shown in **Figure 4.8** (*left*). It is an excellent template to detect the behavior of `dustbathing` in chickens. Any subsequence that is within 2.3 of z-normalized Euclidean distance of this template is essentially guaranteed to be dustbathing.

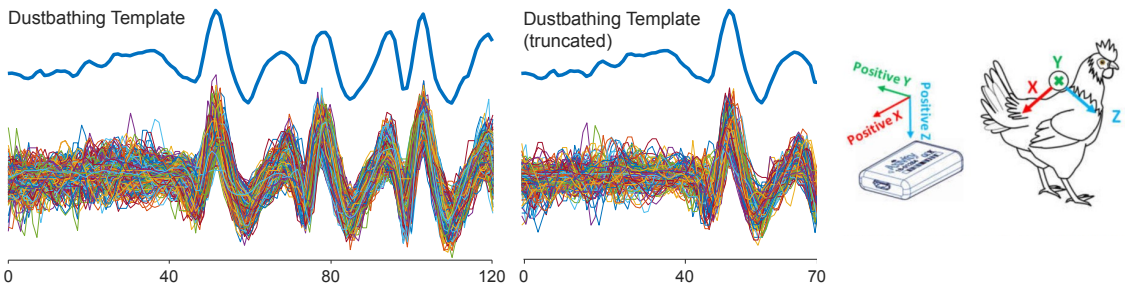

Figure 4.8: (*left*) A template for dustbathing and its 500 nearest neighbors. (*center*) A truncated version of the template and its 500 nearest neighbors. (*right*) The data was obtained from a backpack sensor.

The time series shown in **Figure 4.8** (*center*) is a prefix of the first template, and any subsequence that is within 1.7 of this template can be classified as dustbathing with an accuracy that is not statistically significantly different from the accuracy achieved with the longer template.

One can even make a case for actionability here. Suppose you want to prevent the chicken from conducting long periods of dustbathing. Perhaps if you early classify a `dustbathing` behavior, you could flash a bright light, or play the sound of a chicken's alarm cackle, either one of which would startle the chicken out of its intended behavior. Note that the cost of a false positive is not too high here (although it is not zero, chickens do become desensitized to frequent alarms).

Have we found an example that justifies ETSC? Perhaps, but consider:

- A reader might reasonably say that this is not *early* classification, but rather simply classification with an awareness of the obvious fact that the sensitivity and specificity of time series template will (typically non-linearly) change as you add or delete points to either end.

- We did not need any special algorithms or models to understand that the shorter template is as effective as the longer template. This took common sense and a few minutes of low-code exploration of the data.

- No data from this domain was ever placed into the UCR format. At a minimum, discovering template(s) would need to be done *before* we could even attempt to put the data into the UCR format.

Clearly, absence of evidence is not evidence of absence. But it is surprising that it is so difficult to find a dataset where ETSC would make sense. More telling, to the best of our knowledge, no one in the community has produced a publicly available dataset where it can be claimed: ETSC would be useful, and some ETSC models have been shown to work.

Finally, at the risk of appearing cynical, it is easy to see that one could use this dataset to write a paper that apparently shows utility for ETSC. We could massage more examples like the longer template in **Figure 4.8** (*left*) into the UCR format and show our "model" learns to predict dustbathing after seeing only 70% of the data! Such a claim would look very impressive, but it is only with the context above that we realize that the claim would be vacuous. Could similar situations explain other apparent ETSC successes?

With this in mind, let us revisit the GunPoint dataset, which is particularly beloved by the ETSC community [7, 8, 21, 23, 24, 27, 28, 29, 30, 32, 35, 37]. We have deep insights into this dataset: in order to create a simple-to-use dataset, a metronome is used to synchronize the performance of the behaviors (pointing or aiming). The metronome sounded a "beep" every five seconds, and the two "actors" were given the following brief: "*When you hear the cue, wait about a second, do the behavior for about two seconds, then return your hand to the side for the remaining time.*" As shown in **Figure 4.9**, this means that the last one to two seconds of most of the GunPoint exemplars are non-informative and non-class discriminating sections where the hand was resting by the actors' side. In addition, as hinted by the dataset's name, the difference between the classes is mostly the actors' fumbling to remove the gun from the hostler, which happens at the *beginning* of the action.

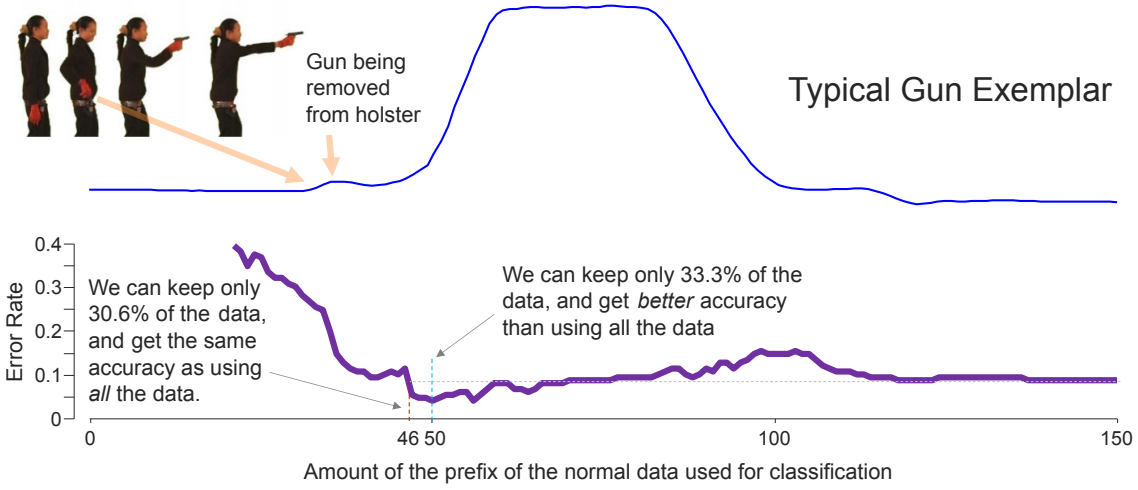

Figure 4.9: (*top*) A typical example from GunPoint annotated to show where the discriminating region is. (*bottom*) The holdout classification error-rate of every prefix of the GunPoint data from lengths 20 to 150 (the full length of the data).

