"Now you see it, now you don't!" Detecting Suspicious Pattern Absences in Continuous Time Series

Vincent Vercruyssen* Wannes Meert* Jesse Davis*

Abstract

Given its large applicational potential, time series anomaly detection has become a crucial data mining task. Its goal is to identify periods of a time series where there is a deviation from the expected behavior. Existing approaches focus on analyzing whether the currently observed behavior differs from previously seen, normal behavior. In contrast, this paper tackles the the task where the *absence* of a previously observed behavior is indicative of an anomaly. In other words, a pattern that is expected to recur in the time series is *absent*. In real-world use cases, absent patterns can be linked to serious problems. For instance, if a scheduled, regular maintenance operation of a machine does not take place, this can be harmful to the machine at a later time. In this paper, we introduce the task of detecting when a specific pattern is absent in a real-valued time series. We propose a novel technique called FZapPa that can address this task. Empirically, FZapPa outperforms existing anomaly techniques on a benchmark of real-world datasets.

Keywords — absent patterns; anomaly detection; time series; PU learning

1 Introduction

The standard time series anomaly detection task involves identifying portions of the data characterized by the presence of unexpected or abnormal behavior [7]. Figure 1b illustrates the canonical anomaly detection problem where the grey-shaded region highlights a pattern (i.e., a collection of points) that substantially differs from the other patterns present in the data such as the highlighted green bell-shaped pattern in Figure 1a. In contrast, this paper addresses detecting a radically different type of anomalous behavior: the *absence* of a pattern. The red-shaded region in Figure 1c shows a portion of the time series where we could reasonable expect to see an occurence of the bell-shaped pattern. Detecting such an anomaly is challenging as the observed measurements in the red-shaded region also correspond to typical, normal behavior in the time series.

In many real-world use cases, absent patterns often correspond to significant anomalies. Figure 2 shows an illustrative real-world resource monitoring use case. It involves monitoring a retail store's water usage over several weeks. The store's water distribution system contains a device that automatically performs self-cleaning. Although the water usage during a self-cleaning action is always slightly different, the highlighted segments in Figure 2 show that a recognizable pattern emerges in the data. A failure of the cleaning action neither causes any immediate problems nor any noticeable effects in the water usage data. However, after a while a sudden breakdown will occur that halts the water distribution and a large amount of water will be disposed through an unmonitored valve. Other real-world use cases, for instance, are found in network monitoring where CPU load is used to monitor data backup operations and recurring failover system tests.

An approach for detecting this type of anomaly requires determining when a *sporadically occurring pattern is absent*. Standard approaches struggle in this setting as it *violates the common assumption of anomaly detection* that frequent behavior is normal and infrequent behavior is abnormal. By definition, the most common behavior of the sporadically occurring pattern is absence. Furthermore, an absent pattern is not necessarily replaced by anomalous behavior. A successful approach to this task must address two challenging subproblems:

1. **Identify the pattern of interest and all its occurrences.** This poses two challenges. First, the pattern does not appear all that frequently, in this snippet of 64 days it only occurs six times. In contrast, most pattern mining algorithms focus on identifying frequent patterns. Additionally, there will be many other patterns in the data that occur more frequently than the pattern of interest. Second, the pattern appears at slightly different times during the day and its shape can vary.
2. **Learn how the pattern's occurrence is distributed within the time series.** In Figure 2, it is not immediately obvious what characteristics of the data are predictive of the pattern's occurrence. Its appearance does not follow a fixed time or interval schedule, there are gaps of three days, nine days, ten days, etc. between occurrences of the pattern. Similarly, the immediate context (i.e., the previous or next's days values) are not predictive of the pattern's occurrence. *This is a more*

*Department of Computer Science, KU Leuven, Belgium. firstname.lastname@cs.kuleuven.be

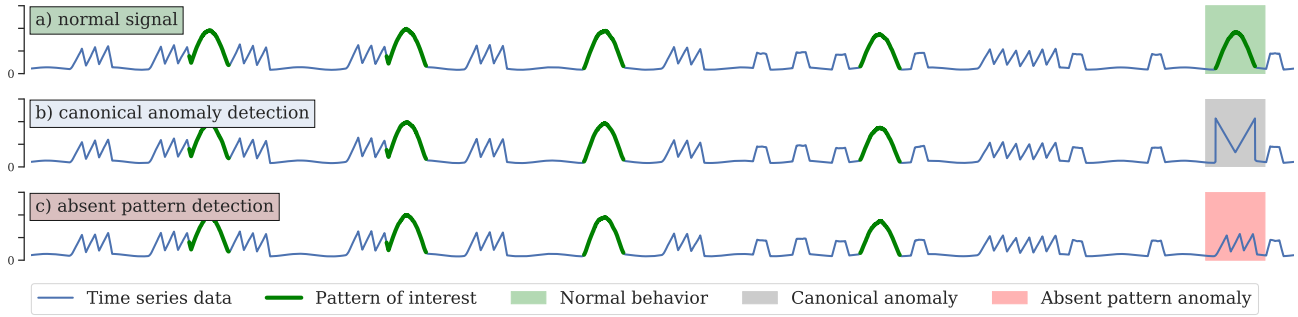

Figure 1: The figure contrasts canonical anomaly detection with *absent pattern detection* on a fictional time series fragment. The TOP plot shows the fragment without any anomalies. The MIDDLE plot shows the fragment with a single anomaly which is picked up by canonical detection approaches because it looks different from the rest of the observed behavior, The BOTTOM plot shows the fragment with an *absent pattern anomaly*. None of the classic approaches can detect it because the behavior observed in its place is not anomalous.

challenging variation of contextual anomaly detection [7] than is seen in most research because there is no direct link between the time series and the context.

Clearly, this is a problem setting that requires a data-driven solution. For instance, the retail company needs to monitor over 400 stores and the behavior of each store's cleaning device is unique and might change over time. Furthermore, feedback is given directly by local domain experts and should be processed without them requiring intimate knowledge of the approach.

Despite the prevalence of this type of anomaly in practice, little attention has been paid toward developing algorithms that can detect the suspicious absence of a pattern. This paper's contribution is to fill this gap by formalizing the problem of *absent pattern* detection and developing FZapPa, a data-driven algorithm that can automatically detect anomalies characterized by an absent pattern. First, FZapPa views pattern detection through the lens of learning from positive and unlabeled (PU) data. It uses a small number of example occurrences of the pattern to train a PU classifier, and the learned model is used to identify the remaining pattern occurrences. Second, FZapPa analyzes the time series to learn the relevant indirect context in which the pattern is expected to appear and models its occurrence using a probabilistic model. Based on this model, it then predicts an anomaly score to detect absent occurrences. We evaluate our FZapPa on three real-world water usage datasets and three real-world power usage datasets and find that FZapPa outperforms the state-of-the-art anomaly detection approaches.

