Detecting anomalies from streaming time series using matrix profile and shapelets learning

Mohammad Alshaer, Sandra Garcia Rodriguez

Detecting Anomalies from Streaming Time Series using Matrix Profile and Shapelets Learning

1st Mohammad ALSHAER
CEA, LIST, Data Analysis and Systems Intelligence Laboratory
Paris, France
mohammad.alshaer@cea.fr

2nd Sandra GARCIA-RODRIGUEZ
CEA, LIST, Data Analysis and Systems Intelligence Laboratory
Paris, France
sandra.garciarodriguez@cea.fr

3rd Cedric GOUY-PAILLER
CEA, LIST, Data Analysis and Systems Intelligence Laboratory
Paris, France
cedric.gouy-pailler@cea.fr

***Abstract*—Detecting anomalies in streaming time series data with no prior labels is considered a challenging issue, especially, when anomalies may vary with time. There is a need to deal with time series streams by identifying the anomalous patterns. These patterns can be described by representative features extracted from the data, which expresses abnormal behavior. This work addresses the challenge of performing online and continuous learning over time series data. In this paper, a solution based on the Matrix Profile algorithm and representation learning approach is developed. In light of that, we will show how the integration of these widely used approaches in the streaming context is quite important for learning and detecting anomalies in realtime.**

***Index Terms*—anomaly detection, matrix profile, shapelets learning, time series analysis, streaming time series, continuous learning**

I. Introduction

Time Series (TS) is a sequence of data records that can be composed of one or more dimensions, such as the sensor data for recording observations. This type of data is widely used in medicine such as electrocardiogram (ECG) data, smart cities, GPS or accelerometer data in activity recognition, etc. Detecting anomalies from unsupervised time series data in the streaming context is very challenging since it requires gathering the experiences from different domains including time series analytics [1], realtime analytics [2], and representation learning [3]. In this paper, we tried to benefit the most out of the previously mentioned domains to come up with a feasible solution that will help the medical doctors identifying heart's abnormal behavior. This identification is based upon monitoring the ECG data collected from a wearable sensor placed on the chest of the patient [4]. For each patient, we build a personalized data set out of the temporal sequence of measurements collected from his heart beats. While monitoring these measurements, any sign which looks like a deviation from other records (anomalies) should be checked to identify whether the patient is developing a first sign of arrhythmia or other probable heart failures [5].

Time series algorithms usually rely on the statistical features extracted from time series, such as mean or standard deviation of sub-sequences, which are assumed to represent the global characteristics of time series [6]. In [6] the authors mentioned that these statistical features represent very superficial information that does not lead to high accuracy in terms of dealing with time series anomaly detection. The Shapelet approach was adopted over the past years by researchers due to its high discriminative feature and good interpretability. Its major limitation is the high computational cost in extracting the shapelets out of the data set. It is known that even for small datasets, the algorithm can take days to extract the shapelets [7]. The reason behind that is the repeated similarity search between a set of subsequences (discriminant shapelets) and the time series instances in the dataset. This high computational cost to extract the shapelets and perform the similarity search is tackled in this paper using the Matrix Profile algorithm. The algorithm is used to find, through fast calculation (*Fast Fourier transform*), the part of the time series that looks anomalous.

Our work will address the previously mentioned challenges for extracting the shapelets based on Matrix Profile algorithm to detect and learn anomalies in realtime. Thus, we will be dealing with unsupervised sensor data measurements that are arriving in realtime. We focus on scalability and distribution in terms of processing and analysis while ensuring that no record is discarded or missed. Our contribution can be summarized by the following points:

1) We propose a novel solution for learning the shapelets that represent anomalies over unsupervised data streams.
2) Based on the learned shapelets (features), we do the detection of the anomalies by performing a similarity function based on sliding window.
3) Convert our approach into a real world application by using STREAMER, a proposed framework for streaming and analyzing sensor measurements in realtime. This paper will not focus on the framework itself but on the implementation and testing of our solution in it.

The remainder of this article is organized as follows: Section 2 defines the background and provides an overview of the state of the art approaches which are related to the problem we are tackling. The proposed solution for extracting anomalies based on the Shapelet approach and Matrix Profile algorithm on the time series data streams will be mentioned in detail in section 3. Section 4 states and gives an overview of STREAMER, used to get scalable, fault-tolerance and optimized execution of the

This research was supported by DATAIA convergence institute as part of the project *Open Platform for Research and Integration of Massive Time Series Stream Analytics* "StreamOps".

model in streaming manner. Then in section 5 we evaluate our approach, discuss the results and the applicability of our proposed solution for anomaly detection. Finally, we conclude the work and give our perspectives for future work in Section 6.