The plot shown in **Figure 4.9** (*bottom*) resembles many plots shown in ETSC papers (actually, it is better than most of them, as we are correctly z-normalizing the truncated data, see **Table 4.1**). However, it is important to note that we are not claiming contribution by this plot, this is just basic data cleaning, not a publishable research model.

Note that a large number of UCR datasets have similar formatting conventions, some "events" bookended by constant regions that are simply there to make all the data objects have the same length (CricketX, CBF, Trace, etc.). Thus, it seems possible that some (possibly a very large) fraction of the apparent success of ETSC may be due to nothing more than a formatting convention that padded the right side of events with uninformative data, just to make the objects the same length.

4.5 On the Term *Early Classification*

The term "Early Classification" is unfortunately overloaded and vague. There are several tasks that might be named as such, which do not fall under the purview of this chapter. For example:

- Suppose that a boiler is rated for at most 200 psi. If a sensor detects increasing pressure readings: 180, 181, 182, ..., it would make perfect sense to sound an early warning that the pressure may approach 200 psi. Note that this setting only considers the *value* of a time series, not the *shape* of the time series. The same is true for many medical domains: if a person's BMI is measured monthly and begins to creep up to 20, 21, 22, ..., it might be better for a doctor to suggest an intervention before it reaches 25. But again, only the *value*, not the *shape* matters.

- Monitoring of batch processes is a slight generalization of the above. At every time point in a single run (plus or minus some "wiggle room" that can be modeled [96]), we know what range of values are acceptable. If the reading begins to drift outside that range, we can sound an alarm. Once again, this problem only considers the *value* of a time series, not the *shape* of the time series.

- Suppose that a chicken engaging in dustbathing more than 40 times a day is required to be culled by local ordinance (because dustbathing is often caused by the presence of mites or other pests) [97]. If we detect 10 bouts of dustbathing one day and 25 the next day, we may want to take some early intervention. Note that this setting only considers the *frequency* of (fully observed, not "early" observed) behaviors.

More generally, there may be other problems that have been labeled "early classification" by someone. We make no claims about such work. Our claims are limited to the sense of early classification used in [7, 8, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37], where the prefix of the *shape* of the time series is assumed to contain information that we can act upon before seeing the remainder of the shape.

4.6 Objections to Our Claims

Given the unusual nature of our claims, we solicited feedback from the community while drafting this chapter. We did this by writing to every author that published a paper on ETSC, and by general postings on discussion boards such as *r/MachineLearning*.

Most of the feedback has been (gratefully) incorporated into the main text. Here, we respond to a few questions that are worth addressing:

- **Q)** *Doesn't the fact that there are commercial predictive text algorithms for handwriting tell us that the prefix/inclusion/homophone problems can be overcome?*

  A) These systems are not doing predictive classification based on *words*, they are classifying individual *letters*, and then using classic ASCII predictive text algorithms. Moreover, as the Google help page notes "*Stand-alone symbols that are just a line (1/l/I) or circle (o/O/0) can be difficult to distinguish*" [98], exactly because those groups of symbols appear as homophones in time series space.

- **Q)** *Your claim* "it is not clear any of them could ever work in a real-world setting" *seems too strong.*

  A) Let us clarify what it means for a model to "work" here. Simply producing plots like **Figure 4.8** is not sufficient. Every event we are trying to detect has a cost. For concreteness, let us consider petrochemical engineering, and say the target event is the undesirable foaming of a distillation column. Assume it costs $1,000 to clean out the apparatus after such an event. Let us further imagine that if we get "early" notice that this is about to happen, we can warn an engineer to throttle some valve, and stop the damage. This action must also have some cost, let us say $200. Thus, in order for an ETSC model to be said to work, it must at least break even, producing at least one true positive for every five false positives. A handful of ETSC papers do have costs built into their models [29, 36], but they only test on UCR datasets and never estimate costs for any real-world applications. The results shown in this chapter suggest that the vast majority of positives will be false positives. For example, we applied the model in [21] to the GunPoint problem, with the exemplars inserted in

between long stretches of random walks, and we see thousands of false positives for every true positive (see [40]).

- **Q)** *Doesn't the homophone problem imply that all time series classification is hard, not just ETSC?*

  A) Yes, it does to some extent. Even if you ignore the issues of *early* classification, and consider only *classic* time series classification, the UCR datasets seem to have led to an illusion of progress. However, at least some applications do bypass this problem. For example, there are many papers on using the time series obtained from the sensors in a Wii Remote to classify gestures as inputs to the system. Normally, the user presses a button that indicates "start classifying" and releases it once the gesture is recognized. This means that the algorithm is not asked to deal with spurious data that might be thousands of times more frequent than target data. Such uses of time series classification *do* largely fit into the UCR format assumptions. Likewise, objects that come from the *spectrogram* and (converted from 2D) *shape* datatypes are presented as discrete vectors, not part of a stream.