2 Methodology

The problem setting illustrated in Figures 1 and 2 can be formalized as follows:

Given: A univariate time series $T = \{(t_i, v_1), \ldots, (t_n, v_n)\}$ that contains a sporadically reoccurring pattern $\mathcal{P}$, where a pattern is a specific shape of length m with $m \ll n$.

Do: Identify anomalous periods of T. That is, find places in T where the pattern $\mathcal{P}$ is expected to occur but does not.

By sporadically reoccurring, we mean that $\mathcal{P}$ occurs multiple times in T but that the number of times it occurs is much smaller than $\frac{n}{m}$.

To achieve this goal, we introduce an approach called *Finding Zapped Patterns* or FZapPa.[1] It partitions the time series into a sequence of overlapping segments $S_{1,l}, S_{2,l}, \ldots, S_{n-l,l}$, where segment $S_{i,l}$ is a contiguous subsequence of T starting at time t_i with length l and $l \ll n$. For each segment $S_{i,l}$, FZapPa computes an anomaly score, which is the probability that a pattern is expected to appear in $S_{i,l}$ but is absent given information about its previous occurrence and the current context:

$$score_i = P(absent_i | S_{i,l}, \ldots, S_{j,l}, \ldots, S_{k,l}) \tag{2.1}$$

where $S_{k,l}$ is most recent segment where the pattern was detected and $i > j > k$. A high score indicates a period that is likely to be anomalous.

FZapPa has three components. First, it must be able to detect occurrences of the pattern in T. It views this as a positive and unlabeled (PU) learning problem, because this step has only access to a few known example occurrences of the pattern. The trained PU model is used to detect the remaining occurrences. Second, it automatically learns a context of T that is fed to a Weibull distribution to model the distribution

[1] We loosely use the term *zapped* instead of *absent*.

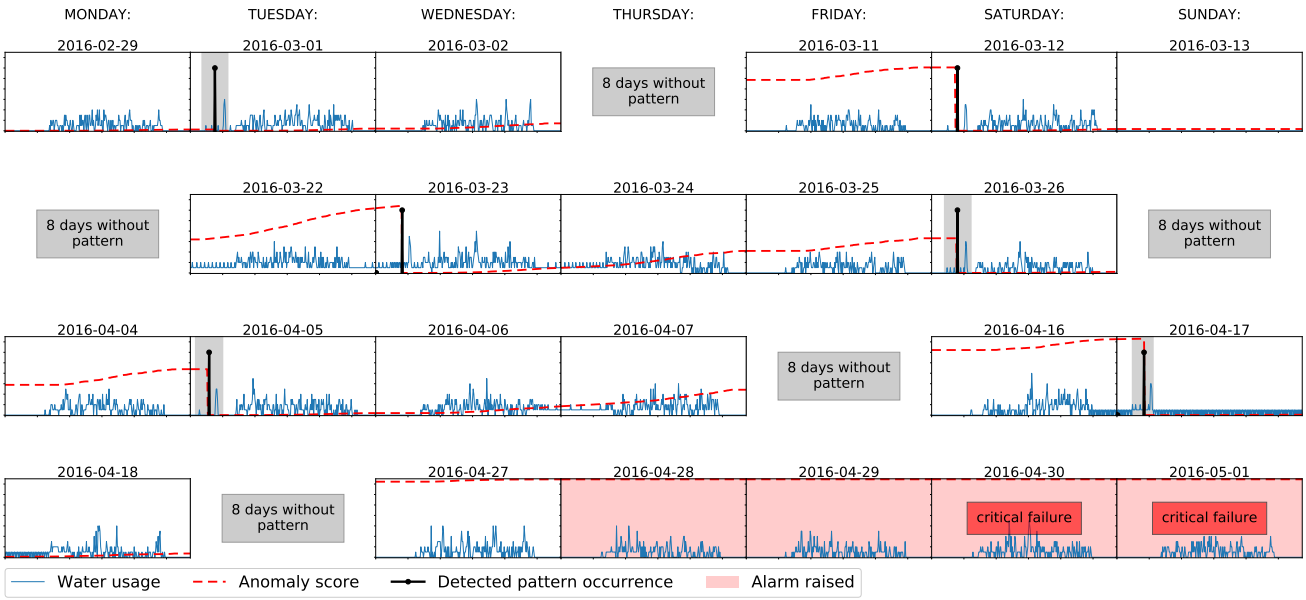

Figure 2: Water usage in one retail store over a period of three months in 2016. The store has an automatically-operated water softener installed to control the water quality. Its self-cleaning operation results in a signature pattern that can be observed in the data. The highlighted segments show all days with such a pattern. The softener works fine until April 17 and we observe the signature pattern on that day. The following two weeks, however, the softener fails to perform self-cleaning and the pattern is not observed. Finally, the system breaks down 12 days after the last self-cleaning operation. We apply FZapPa to detect the absent pattern occurrence between the April 18 and May 1. The black spikes indicate where the PU-classifier has detected an occurrence of the pattern. The dashed, red curve shows the anomaly score that is computed by FZapPa at each time step. Two days before the critical failure, the anomaly score is at its maximal value and FZapPa issues a warning for an imminent failure. Note that immediately after a pattern occurrence is detected in the data, FZapPa's anomaly score drops to 0, only to increase again during the periods the pattern is absent.

of pattern occurrences within T. Finally, at test time, FZapPa uses the trained context and the Weibull distribution to assign an anomaly score to the current segment, which can be communicated to the user or thresholded to generate a discrete prediction.

2.1 Detecting pattern occurrences Detecting occurrences of patterns in the data is important because:

1. At training time, we need to identify all occurrences of the pattern in the training data in order to accurately model its distribution in the data.
2. At test time, it is paramount to recognize each pattern occurrence to avoid raising a false alarm, i.e., reporting an absent pattern when the pattern was in fact present.

However, the pattern's relative rarity presents two complications. One, automatically discovering which pattern in the data is relevant would be challenging as the relevant pattern does not correspond to traditional measures of interestingness such as frequency. Two, even given examples of the pattern, its rarity, its subtle variations and its similarity to other data means that the standard approach of detecting an occurrence by computing the current segment's distance to a prototypical examplar of the pattern is likely to produce false detections.