- **Q)** *I don't see why z-normalization would be imperative in all real problems.*

  A) We think this question has been addressed in [95] and elsewhere by the community. However, in brief: it *is* meaningful to compare time series based on z-normalized shape; it is *sometimes* meaningful to compare time series based on mean value; but it is almost never meaningful to cluster on *both* at the same time (which is equivalent to comparing non-normalized time series with shape measures). The reason is that even small differences in the mean (and/or the standard deviation) com-

pletely drown out any shape information. In other words, for *non-normalized* data, $dist(\text{mean}(a), \text{mean}(b)) \propto dist(a, b)$ where *dist* is the Euclidean distance or DTW, etc. To summarize, if z-normalization is not important in your domain, it is virtually certain that the *shapes* do not matter – only the absolute values do. We make no claim about such situations other than the obvious empirical observation that such domains are very rare.

4.7 Conclusion

The time series early classification task as commonly understood may not be a meaningful problem to solve. All current research efforts that address this problem will be condemned to being overwhelmed by false positives if actually deployed in a real-world setting. Of course, false positives are a fact of life for any machine learning problem. However, the unique claims of *immediate actionability* mean that these false positives will have a cost, and the false positives may be many orders of magnitude more common than true positives. In addition, virtually all the algorithms are making the assumption that the data they are seeing *now* is normalized relative to data that only exists in the *future*. All those algorithms are condemned to producing mostly false negatives.

We believe that the issue is not with the proposed algorithms per se. The issue is that the definition of the problem itself is intrinsically underspecified and vague. The following are our recommendations to bring clarity to the ETSC area:

- An effort should be made to provide a concrete, testable, falsifiable, and useful definition of early classification of time series. While we have no interest in providing this

definition (in any case, a consortium of researchers would be better), we believe that any such definition would, at a minimum, have to consider:

1. The cost of a false positive for the actionable class(es) vs. the cost of a false negative [29, 36]. Even if the only early action taken is to sound an alarm, false alarm fatigue is known to have a high cost [92].
2. The probability that the domain of interest contains *prefixes*, *inclusions*, and *homophones* that resemble the actionable class(es).
3. The prior probability of seeing a member of the actionable class(es).
4. The appropriateness of the normalization assumptions for the domain.

- Anyone proposing an ETSC model needs to carefully explain what the model offers beyond simply classification with trivial awareness that not all datapoints matter (recall **Figure 4.9**).
- It is hard to see how any genuine progress could be made without access to a real-world publicly available dataset(s) that could benefit from the more concrete definition. The overreliance on the UCR datasets seems to have led the community astray here. Proxy datasets and synthetic datasets *do* have their place in research, especially in fledging areas. However, we are now two decades and many dozens of papers into this area.

It is hard to overemphasize the last point. If no real-world publicly available dataset(s) where some form of ETSC is useful can be obtained, this seems tantamount to saying that there is no problem to solve, and the community should stop publishing on this topic. It is stunning to think of the ease with which a grad student can obtain seismic data

recorded on Mars, or the mitochondria DNA of a mammoth that has been extinct for a million years, yet everyone publishing on ETSC must resort to proxy datasets.

Chapter 5

Conclusions

We unfortunately discovered that several highly cited papers [2, 3, 4, 5, 6, 7, 8] in the field of data mining have surprising problems with their proposed algorithm, dataset, or definition.

The widely used FastDTW algorithm [2] does not actually achieve its proposed speedup in any realistic settings. The vast majority of the papers using FastDTW would have been better off using cDTW, which is *faster* in 99% of all use cases, and more importantly, produces *exact* results. FastDTW can fail to approximate well, and there are no guidelines *when* this would happen. The takeaway is worth noting: we should be little bit surprised and humbled that such a poor idea has been "taken as gospel" for over 15 years and in more than 2,500 papers. What *other* bad ideas are out there?

The frequently tested TSAD benchmark datasets from Yahoo [3], Numenta [4], NASA [5] or Pei's Lab (SMD) [6], etc., suffer from one or four flaws: *triviality*, *unrealistic anomaly density*, *mislabeled ground truth* and *run-to-failure bias*. These flaws make them

unsuitable for evaluating or comparing anomaly detection algorithms, thus existing benchmarks should be abandoned. On the bright side, we introduced UCR Time Series Anomaly Archive [20] that is largely free of aforementioned four flaws. We wish this work could act as an inspiration to the community on creating a crowdsourced set of benchmark datasets.

The commonly understood ETSC task may not be a meaningful problem to solve. Almost all ETSC algorithms [7, 8, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37] made a false assumption that the data they are seeing *now* is normalized relative to data that only exists in the *future*. However, we believe that the issue is not with algorithms per se, but the unclear definition of the ETSC problem itself. The vague and underspecified definition of ETSC suffers from three issues: *prefix*, *inclusion*, *homophone*. We recommend the community to provide a concrete definition with the consideration of the issues we observed and produce real-world publicly available datasets.

While this work is very recent, there has already been an impact in the community.

We wrote to the author of [62] who had used FastDTW. He agreed to do an independent test (see **Section 2.5**). He found that "(cDTW) *was approx. 24x faster than FastDTW on average*" and "*in the 'slowest' case,* (cDTW) *was still approx. 5.8x faster than FastDTW*". He concluded that "*these tests suggest that* (cDTW) *is indeed superior in terms of speed as well as for usage in time-series classification.*"

A team from UC Berkeley had been using FastDTW for many projects [99]. After they read our work on FastDTW, they wrote "*... At the time, I chose FastDTW ... More recently, I had the opportunity to develop a real-time streaming anomaly detection algorithm for my present company, ... results confirm that FastDTW is the slowest among all ...*".

Besides direct replies we received from authors that had recently used FastDTW, several other papers also observed that FastDTW does not achieve expected optimization over DTW:

- "*this is because* (our time series) *are usually short, which makes the cost of* (FastDTW) *exceed the cost of calculating DTW directly*" [58].
- "*The difference between the computation times* (of FastDTW and DTW) *was already analysed in* (Wu and Keogh's work) *and could therefore be verified*" [64].
- "*In contrast, fast-DTW was actually slower than standard DTW, thereby confirming the findings of Wu and Keogh*" [65].
- "*We observe that for this implementation of DTW indeed FastDTW is outperformed frequently*" [66].
- "*we confirmed an independent claim that FastDTW can in actuality be slower than DTW in certain cases*" [67].
- "(We) *also confirm the claim that cDTW is faster and more accurate than FastDTW, which is widely misunderstood and misused*" [68].
- "*We found that the cDTW algorithm has a faster recognition speed than the fastdtw algorithm in the air writing recognition system*" [69].

We sent a preview of our work on TSAD benchmarks to a team from CMU / Rice / Penn State that was originally very positive about deep learning for time series. They seems to have completely rebooted. Their recent paper cites our work, and notes

"*Surprisingly, we observe that some classical algorithms could outperform many recent deep learning approaches*" [100].

At the time of writing, our work on TSAD benchmarks has received 197 citations in just three years. Recent research has also cast some doubt on the apparent progress in TSAD. For example,

- [101] shows that "*The results for the modified Eclipse dataset . . . are rather surprising, as the one-liner baseline methods show perfect detection across all metrics*".
- [102] suggests that the popular "*SWaT and WADI benchmarks are highly unreliable and that these datasets are not suited for multivariate time-series AD evaluation*".
- [103] notes that "*classical machine learning methods generally outperform deep learning methods across a range of* (time series) *anomaly types*".
- [104] concludes that "*This event* (in SWaT) *contributes to the F1 score much more than other events and explains the relatively high scores of many algorithms*".
- [105] claims that "*recently proposed deep* (learning) *algorithms fail to effectively detect even these simple anomalies in* (multivariate time series) *datasets*".

We really appreciate these people who treat our work seriously and take our recommendations into consideration. Our observations have a more sobering lesson for the community. Our work may serve as a reminder to the community to exercise more caution in uncritically accepting published results.

Bibliography

[1] David J. Hand. Data mining: Statistics and more? *The American Statistician*, 52(2):112–118, 1998.

[2] Stan Salvador and Philip Chan. Toward accurate dynamic time warping in linear time and space. *Intelligent Data Analysis*, 11(5):561–580, 2007.

[3] Nikolay Laptev, Saeed Amizadeh, and Youssef Billawala. S5 - a labeled anomaly detection dataset, version 1.0 (16M), March 2015. `https://webscope.sandbox.yahoo.com/catalog.php?datatype=s&did=70`.

[4] Subutai Ahmad, Alexander Lavin, Scott Purdy, and Zuha Agha. Unsupervised real-time anomaly detection for streaming data. *Neurocomputing*, 262:134–147, 2017.

[5] Kyle Hundman, Valentino Constantinou, Christopher Laporte, Ian Colwell, and Tom Soderstrom. Detecting spacecraft anomalies using LSTMs and nonparametric dynamic thresholding. In *Proc. 24th ACM SIGKDD Intl. Conf. Knowledge Discovery and Data Mining*, pages 387–395, 2018.

[6] Ya Su, Youjian Zhao, Chenhao Niu, Rong Liu, Wei Sun, and Dan Pei. Robust anomaly detection for multivariate time series through stochastic recurrent neural network. In *Proc. 25th ACM SIGKDD Intl. Conf. Knowledge Discovery and Data Mining*, pages 2828–2837, 2019.

[7] Zhengzheng Xing, Jian Pei, and Philip S. Yu. Early classification on time series. *Knowledge and Information Systems*, 31(1):105–127, 2012.

[8] Zhengzheng Xing, Jian Pei, Philip S. Yu, and Ke Wang. Extracting interpretable features for early classification on time series. In *Proc. 2011 SIAM Intl. Conf. Data Mining*, pages 247–258, 2011.

[9] Chotirat Ann Ratanamahatana and Eamonn Keogh. Everything you know about dynamic time warping is wrong. In *Proc. 3rd Intl. Workshop Mining Temporal and Sequential Data*, volume 1, pages 53–63, 2004.

[10] 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 *Proc. 18th ACM SIGKDD Intl. Conf. Knowledge Discovery and Data Mining*, pages 262–270, 2012.

[11] Abdullah Mueen, Nikan Chavoshi, Noor Abu-El-Rub, Hossein Hamooni, Amanda Minnich, and Jonathan MacCarthy. Speeding up dynamic time warping distance for sparse time series data. *Knowledge and Information Systems*, 54(1):237–263, 2018.

[12] Chang Wei Tan, Matthieu Herrmann, Germain Forestier, Geoffrey I. Webb, and François Petitjean. Efficient search of the best warping window for dynamic time warping. In *Proc. 2018 SIAM Intl. Conf. Data Mining*, pages 225–233. Society for Industrial and Applied Mathematics, 2018.

[13] Hoang Anh Dau, Eamonn Keogh, Kaveh Kamgar, Chin-Chia Michael Yeh, Yan Zhu, Shaghayegh Gharghabi, Chotirat Ann Ratanamahatana, Yanping Chen, Bing Hu, Nurjahan Begum, Anthony Bagnall, Abdullah Mueen, Gustavo Batista, and Hexagon-ML. The UCR time series classification archive, October 2018. `https://www.cs.ucr.edu/~eamonn/time_series_data_2018/`.

[14] Miguel Pfitscher, Daniel Welfer, Evaristo José Do Nascimento, Marco Antonio De Souza Leite Cuadros, and Daniel Fernando Tello Gamarra. Article users activity gesture recognition on kinect sensor using convolutional neural networks and Fast-DTW for controlling movements of a mobile robot. *Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial*, 22(63):121–134, 2019.