To address these problems, we view pattern detection as a PU learning problem [2]. The goal is train a classifier to predict whether or not a given segment of the time series contains the pattern. We assume that an expert has provided a small number of segments that are known to contain the pattern, which can be viewed as positive examples. The labeled positive patterns are assumed to adhere to the selected completely at random assumption, which is the standard assumption made in PU learning [2]. However, for all other segments it is unknown whether the pattern is present. That is, each one may or may not contain the pattern, which gives rise to the PU setting.

For our pattern detector, we use the inductive bagging SVM classifier [16], which is designed for PU data. We construct training data by generating $n - \hat{m}$

time series segments from T with $\hat{m}$ the average length of the known patterns. We convert each segment into a feature vector containing the three types of features:

- **Summary statistics:** max, min, mean, median, standard deviation, skewness, kurtosis, and entropy.
- **Time-based features:** two cyclical features that indicate the time-of-day: $\sin(2\pi t_i/24)$ and $\cos(2\pi t_i/24)$, where t_i is the hour in which the segment starts. This ensures for example that a segment at 1am is equidistant to both a segment at 11pm and a segment at 3am.
- **Shape features:** the known pattern occurrences provide valuable information. Therefore, we construct one feature for each known pattern occurence whose value is the segment's Dynamic Time Warping (DTW) distance to the pattern. This captures the segment's similarity to the known occurrences while allowing a certain amount of deviation [3].

A segment containing a known pattern occurrence is labeled as a positive example, the remaining segments are unlabeled. Importantly, the PU classifier cannot be used for performing anomaly detection, it can merely predict if the pattern is present in a segment. Simply because a pattern is not detected in a segment does not mean that the segment is anomalous: the vast majority of segments are not expected to contain the pattern.

The learned model makes a prediction for each segment in the training data, and our final set of detected pattern occurrences consists of all segments that are predicted to belong to the positive class. The use of a sliding window results in the classifier predicting a pattern occurrence in multiple subsequent segments of T. We require the classifier to predict an occurrence in at least q consecutive segments of T to achieve a high confidence in a pattern detection and eliminate spurious matchings. The value of q is set in a data-driven way to be the maximum possible value at which all known pattern occurrences are still detected.

2.2 Modeling the pattern's occurrences Predicting when a pattern is absent requires learning how the pattern occurrences are distributed within T. FZapPa attempts to learn a context based on (indirect) properties of T that is predictive of when the pattern will occur in the data. Building the model proceeds in two steps. First, it searches for a novel descriptor, or feature, X of T that describes the context when the pattern appears. We can rewrite the right-hand side of Eq. 2.1:

$$P(absent_i|X_i = x_i) \text{ with } x_i = f(S_{i,l}, \ldots, S_{j,l}, \ldots, S_{k,l}).$$

The descriptor X_i is defined by a function f that has as inputs the previous segments and as output the value x_i. This function can be instantiated as directly observing a value (e.g., the first measurement $x_i = v_i$ or time stamp $x_i = t_i$) or a constructed descriptor. In this work we consider descriptors that can be constructed from the segments since the last occurrence of the pattern in segment $S_{k,l}$ up to the current segment $S_{i,l}$. Specifically, we consider the cumulative sum $x_i = \sum_{k \leq j \leq i} v_j$ and the time interval $x_i = t_i - t_k$. This naturally generalizes to considering all previous pattern occurrences.

Second, for each type of descriptor X_i, we take the values x from the segments in which the pattern occurs and fit a single Weibull probability density function on the descriptive descriptor values x:

$$f(x; \lambda, k) = \begin{cases} \frac{k}{\lambda}\left(\frac{x}{\lambda}\right)^{k-1} e^{-(x/\lambda)^k} & x \geq 0 \\ 0 & x < 0 \end{cases}$$

where $\lambda > 0$ is the shape parameter of the distribution and k is the scale parameter. The parameters λ and k are determined by minimizing the negative log-likelihood over all x values. The Weibull distribution has a number of useful properties for modeling the distribution of the pattern [17]. It offers a flexible shape parameter that can represent different types of behavior (e.g., both decreasing and increasing occurrence rates over x), and it takes into account that negative values cannot occur. These properties allow modeling of a variety of life behaviors, which makes it also a popular distribution for survival analysis and failure analysis.

Finally, we select the descriptor X^* with the corresponding density function that results in the best fit. Note that in contrast to other popular uses of the Weibull distribution, in our approach the variable X^* is not necessarily time.

2.3 Making predictions At test time, FZapPa uses the learned bagged SVM PU classifier to predict whether the current segment ending at time point t_i contains the pattern. If it does, its anomaly score is zero. Otherwise, it uses the learned Weibull distribution and descriptor to compute the probability that an occurrence of the pattern should have been observed by now, which can be converted into an anomaly score.

Given the best-fitting $f(x^*; \lambda, k)$ and the corresponding best descriptor X^*, computing the anomaly score works as follows. First, for a given time point t_i, we find its corresponding time series interval as the interval between t_i and the timestamp of the last known pattern occurrence. Second, we compute the descriptive value for this interval x_i^*. Third, we compute the probability of observing the descriptive value under the best-fitting model $P(X^* < x_i^*) = F(x_i^*; \lambda, k)$, where

$F(x^*; \lambda, k)$ is the cumulative distribution function for the Weibull distribution:

$$F(x^*; \lambda, k) = 1 - e^{-(x^*/\lambda)^k}.$$

Intuitively, this captures how unusual it is to observe x_i^* under the fitted model. The periods characterized by a high probability are the periods where our model suspects a pattern occurrence. Note that, as long as no new pattern occurrence is observed, $P(X^* < x_i^*)$ can only increase over time because of the cumulative nature of the descriptors. This is desirable in practice as it corresponds to a model that gradually becomes more confident that a pattern is absent. Now, this probability can be used to compute Eq. 2.1:

$$P(absent_i | X_i^* = x_i^*) = 1 - P(X^* < x_i^*).$$

In Figure 2, the dashed red line shows the anomaly score after fitting the model on the earlier-detected pattern occurrences. The descriptor selected by the model in this case is the cumulative water usage. During the final days of April, the anomaly score surpasses the threshold and an alarm goes off, two days before the critical failure.

3 Related work

3.1 Anomaly detection Traditionally, anomaly detection is treated as an unsupervised learning task. Empirically [6, 10], three anomaly detection techniques tend to perform the best: LOF [4], IFOREST [15], and KNNO [18]. Recently, semi-supervised anomaly detection methods that make use of a limited number of labeled examples have become more prevelant [11, 21]. Although successful in many settings, the described anomaly detection techniques are not particularly well-suited to the absent pattern setting because when the (infrequent) pattern is absent, typically (frequent) normal behavior is observed in its place.

Finding absent patterns can be thought of as identifying contextual anomalies, because the absence of a pattern depends on contextual factors (e.g., the time passed since the previous occurrence). Thus, one approach would be to rely on the user hand-crafting the context, but this is often not realistic.

3.2 Time series anomaly detection Most standard anomaly detection techniques can be applied to a time series by sliding a window over the series and computing features for each window. However, a number of dedicated time series approaches also exist.

The first type are residuals models [7], which use regression techniques to forecast the time series data. An anomaly is detected when the difference between the observed and predicted time series value exceeds a threshold. The second type detects deviating temporal patterns as anomalous. HOTSAX and its faster variant WAT [5] compute a discord score as the distance of a time series segment to the nearest non-self matching neighbor. WCAD [12] finds the most deviating segment according to a compression-based distance metric. A recent improvement is MATRIXPROFILE [22], which is an *all-pairs-similarity-search* technique for time series. To detect anomalies with MATRIXPROFILE, one selects the segments with the lowest similarity to the rest of the time series data. This method subsumes both HOTSAX and WAT. Both classes of techniques suffer from the same issues as the standard anomaly detection techniques: they only detect anomalous behavior that differs from typical or common behaviors.

3.3 Arrival time modeling One technique akin to time series anomaly detection detects deviations in arrival times or frequency of arrival [9]. These methods are often used for intrusion detection or traffic analysis where events occur very frequently. Counting or frequency-based methods fail to detect small or moderate changes in behavior without long accumulation periods. A better approach is to look at the arrival times of patterns. The main assumption is that the time since the last event may be irregular, but it is not unbounded. An anomaly is then reported some time period beyond what is reasonable to expect for the next pattern. FZAPPA is inspired by this idea but generalizes beyond interval times to learn the appropriate context that is predictive of when to expect a pattern. Additionally, they take discrete events as input and cannot handle continuous time series directly.

3.4 Health monitoring Prognostics involves predicting the time progression of a specific failure mode from its incipience to the time of component failure, thus the remaining useful lifetime [19]. Our approach relates to this line of research since the absence of a pattern might be an indication of or lead to a failure. We can differentiate between three types of models and data: (1) similarity models require run-to-failure history, (2) degradation models require known failure thresholds, and (3) survival models require life time data [13, 19]. All these approaches assume that the observed behavior can be directly associated with degradation and thus can be used as an indication of the time until failure. Furthermore they require a large number of labeled examples to learn from. Both assumptions do not hold in our setting.

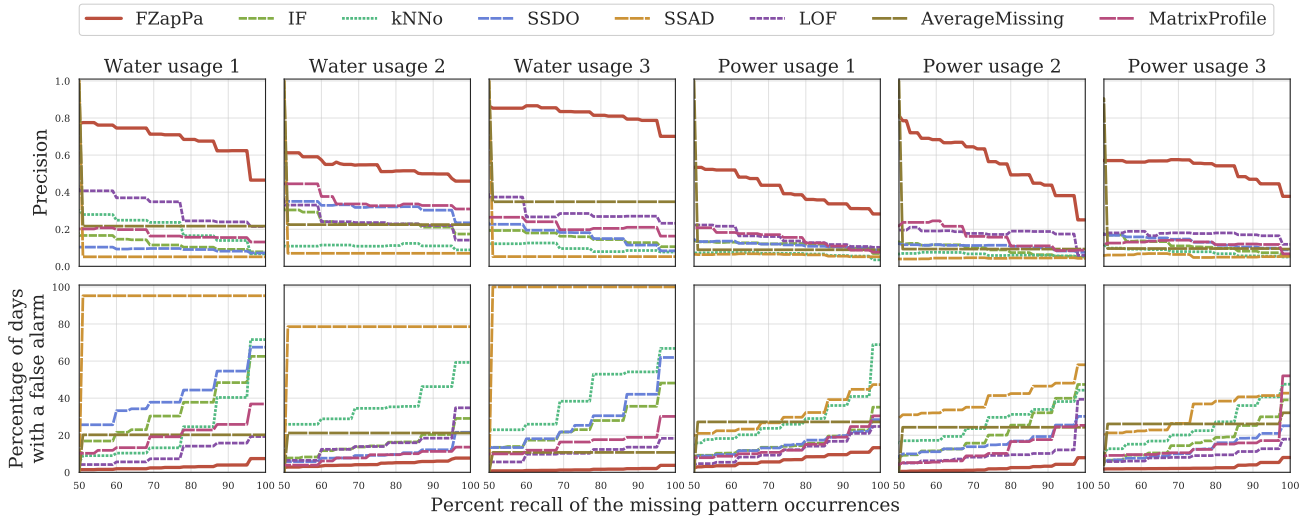

Figure 3: Comparison of FZapPa with the baselines at the absent pattern detection task. The plots show the precision (TOP) and the percentage of days in the test set that trigger a false alarm (BOTTOM) as a function of recall. On each dataset, FZapPa achieves a substantially higher precision than all of its competitors regardless of the level of recall. Similarly, it has the fewest number of false alarms at each level of recall.

4 Experiments

Our empirical evaluation focusses on three questions. First (**Q1**), how does FZapPa compare against existing anomaly detection techniques at detecting absent pattern occurrences? Second (**Q2**), can FZapPa automatically determine the appropriate context (i.e., the best description X^*) that is predictive of the pattern's occurrence? Third (**Q3**), how does using our PU-learning based pattern detector affect FZapPa's performance compared to using standard pattern detectors?

4.1 Experimental setup

Methods. We experimentally compare eight state-of-the-art approaches, divided into three categories:[2]

Unsupervised anomaly detectors. We consider three state-of-the-art anomaly detection techniques: the distance-based Knno [18], the density-based Lof [4], and the isolation-based iForest [15]. We also consider MatrixProfile, a state-of-the-art time series anomaly detection technique [22].

Semi-supervised anomaly detectors. Ssad[3] is a semi-supervised extension of the one-class support vector machine algorithm for anomaly detection [11] and Ssdo[4] updates a prior unsupervised anomaly score by propagating label information [21].