[15] Miguel Pfitscher, Daniel Welfer, Marco Antonio de Souza Leite Cuadros, and Daniel Fernando Tello Gamarra. Activity gesture recognition on kinect sensor using convolutional neural networks and FastDTW for the MSRC-12 dataset. In *Proc. 18th Intl. Conf. Intelligent Systems Design and Applications*, pages 230–239, 2019.

[16] Kun Qian, Ge Gao, Fang Fang, and Liangjun Zhang. RGB-D based daily activity recognition for service robots using clustering with gaussian mixtures and FastDTW. In *Proc. 2016 Intl. Conf. Advanced Robotics and Mechatronics*, pages 651–656, 2016.

[17] Kevin Yeo, Ooi Shih Yin, Pang Ying Han, and Wee Kuok Kwee. Real time mobile application of in-air signature with fast dynamic time warping (FastDTW). In *Proc. 2015 IEEE Intl. Conf. Signal and Image Processing Applications*, pages 315–320, 2015.

[18] Jurgen Lohrer and Markus Lienkamp. Building representative velocity profiles using FastDTW and spectral clustering. In *Proc. 14th Intl. Conf. ITS Telecommunications*, pages 45–49, 2015.

[19] E. S. Page. On problems in which a change in a parameter occurs at an unknown point. *Biometrika*, 44(1-2):248–252, 1957.

[20] E. Keogh, T. Dutta Roy, U. Naik, and A. Agrawal. Time-series anomaly detection datasets, SIGKDD 2021, June 2021. `https://www.cs.ucr.edu/~eamonn/time_series_data_2018/UCR_TimeSeriesAnomalyDatasets2021.zip`.

[21] Patrick Schäfer and Ulf Leser. TEASER: early and accurate time series classification. *Data Mining and Knowledge Discovery*, 34(5):1336–1362, 2020.

[22] Marc Rußwurm, Sébastien Lefèvre, Nicolas Courty, Rémi Emonet, Marco Körner, and Romain Tavenard. End-to-end learning for early classification of time series. *arXiv*, January 2019. `https://arxiv.org/abs/1901.10681`.

[23] Mohamed F. Ghalwash, Vladan Radosavljevic, and Zoran Obradovic. Utilizing temporal patterns for estimating uncertainty in interpretable early decision making. In *Proc. 20th ACM SIGKDD Intl. Conf. Knowledge Discovery and Data Mining*, pages 402–411, 2014.

[24] Usue Mori, Alexander Mendiburu, Eamonn Keogh, and Jose A. Lozano. Reliable early classification of time series based on discriminating the classes over time. *Data Mining and Knowledge Discovery*, 31(1):233–263, 2017.

[25] Nathan Parrish, Hyrum S. Anderson, Maya R. Gupta, and Dun Yu Hsiao. Classifying with confidence from incomplete information. *J. Machine Learning Research*, 14(76):3561–3589, 2013.

[26] Asma Dachraoui, Alexis Bondu, and Antoine Cornuéjols. Early classification of time series as a non myopic sequential decision making problem. In *Proc. European Conf. Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, pages 433–447, 2015.

[27] Usue Mori, Alexander Mendiburu, Sanjoy Dasgupta, and Jose A. Lozano. Early classification of time series by simultaneously optimizing the accuracy and earliness. *IEEE Trans. Neural Networks and Learning Systems*, 29(10):4569–4578, 2018.

[28] Wenlin Wang, Changyou Chen, Wenqi Wang, Piyush Rai, and Lawrence Carin. Earliness-aware deep convolutional networks for early time series classification. *arXiv*, November 2016. `https://arxiv.org/abs/1611.04578`.

[29] Romain Tavenard and Simon Malinowski. Cost-aware early classification of time series. In *Proc. 2016 European Conf. Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, pages 632–647, 2016.

[30] C. Martinez, G. Perrin, E. Ramasso, and M. Rombaut. A deep reinforcement learning approach for early classification of time series. In *Proc. 26th European Signal Processing Conf.*, pages 2030–2034, 2018.

[31] Hyrum S. Anderson, Nathan Parrish, Kristi Tsukida, and Maya R. Gupta. Reliable early classification of time series. In *Proc. 2012 IEEE Intl. Conf. Acoustics, Speech and Signal Processing*, pages 2073–2076, 2012.

[32] Liuyi Yao, Yaliang Li, Yezheng Li, Hengtong Zhang, Mengdi Huai, Jing Gao, and Aidong Zhang. DTEC: Distance transformation based early time series classification. In *Proc. 2019 SIAM Intl. Conf. Data Mining*, pages 486–494, 2019.

[33] Junwei Lv, Xuegang Hu, Lei Li, and Peipei Li. An effective confidence-based early classification of time series. *IEEE Access*, 7:96113–96124, 2019.

[34] Guoliang He, Wen Zhao, and Xuewen Xia. Confidence-based early classification of multivariate time series with multiple interpretable rules. *Pattern Analysis and Applications*, 23(2):567–580, 2019.

[35] Anshul Sharma and Sanjay Kumar Singh. Early classification of time series based on uncertainty measure. In *Proc. 2019 IEEE Conf. Information and Communication Technology*, pages 1–6, 2019.

[36] Youssef Achenchabe, Alexis Bondu, Antoine Cornuéjols, and Asma Dachraoui. Early classification of time series: Cost-based optimization criterion and algorithms. *Machine Learning*, 110(6):1481–1504, 2021.

[37] Wenhe Yan, Guiling Li, Zongda Wu, Senzhang Wang, and Philip S. Yu. Extracting diverse-shapelets for early classification on time series. *World Wide Web*, 23(6):3055–3081, 2020.