Absent pattern detectors. AverageMissing is a baseline that predicts absent pattern occurrences based on the average interarrival time between occurrences [9]. AverageMissing only works in our setting if we allow it to use the pattern detection subroutine of FZapPa, because it was originally intended to work with discrete event streams [9]. FZapPa is our proposed method.

Resource usage data. We use six real-world datasets, three water usage datasets obtained from a company and three power usage datasets from the UCI repository.[2] Each water dataset records water usage in a retail store, measured every five minutes, over the course of two years. Each store has a water softener whose self-cleaning operation results in a recognizable usage pattern (see Figure 2). A failure of this self-cleaning action is not noticeable in the data, except the absence of the pattern it causes, but it leads to a degradation of the system later in time.

Each power usage dataset records the power usage of a French household, recorded every minute, over the course of three years.[5] We inject a recurring pattern in the data based on a randomly selected shape from the *ItalyPowerDemand* UCR dataset.[6] Each pattern occurrence is a slightly altered version of this shape scaled to a length of around 2 hours. Two versions of each dataset are constructed: one where the occurrences

[2] Implementations and online appendix: `https://github.com/Vincent-Vercruyssen/absent_pattern_detection`

[3] `https://github.com/nicococo/tilitools`

[4] `https://github.com/Vincent-Vercruyssen/anomatools`

[5] Data available at: `https://archive.ics.uci.edu/ml/datasets/individual+household+electric+power+consumption`

[6] Data available at: `https://www.cs.ucr.edu/~eamonn/time_series_data_2018/`

are distributed in T based on time, and one where the distribution is based on cumulative usage. In the experiments, results are averaged.

Experimental setup. The experiment simulates how the absence detection model is used in real-world applications. For each of the six time series datasets T, the following steps are repeated for increasing values of i: (1) extract the first i months from T as training data and month $i+1$ as test data; (2) remove the final pattern occurrence from the test data to simulate an absent pattern; (3) randomly label 10 pattern occurrences in the training data and train the model; (4) detect the absent pattern occurrences in the test data. The first value of i is 6 and the experiment continues until the end of T is reached. One week after the absent pattern occurrence, the test set ends, mimicking an intervention. Each experiment is repeated five times with different initializations of the user-annotated patterns and the results are averaged over all runs.

Evaluation. The goal is to detect each day with an absent pattern occurrence (maximize *recall*) while reducing the number of days with a false alarm (maximize *precision*) given that an alarm requires an intervention and thus bears a physical cost. Each method outputs an anomaly score for each segment of T in the test set. Taking the maximum score over all segments in a day, yields a single score for that day. Thresholding this score results in a discrete prediction: alarm or no alarm. In the experiments, we vary the threshold to observe the precision and false alarm rate under different recall rates. There should be no alarm on the days prior to the day the pattern is absent, and any such alarm is a false positive. A true positive occurs if an alarm is raised within seven days of the absent pattern.

Hyperparameters. Our hyperparameter strategy is designed to give maximal advantage to the baselines over FZapPa. For each of the baselines, we tune the hyperparameters and report the best-achieved results. See the online Appendix 6.2 for full details. For FZapPa, the parameters for the bagging SVM are set to the values recommended in the original paper [16] and the warping width for the DTW-distance is set to 0.1, which is a routinely used value that allows for some deviation between the pattern occurrences [20].

4.2 Q1: absent pattern detection Figure 3 shows the precision (top row plots) and the percentage test set days with a false alarm (bottom) as a function of recall. On each of the six datasets, FZapPa outperforms all the baselines: for any level of recall, it achieves a higher precision. Looking at false alarm rate, averaged over all datasets, at 90% recall, FZapPa issues only 5 ± 2 false alarms per 100 days whereas Lof and MatrixProfile, its two nearest competitors, generate respectively 14 ± 3 and 18 ± 4 false alarms. If one wishes to detect all anomalies, the baselines generate on average 42 ± 21 false alarms per 100 days. FZapPa issues 8 ± 3 false alarms per 100 days. Lof and AverageMissing, its two nearest competitors at this level of recall, issue 26 ± 8 and 23 ± 7 false alarms respectively. Thus, the baselines generate three times as many false alarms as FZapPa and issue a false alarm roughly every three days! Such rates are unacceptable in practice because they create distrust in the detection model. The main reason the baselines perform poorly, is because they are optimized to detect behavior that is different from the normally observed behavior, not normal behavior that is *absent*. Note that the semi-supervised detectors include the labeled pattern occurrences during training. The online appendix discuss the performance of the approaches in more detail.

4.3 Q2: impact of the descriptor selection A key component of FZapPa is its ability to learn the appropriate context that is predictive of the pattern's occurrence. This entails performing the data-driven selection of the descriptor variable in the Weibull distribution. To illustrate the importance of learning the right context, we compare FZapPa with two variants:

FZapPa-Time which is a variant where the descriptor is hand-selected to be the time interval between pattern occurrences).

FZapPa-Usage which is a variant that uses the descriptor is hand-selected to be either cumulative sum of the measured water or power usage since the last detected pattern.

Different descriptors are appropriate for different types of data, necessitating the automatic selection of the descriptor. This is illustrated in Figure 4, which shows for each dataset the percentage of test set days with a false alarm (top row plots) and the precision (bottom) as a function of recall. With the exception of the Water usage 1 dataset, FZapPa always achieves a higher or similar precision than FZapPa-Time and FZapPa-Usage. For Power usage 3 and Water usage 3, *time* is an apt descriptor to capture the distribution of the pattern occurrences, while for Power usage 2 and Water usage 2, *usage* is the better descriptor. In each case, FZapPa figures out the correct descriptor to use.

4.4 Q3: Choice of pattern detector FZapPa internally uses a PU-classifier to detect the pattern occurrences. We empirically compare FZapPa with two variants that use well-known alternative approaches to the pattern detection subroutine of FZapPa. First,

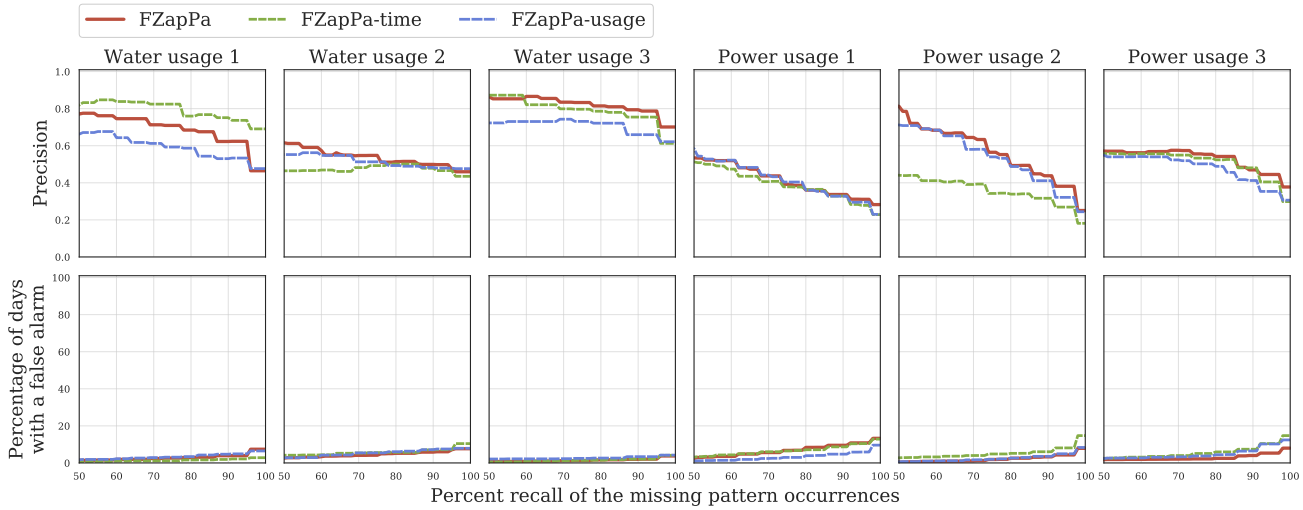

Figure 4: Precision (BOTTOM) and the percentage of test set days with a false alarm (BOTTOM) as a function of recall for FZapPa, FZapPa-Time, and FZapPa-Usage. In 5 out of 6 datasets, FZapPa raises a fewer or similar number of false alarms compared to FZapPa-Time and FZapPa-Usage. FZapPa maintains higher or equivalent precision compared to its variants for recalls $> 95\%$ in 5 of the 6 datasets.

FZapPa-Dtw takes the classic shape-based pattern-matching approach to detection [14]. For a segment, it computes the DTW cost to nearest known example of the pattern. If this cost is below a threshold, then the pattern is detected.[7] Second, FZapPa-Fv looks at the feature-vector representation of the segments that contain a known pattern occurrence [1, 14]. For each segment, it computes the Euclidean distance between its feature vector representation and the feature-vector representation of the closest segment containing a known pattern. Again, if this distance is below a threshold, then the pattern is considered to be detected.[7]

Figure 5 shows for each dataset the percentage of days in the test set with a false alarm (top) and the precision (bottom) as a function of recall for the three methods. Regardless of dataset or metric, FZapPa achieves superior or equivalent performance compared to its variants. The lone exception is on the Water usage 3 dataset where using the DTW detector results in better precision at recalls of less than 70%. These results demonstrate the benefit of our novel PU learning approach to pattern detection in this setting compared to the more traditional approaches. The advantage stems from the PU learner being able to learn an accurate detection threshold.

5 Conclusion

In this paper we introduced the challenging problem of detecting the absence of a sporadically occurring pattern in a real-valued, univariate time series. Notwithstanding its practical applications, this task has not been properly explored within the field of anomaly detection. Moreover, existing anomaly detection techniques are not equipped to detect absent patterns. We designed FZapPa, a novel approach that tackles this problem by first detecting the pattern occurrences and then learning how they are distributed over the time series. FZapPa outperforms a range of general anomaly detection techniques and techniques specifically designed for time series anomaly detection on six resource usage datasets.

Acknowledgements. This work is supported by the KU Leuven Research Fund (C14/17/070) (JD); Research Foundation Flanders under EOS No. 30992574 (JD, WM); VLAIO-SBO grant HYMOP (150033) (JD, WM, VV); *Onderzoeksprogramma Artificiele Intelligentie (AI) Vlaanderen* programme (JD, WM, VV).

References

[1] M. G. Baydogan, G. Runger, and E. Tuv, *A bag-of-features framework to classify time series*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 35 (2013), pp. 2796–2802.

[2] J. Bekker and J. Davis, *Learning from positive and unlabeled data: A survey*, arXiv preprint arXiv:1811.04820, (2018).

[3] D. J. Berndt and J. Clifford, *Using dynamic time warping to find patterns in time series.*, in KDD workshop, vol. 10, Seattle, WA, 1994, pp. 359–370.

[7] The threshold is set to the minimal possible value such that all known pattern occurrences are detected.

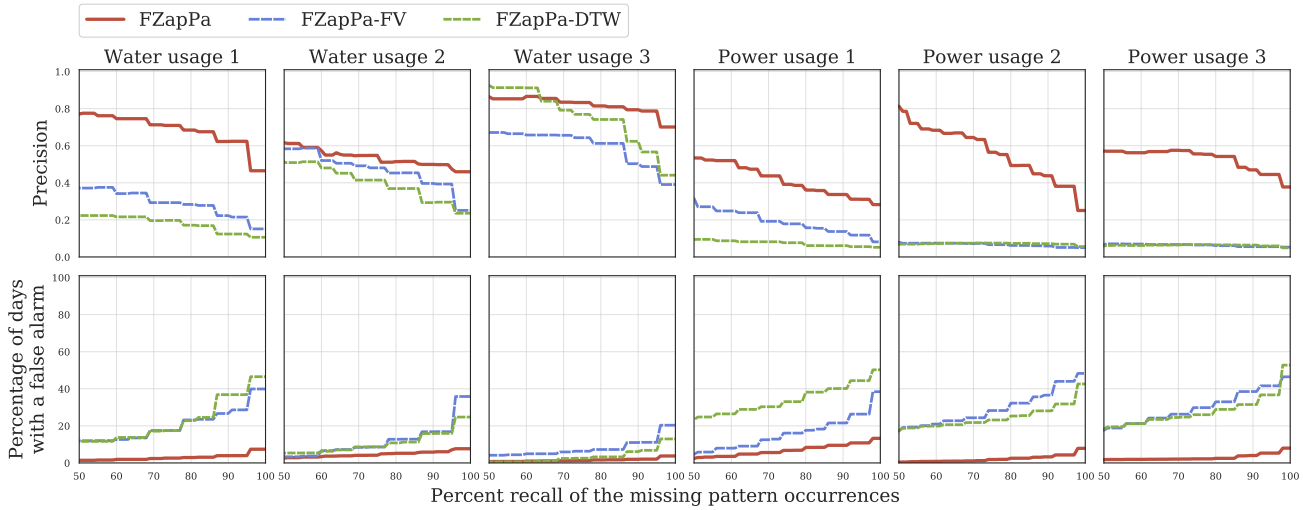

Figure 5: Precision (BOTTOM) and the percentage of test set days with a false alarm (BOTTOM) as a function of recall for FZapPa, FZapPa-Dtw, and FZapPa-Fv. On each dataset, FZapPa raises a fewer or similar number of false alarms compared to FZapPa-Time and FZapPa-Usage. FZapPa maintains higher or equivalent precision compared to its variants for all levels of recall in 5 of the 6 datasets.