[38] Renjie Wu and Eamonn J. Keogh. Supporting page for FastDTW is approximate and generally slower than the algorithm it approximates, March 2020. `https://wu.renjie.im/research/fastdtw-is-slow/`.

[39] Renjie Wu and Eamonn J. Keogh. Supporting page for current time series anomaly detection benchmarks are flawed and are creating the illusion of progress, September 2020. `https://wu.renjie.im/research/anomaly-benchmarks-are-flawed/`.

[40] Renjie Wu, Audery Der, and Eamonn J. Keogh. Supporting page for when is early classification of time series meaningful, February 2021. `https://wu.renjie.im/research/ETSC-is-meaningless/`.

[41] Renjie Wu and Eamonn J. Keogh. FastDTW is approximate and generally slower than the algorithm it approximates. *IEEE Trans. Knowledge and Data Engineering*, 34(8):3779–3785, 2022.

[42] Renjie Wu and Eamonn J. Keogh. FastDTW is approximate and generally slower than the algorithm it approximates (Extended Abstract). In *Proc. 37th IEEE Intl. Conf. Data Engineering*, 2021.

[43] Renjie Wu and Eamonn J. Keogh. Current time series anomaly detection benchmarks are flawed and are creating the illusion of progress. *IEEE Trans. Knowledge and Data Engineering*, 35(3):2421–2429, 2023.

[44] Renjie Wu and Eamonn J. Keogh. Current time series anomaly detection benchmarks are flawed and are creating the illusion of progress (Extended Abstract). In *Proc. 38th IEEE Intl. Conf. Data Engineering*, 2022.

[45] Renjie Wu, Audrey Der, and Eamonn J. Keogh. When is early classification of time series meaningful? *IEEE Trans. Knowledge and Data Engineering*, 35(3):3253–3260, 2023.

[46] Renjie Wu, Audrey Der, and Eamonn J. Keogh. When is early classification of time series meaningful? (Extended Abstract). In *Proc. 38th IEEE Intl. Conf. Data Engineering*, 2022.

[47] Yingqiu Cao, Nikolai Rakhilin, Philip H. Gordon, Xiling Shen, and Edwin C. Kan. A real-time spike classification method based on dynamic time warping for extracellular enteric neural recording with large waveform variability. *J. Neuroscience Methods*, 261:97–109, 2016.

[48] Huiwen Zhang, Mingliang Fu, Haitao Luo, and Weijia Zhou. Robust human action recognition using dynamic movement features. In *Proc. 10th Intl. Conf. Intelligent Robotics and Applications*, pages 474–484, 2017.

[49] Manika Kapoor and David C. Anastasiu. A data-driven approach for detecting autism spectrum disorders. In *Proc. 2nd Intl. Data Science Conf.*, pages 51–56, 2019.

[50] Ankur Gupta. *Using Unlabeled 3D Motion Examples for Human Activity Understanding*. PhD thesis, Dept. Computer Science, Faculty of Science, Univ. of British Columbia, 2016.

[51] Christine Kühnel, Tilo Westermann, Fabian Hemmert, Sven Kratz, Alexander Müller, and Sebastian Möller. I'm home: Defining and evaluating a gesture set for smart-home control. *Intl. J. Human-Computer Studies*, 69(11):693–704, 2011.

[52] Zhuolin Jiang, Zhe Lin, and Larry S. Davis. Recognizing human actions by learning and matching shape-motion prototype trees. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 34(3):533–547, 2012.

[53] Melih C. Yesilli, Firas A. Khasawneh, and Andreas Otto. Chatter detection in turning using machine learning and similarity measures of time series via dynamic time warping. *arXiv*, August 2019. `https://arxiv.org/abs/1908.01678/`.

[54] Thomas W. Mitchel, Sipu Ruan, and Gregory S. Chirikjian. Signal alignment for humanoid skeletons via the globally optimal reparameterization algorithm. In *Proc. 18th IEEE-RAS Intl. Conf. Humanoid Robots*, pages 217–223, 2018.

[55] Taegyun Kwon, Dasaem Jeong, and Juhan Nam. Audio-to-score alignment of piano music using RNN-based automatic music transcription. In *Proc. 14th Sound and Music Computing Conf.*, pages 380–385, 2017.

[56] Stan Salvador and Philip Chan. An email chain of correspondence with Stan and Philip, March 2020. `https://wu.renjie.im/research/fastdtw-is-slow/emails/stan-and-philip/#philip-on-feb-27-2020-1301`.

[57] Thanawin Rakthanmanon, Bilson Campana, Abdullah Mueen, Gustavo Batista, Brandon Westover, Qiang Zhu, Jesin Zakaria, and Eamonn Keogh. UCR Suite: Fast nearest neighbor search (top-1 NN), August 2012. `https://www.youtube.com/watch?v=d_qLzMMuVQg`.

[58] Yongpan Zou, Dan Wang, Shicong Hong, Rukhsana Ruby, Dian Zhang, and Kaishun Wu. A low-cost smart glove system for real-time fitness coaching. *IEEE Internet of Things J.*, 7(8):7377–7391, 2020.

[59] Pavel Jurak, Josef Halamek, Pavel Leinveber, Vlastimil Vondra, Ladislav Soukup, Petr Vesely, Josef Sumbera, Karel Zeman, Libuse Martinakova, Tereza Jurakova, and Miroslav Novak. Ultra-High-Frequency ECG measurement. In *Computing in Cardiology*, pages 783–786, 2013.