[4] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, *LOF: identifying density-based local outliers*, in ACM SIGMOD International Conference on Management of Data, vol. 29, 2000, pp. 93–104.

[5] Y. Bu, T. Leung, A. Fu, E. Keogh, J. Pei, and S. Meshkin, *WAT: Finding Top-K Discords in Time Series Database*, in SIAM International Conference on Data Mining, Apr. 2007, pp. 449–454.

[6] G. O. Campos, A. Zimek, J. Sander, R. J. Campello, B. Micenková, E. Schubert, I. Assent, and M. E. Houle, *On the evaluation of unsupervised outlier detection: measures, datasets, and an empirical study*, Data Mining and Knowledge Discovery, 30 (2016), pp. 891–927.

[7] V. Chandola, A. Banerjee, and V. Kumar, *Anomaly detection: A survey*, ACM computing surveys (CSUR), 41 (2009), pp. 1–72.

[8] H. Ding, G. Trajcevski, P. Scheuermann, X. Wang, and E. Keogh, *Querying and mining of time series data: experimental comparison of representations and distance measures*, Proceedings of the VLDB Endowment, 1 (2008), pp. 1542–1552.

[9] T. Dunning and E. Friedman, *Practical machine learning: a new look at anomaly detection*, O'Reilly Media, Inc., 2014.

[10] M. Goldstein and S. Uchida, *A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data*, PloS One, 11 (2016), p. e0152173.

[11] N. Görnitz, M. Kloft, K. Rieck, and U. Brefeld, *Toward supervised anomaly detection*, Journal of Artificial Intelligence Research, 46 (2013), pp. 235–262.

[12] E. Keogh, S. Lonardi, and C. A. Ratanamahatana, *Towards parameter-free data mining*, in ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2004, pp. 206–215.

[13] Y. Lei, N. Li, L. Guo, N. Li, T. Yan, and J. Lin, *Machinery health prognostics: a systematic review from data acquisition to RUL prediction*, Mechanical Systems and Signal Processing, 104 (2018), pp. 799–834.

[14] J. Lines and A. Bagnall, *Time series classification with ensembles of elastic distance measures*, Data Mining and Knowledge Discovery, 29 (2015), pp. 565–592.

[15] F. T. Liu, K. M. Ting, and Z.-H. Zhou, *Isolation forest*, in IEEE International Conference on Data Mining, 2008, pp. 413–422.

[16] F. Mordelet and J.-P. Vert, *A bagging SVM to learn from positive and unlabeled examples*, Pattern Recognition Letters, 37 (2014), pp. 201–209.

[17] G. S. Mudholkar and D. K. Srivastava, *Exponentiated weibull family for analyzing bathtub failure-rate data*, IEEE Transactions on Reliability, 42 (1993), pp. 299–302.

[18] S. Ramaswamy, R. Rastogi, and K. Shim, *Efficient algorithms for mining outliers from large data sets*, ACM SIGMOD Record, 29 (2000), pp. 427–438.

[19] J. Sikorska, M. Hodkiewicz, and L. Ma, *Prognostic modelling options for remaining useful life estimation by industry*, Mechanical Systems and Signal Processing, 25 (2011), pp. 1803 – 1836.

[20] T. Van Craenendonck, W. Meert, S. Dumančić, and H. Blockeel, *Cobras ts: A new approach to semi-supervised clustering of time series*, in International Conference on Discovery Science, Springer, 2018, pp. 179–193.

[21] V. Vercruyssen, W. Meert, G. Verbruggen, K. Maes, R. Bäumer, and J. Davis, *Semi-supervised anomaly detection with an application to water analytics*, in IEEE International Conference on Data Mining, 2018.

[22] C.-C. M. Yeh, Y. Zhu, L. Ulanova, N. Begum,

Y. Ding, H. A. Dau, D. F. Silva, A. Mueen, and E. Keogh, *Matrix profile I: all pairs similarity joins for time series: a unifying view that includes motifs, discords and shapelets*, in IEEE International Conference on Data Mining, 2016, pp. 1317–1322.

6 Appendix

6.1 Dataset details The following paragraphs provide more details about the six resource usage datasets that were used in the experiments, three water usage datasets and three power usage datasets. The power usage datasets and the scripts to construct them are made available online.[8]

Water usage data. The water datasets were obtained from a large retail company that operates hundreds of retail stores. Each dataset details water usage (in cubic meters) in a single retail store over the course of two full years (2015 and 2016). The data are recorded every five minutes with no missing or extreme values. For an excerpt of the data of one particular store, see Figure 2.

In each store, a water softener is installed that controls the water hardness. To prevent malfunctioning, the softener regularly executes a self-cleaning operation, resulting in a signature pattern that can be observed in the water usage data. As can be seen in Figure 2, every occurrence of the pattern is slightly different. The goal of the company is to detect when a self-cleaning operation should have happened, but did not. Store 1, 2, and 3 contain respectively 64, 141, and 129 occurrences of the water softener pattern over the two years. Initially, there are no missing pattern occurrences in the data.

Power usage data. The power usage datasets were obtained from the freely-available *household electric power consumption* UCI dataset. The UCI dataset contains the power usage (in kilowatt-hour) of a French household, recorded every minute, over the course of 47 months (resulting in a total of $2,075,259$ measurements). In our experiments, we only use the first three full years of the recorded data (starting from January 1 2007). About 0.53% of the said data is missing. We replace the missing values with *base noise*, i.e., the power consumption level when no appliances are used in the house. Finally, we subsample the data to obtain a measurement every five minutes.

Each of the three datasets used in the experiments, is derived from the UCI dataset by injecting a sporadically recurring pattern in the data. Each occurrence of the pattern is obtained by randomly selecting an instance from the freely-available *ItalyPowerDemand* UCR time series dataset, and scaling it to a length of approximately 2 hours. As such, each occurrence is slightly different, yet shows a recognizable shape. Note that this is similar to the water usage case.