[60] Ohhwan Kwon, Jinwoo Jeong, Hyung Bin Kim, In Ho Kwon, Song Yi Park, Ji Eun Kim, and Yuri Choi. Electrocardiogram sampling frequency range acceptable for heart rate variability analysis. *Healthcare Informatics Research*, 24(3):198–206, 2018.

[61] Radu-Daniel Vatavu. The impact of motion dimensionality and bit cardinality on the design of 3D gesture recognizers. *Intl. J. Human-Computer Studies*, 71(4):387–409, 2013.

[62] Pascal Schneider, Raphael Memmesheimer, Ivanna Kramer, and Dietrich Paulus. Gesture recognition in RGB videos using human body keypoints and dynamic time warping. In *Robot World Cup XXIII*, pages 281–293, 2019.

[63] Pascal Schneider. An email chain of correspondence with Pascal, March 2020. `https://wu.renjie.im/research/fastdtw-is-slow/emails/pascal/#pascal-on-mar-10-2020-0920`.

[64] Manuel Schneider, Norbert Greifzu, Lei Wang, Christian Walther, Andreas Wenzel, and Pu Li. An end-to-end machine learning approach with explanation for time series with varying lengths. *Neural Computing and Applications*, 36(13):7491–7508, 2024.

[65] Kenan Li, Katherine Sward, Huiyu Deng, John Morrison, Rima Habre, Meredith Franklin, Yao-Yi Chiang, Jose Luis Ambite, John P. Wilson, and Sandrah P. Eckel. Using dynamic time warping self-organizing maps to characterize diurnal patterns in environmental exposures. *Scientific Reports*, 11(1), 2021.

[66] Dominik Bünger, Miriam Gondos, Lucile Peroche, and Martin Stoll. An empirical study of graph-based approaches for semi-supervised time series classification. *Frontiers in Applied Mathematics and Statistics*, 7, 2022.

[67] David Grethlein, Aleksanteri Sladek, and Santiago Ontañón. Identifying on-road scenarios predictive of ADHD using driving simulator time series data. *arXiv*, 2021.

[68] Yuqi Luo, Wei Ke, and Chan-Tong Lam. Wearable real-time air-writing system employing knn and constrained dynamic time warping. In *Proc. 2023 IEEE Wireless Communications and Networking Conference*, 2023.

[69] Yuao Ye, Jiang Liu, Wen Zhao, Peng Liu, and Shigeru Shimamoto. Machine learning based on air-writing recognition system. In *Proc. 2023 Intl. Conf. Intelligent Computing and Control*, 2023.

[70] 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 *Proc. 16th IEEE Intl. Conf. Data Mining*, pages 1317–1322, 2016.

[71] Xuanhao Chen, Liwei Deng, Feiteng Huang, Chengwei Zhang, Zongquan Zhang, Yan Zhao, and Kai Zheng. DAEMON: Unsupervised anomaly detection and interpretation for multivariate time series. In *Proc. 37th IEEE Intl. Conf. Data Engineering*, 2021.

[72] Sebastian Schmidl, Phillip Wenig, and Thorsten Papenbrock. Anomaly detection in time series: A comprehensive evaluation. *Proc. VLDB Endowment*, 15(9):1779–1797, 2022.

[73] Juan Qiu, Qingfeng Du, and Chongshu Qian. KPI-TSAD: A time-series anomaly detector for KPI monitoring in cloud applications. *Symmetry*, 11(11):1350, 2019.

[74] Junyoung Park, Dong In Kim, Byoungjo Choi, Woochul Kang, and Hyung Wook Kwon. Classification and morphological analysis of vector mosquitoes using deep convolutional neural network. *Scientific Reports*, 10(1):1012, 2020.

[75] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM Computing Surveys*, 41(3):1–58, 2009.

[76] Ane Blázquez-García, Angel Conde, Usue Mori, and Jose A. Lozano. A review on outlier/anomaly detection in time series data. *ACM Computing Surveys*, 54(3):1–33, 2021.

[77] Iman Vasheghani Farahani, Alex Chien, Russell E. King, Michael G. Kay, and Brad Klenz. Time series anomaly detection from a markov chain perspective. In *Proc. 18th IEEE Intl. Conf. Machine Learning and Applications*, pages 1000–1007, 2019.

[78] Sina Däubener, Sebastian Schmitt, Hao Wang, Thomas Bäck, and Peter Krause. Large anomaly detection in univariate time series: An empirical comparison of machine learning algorithms. In *19th Industrial Conf. Data Mining*, July 2019. preprint, `https://www.honda-ri.de/pubs/pdf/4015.pdf`.

[79] Minh Tuan Doan, Sutharshan Rajasegarar, Mahsa Salehi, Masud Moshtaghi, and Christopher Leckie. Profiling pedestrian distribution and anomaly detection in a dynamic environment. In *Proc. 24th ACM Intl. Conf. Information and Knowledge Management*, pages 1827–1830, 2015.

[80] Hanyu Zhang, Che-Lun Hung, Meiyuan Liu, Xiaoye Hu, and Yi-Yang Lin. NCNet: Deep learning network models for predicting function of non-coding DNA. *Frontiers in Genetics*, 10:432, 2019.

[81] Chengqiang Huang, Geyong Min, Yulei Wu, Yiming Ying, Ke Pei, and Zuochang Xiang. Time series anomaly detection for trustworthy services in cloud computing systems. *IEEE Trans. Big Data*, 8(1):60–72, 2022.

[82] Chengqiang Huang. *Featured Anomaly Detection Methods and Applications*. PhD thesis, College of Engineering, Mathematics and Physical Sciences, Univ. of Exeter, 2018.

[83] Jingkun Gao, Xiaomin Song, Qingsong Wen, Pichao Wang, Liang Sun, and Huan Xu. RobustTAD: Robust time series anomaly detection via decomposition and convolutional neural networks. *arXiv*, February 2020. `https://arxiv.org/abs/2002.09545`.