Each power usage dataset has two versions. In the first version, the pattern occurrences are distributed in the data based on time. In the second version, the pattern's distribution is based on cumulative usage.

6.2 Hyperparameters The hyperparameters of the baselines are set to reflect the best possible result that can be achieved by each baseline on the task at hand.

Knno and Lof both have a single hyperparameter k, the number of neighbors. For each dataset, we run kNNo and LOF with different values of k between 1 and 100. To report the results, we pick the value of k yielding the highest AUROC on the test data. iForest has two hyperparameters, the number of trees in the ensemble and the sample size used for building each tree. The influence of these hyperparameters on the performance is limited [15], thus we use the values recommended in the original paper.

Ssdo has two hyperparameters. First, α controls the weight given to label propagation step over the unsupervised prior and is set to its recommended value 2.3 [21]. Second, k controls how many instances are updated by propagating a single instance's label. For each dataset, we optimize k between 1 and 100, picking the value yielding the highest AUROC. We run Ssad with a Gaussian kernel and $\kappa = 1$, its recommended value [11]. For each dataset, we optimize SSAD's regularization hyperparameters η_u and η_l in the range $\{0.01, 0.1, 1, 10, 100\}$, picking again the value yielding the highest AUROC.

The MatrixProfile has a single hyperparameter w, the window size. Because we are applying standard anomaly detection methods, such as Lof and Ssdo, to time series data, they also require setting an appropriate window size w. Therefore, we manually set w to the correct length (i.e., the length of the missing pattern) for each baseline, except for FZapPa.

AverageMissing uses the pattern detection subroutine of FZapPa to detect the pattern occurrences. Thus, the hyperparameters of that subroutine are set to the same values as those selected for FZapPa. Finally, FZapPa's only hyperparameters pertain to its pattern detection subroutine. First, the hyperparameters the bagging SVM are set to the values recommended in the original paper [16]. Second, the warping width for the DTW-distance is set to 0.1.

6.3 Detailed discussion of Q1 The following paragraphs contain a more detailed discussion of the results

[8] All implementations are available at: `https://github.com/Vincent-Vercruyssen/absent_pattern_detection`

of the main experiment, comparing the baselines and FZapPa on six resource usage datasets.

Discussion of AverageMissing's Performance. At 90% recall, AverageMissing produces on average over all six datasets 22 ± 5 false alarms per 100 days, which increases to 22 ± 7 when the user requires 100% recall. At recall rates $>90\%$, the precision of AverageMissing, averaged over the water and power datasets, is always <0.35 and <0.10 respectively. FZapPa is a substantial improvement over AverageMissing because it both selects the best-fitting descriptor in a data-driven way and models the occurrences with the more appropriate Weibull distribution. Note that AverageMissing, though a straightforward technique, cannot function without relying on the pattern detection subroutine of FZapPa to obtain access to the discrete occurrences.

Discussion of unsupervised anomaly detection algorithms' performance. While Lof is the nearest competitor to FZapPa, the unsupervised techniques perform poorly when detecting absent pattern anomalies as seen in Figure 3. At recall rates $>90\%$, Lof's precision averaged over the water datasets is <0.28, and its precision averaged over the power datasets is always <0.19. Similarly, at any recall rate higher than 90%, Knno's average precisions are <0.14 and <0.06 for the water and power datasets, and iForest's average precisions are <0.22 and <0.11. These low precisions lead to high false alarm rates. At recall of 90%, Lof generates an average of 14 ± 3 false alarms per 100 days. This rises to 26 ± 8 if 100% recall is desired. For Knno, these averages are 41 ± 7 and 60 ± 11 respectively (Knno issues a false alarm every other day!). iForest issues 30 ± 10 false alarms at 90% recall and 44 ± 11 at 100% recall.

The explanation for the poor performance of these methods is two-fold. First, the absent pattern occurrences are not automatically replaced by *other* anomalous behavior. Instead, the most frequent behavior, the pattern's absence is what is observed. This contradicts the underlying assumption of these methods that anomalies occur infrequently. Second, the traditional techniques have no internal mechanism to deal with the fact that the pattern's occurrence depends on its previous occurrence. The only way to model this type of dependency is if expert knowledge is available to handcraft relevant features.

Discussion of semi-supervised anomaly detection algorithms' performance. To achieve a recall of 90%, Ssdo raises an average of 28 ± 15 false alarms per 100 days over all datasets, which increases to 40 ± 18 if 100% recall is desired. Similarly, Ssad issues 67 ± 26 and 70 ± 22 false alarms per 100 days for these two recall levels. At recall rates $> 90\%$, Ssdo's average precisions are always < 0.30 and < 0.11 averaged over the water and power datasets respectively. Similarly, Ssad precision values remain < 0.08 and < 0.06. Thus, like the traditional unsupervised anomaly detection methods, the semi-supervised anomaly detection techniques also show poor performance on the task of detecting absent patterns. These techniques use the known pattern occurrences as examples of normal behavior when training. However, this does not mean that the resulting models can accurately detect an absent occurrence of the pattern because it is most likely replaced by frequently occurring normal behavior. Ssad in particular has trouble learning a good model when the provided labels are only of one class (i.e., the known occurrences).

Discussion of MatrixProfile's performance. As can be seen in Figure 3, for recall rates $> 90\%$, MatrixProfile achieves average precisions <0.33 on the water data and <0.13 for the power data. On average, this results in 18 ± 4 and 31 ± 12 false alarms per 100 days at respectively 90% and 100% recall. This makes its performance substantially worse than FZapPa and similar to the performances of both Lof and AverageMissing. MatrixProfile suffers from the aforementioned shortcomings of the traditional anomaly detection techniques as it is similar to applying Knno with the Euclidean distance function and $k = 1$ directly to the time series segments of T. The problem is compounded by a number of additional constraints on the usefulness of the MatrixProfile technique for anomaly detection. First, if T contains two or more anomalies that are more similar than at least one random pair of normal segments they will not be identified as anomalies. To see why this is the case, consider that the MatrixProfile computes the anomaly score for each time series segment as the Euclidean distance to its nearest neighbor in T. Second, MatrixProfile is computed only with respect to the raw time series signal assuming that all the necessary information to detect anomalies is contained within the signal. Anomalies characterized by a combination of other features (e.g., context features such as time-of-day) will not be identified [21]. Third, the MatrixProfile only computes the Euclidean distance. It does not work with the DTW distance which has been found to be useful for discovering information in time series data [8].