[84] Dragomir Yankov, Eamonn Keogh, and Umaa Rebbapragada. Disk aware discord discovery: Finding unusual time series in terabyte sized datasets. In *Proc. 7th IEEE Intl. Conf. Data Mining*, pages 381–390, 2007.

[85] Takaaki Nakamura, Makoto Imamura, Ryan Mercer, and Eamonn Keogh. MERLIN: Parameter-free discovery of arbitrary length anomalies in massive time series archives. In *Proc. 2020 IEEE Intl. Conf. Data Mining*, pages 1190–1195, 2020.

[86] Nesime Tatbul, Tae Jun Lee, Stan Zdonik, Mejbah Alam, and Justin Gottschlich. Precision and recall for time series. In *Proc. 32nd Intl. Conf. Neural Information Processing Systems*, pages 1924–1934, 2018.

[87] ABC News. Protests erupt after Eric Garner grand jury decision: Eighty-three people were arrested in New York City demonstrations, December 2014. `https://abcnews.go.com/US/protests-erupt-eric-garner-grand-jury-decision/story?id=27355323`.

[88] Jehn-Ruey Jiang, Jian-Bin Kao, and Yu-Lin Li. Semi-supervised time series anomaly detection based on statistics and deep learning. *Applied Sciences*, 11(15):6698, 2021.

[89] Shaghayegh Gharghabi, Shima Imani, Anthony Bagnall, Amirali Darvishzadeh, and Eamonn Keogh. An ultra-fast time series distance measure to allow data mining in more complex real-world deployments. *Data Mining and Knowledge Discovery*, 34(4):1104–1135, 2020.

[90] Gustavo E. Batista, Eamonn J. Keogh, Agenor Mafra-Neto, and Edgar Rowton. SIGKDD demo: sensors and software to allow computational entomology, an emerging application of data mining. In *Proc. 17th ACM SIGKDD Intl. Conf. Knowledge discovery and Data Mining*, pages 761–764, 2011.

[91] Sean Kauffman, Murray Dunne, Giovani Gracioli, Waleed Khan, Nirmal Benann, and Sebastian Fischmeister. Palisade: A framework for anomaly detection in embedded systems. *J. Systems Architecture*, 113:101876, 2021.

[92] Kelly Creighton Graham and Maria Cvach. Monitor alarm fatigue: Standardizing use of physiological monitors and decreasing nuisance alarms. *American J. Critical Care*, 19(1):28–34, 2010.

[93] Juan J. Rodríguez, Carlos J. Alonso, and Henrik Boström. Boosting interval based literals. *Intelligent Data Analysis*, pages 245–262, 2001.

[94] German Working Cattle Group. Training cattle, June 2016. https://www.zugrinder.de/en/training-cattle.html.

[95] Thanawin Rakthanmanon, Bilson Campana, Abdullah Mueen, Gustavo Batista, Brandon Westover, Qiang Zhu, Jesin Zakaria, and Eamonn Keogh. Addressing big data time series: Mining trillions of time series subsequences under dynamic time warping. *ACM Trans. Knowledge Discovery from Data*, 7(3):1–31, 2013.

[96] 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 *Proc. 25th ACM SIGKDD Intl. Conf. Knowledge Discovery and Data Mining*, pages 2604–2612, 2019.

[97] Alireza Abdoli, Sara Alaee, Shima Imani, Amy Murillo, Alec Gerry, Leslie Hickle, and Eamonn Keogh. Fitbit for chickens?: Time series data mining can increase the productivity of poultry farms. In *Proc. 26th ACM SIGKDD Intl. Conf. Knowledge Discovery and Data Mining*, pages 3328–3336, 2020.

[98] Google. Google handwriting input, April 2015. https://support.google.com/faqs/faq/6188721?hl=en.

[99] Liyu Wang, Jack Hodges, Dan Yu, and Ronald S. Fearing. Automatic modeling and fault diagnosis of car production lines based on first-principle qualitative mechanics and semantic web technology. *Advanced Engineering Informatics*, 49:101248, 2021.

[100] Kwei-Herng Lai, Daochen Zha, Junjie Xu, Yue Zhao, Guanchu Wang, and Xia Hu. Revisiting time series outlier detection: Definitions and benchmarks. In *Proc. 35th Intl. Conf. Neural Information Processing Systems Datasets and Benchmarks Track*, 2021.

[101] Franz Kevin Stehle, Wainer Vandelli, Giuseppe Avolio, Felix Zahn, and Holger Fröning. DeepHYDRA: Resource-efficient time-series anomaly detection in dynamically-configured systems. In *Proc. 38th ACM Intl. Conf. Supercomputing*, 2024.

[102] Dennis Wagner, Tobias Michels, Florian C.F. Schulz, Arjun Nair, Maja Rudolph, and Marius Kloft. TimeSeAD: Benchmarking deep multivariate time-series anomaly detection. *Trans. Machine Learning Research*, 2023.

[103] Ferdinand Rewicki, Joachim Denzler, and Julia Niebling. Is it worth it? comparing six deep and classical methods for unsupervised anomaly detection in time series. *Applied Sciences*, 13(3):1778, 2023.

[104] Mohamed El Amine Sehili and Zonghua Zhang. Multivariate time series anomaly detection: Fancy algorithms and flawed evaluation methodology. *arXiv*, 2023.

[105] Astha Garg, Wenyu Zhang, Jules Samaran, Ramasamy Savitha, and Chuan-Sheng Foo. An evaluation of anomaly detection and diagnosis in multivariate time series. *IEEE Trans. Neural Networks and Learning Systems*, 33(6):2508–2517, 2022.