II. Related Work

Considering the fact that data generated by the same type of data source, any deviation from the normal behavior, in general, is known as *anomalies* [8]. Assuming that we are dealing with time series data acquired from the same data source, anomalies are rarely to appear and their characteristics may vary. This results in an exhaustive space to gather them all. Besides, in most real-world scenarios, it cannot be assumed to have a priori knowledge over which data samples are anomalous. That is the reason behind REMIX, a modular framework for automated outlier exploration and detection in an interactive setting [9]. Although REMIX shows a good performance, the modules architecture and time complexity is complex enough as for implementing streaming applications. Dealing with anomalies in time series data is more challenging since abnormalities may appear in very short subsequences as well as on a longer duration basis [10]. In general, two different scenarios for anomaly detection in time series can be identified: 1) the entire time series is anomalous; 2) the time series contains short subsequences or even points which must be marked as anomalies. In the first scenario, we identify the entire time series as anomalous in relation to other time series [11] [12] [13]. In the second scenario, we identify specific short subsequences or points within the entire time series as anomalies [14] [15] [16] [17]. Our work will address the second scenario in which we can detect precisely the subsequences that are considered as anomalous.

Detecting anomalies in time series data is not a trivial task, it requires certain approaches such as the approach of [13] which extracts basic representation features, such as mean or trend. These features are calculated from the entire time series. DILOF [18] is another approach that addresses high memory consumption and provides a solution for detecting a long sequence of outliers. However its time complexity is high. Both approaches may not be adequate in detecting the subsequences that look anomalous, but only working on the time series as a whole set(the first scenario referred above). The authors of [19] addressed the previously mentioned challenges by proposing a deep neural network-based framework called RAMODO to learn representations of outlier detection. It shows the ability to extract representative features from complex and high-dimensional data. However, such approaches require having a huge amount of data which is considered paramount to avoid overfitting, which is not always available. Moreover, many deep learning solutions are sensitive to hyperparameters that can be challenging and time-consuming to tune and optimize for best performances.

To be able to overcome these challenges, we use a different representation of features known as "Shapelet approach", introduced by [20]. In their study, they applied this approach over supervised time series by taking into account the most interesting shapelets as features to represent certain classes. Their way of calculating the information gain of each shapelet candidate for classification in a decision tree requires complex computation and a huge amount of time. Researchers focused on optimizing the extraction of these features, as it presents the work of Lines et al. [21] and another work based on it [3]. According to [22], the learned Shapelets using the previous methods need to be present in hand before classification, which is not our case.

When time series streams analysis must be performed in realtime, fast algorithms for allocating anomalies are needed. Matrix Profile algorithm [23] is the proper approach to deal with this issue. [24] describes it as exact, simple and parameter-free, space-efficient of time complexity $\mathcal{O}(n)$, independent of subsequence length, and with a deterministic time. Matrix Profile is based on Mueen's ultra-fast Algorithm (MASS) for similarity search which plays the main role in computing the Distance Profile based on Fast Fourier Transform (FFT) [25]. Matrix Profile is the most appropriate algorithm for dealing with time series because, besides there is much research work tackled to speedup the computation through converting the data to lower-dimensions representation such as PAA [26] or SAX [27], their methods are still very complicated in terms of having many parameters to adjust. Matrix Profile gathers metadata of the computed time series allocating the positions of the frequent patterns (motifs) and the unique ones (discords). These discords will help to identify the anomalies and speeding up the shapelet extraction procedure. Streaming context requires the capability of handling the high-speed time series data to learn new anomalies and detect the old ones in the minimum time possible. Therefore, thanks to Matrix Profile, the shapelets will be allocated in the fastest and most precise way.

A. Problem Definition

Here we present the multiple definitions and notations that are used in this paper:

Definition 1: A *Time Series* T is a sequence of data records. In this paper, we will consider uni-dimensional time series data. let $T = (t_1, t_2, ..., t_n)$, where n is the length of T.

Definition 2: A *subsequence* $T_{k,m}$ of Time Series T of length n is a subset of records of T of length m starting from position k. $T_{k,m} = (t_{k+1}, ...t_{k+m})$ where $k > 0$ and $(k + m) < (n + 1)$.

Definition 3: *Time Series Stream* S is an unbounded infinite number of time series data records. While dealing with Time Series Stream, windowing mechanism should be utilized in order to handle the subsequences as independent time series. These short time series ease the calculation of similarity search algorithm. $S = (t_1, t_2, ..., t_n, ...)$ where n is an unbounded number and in each window a time series subsequence can be formulated as $T_{k,m} = (t_{k+1}, ...t_{k+m})$ of S for every $k > 0$ and $(k + m) < (n + 1)$.

Definition 4: *Shapelet* ŝ is a $T_{k,m}$ which is used to represent the anomaly in the S. It represents a distinguishable feature

that can be used to detect anomalies from non-anomalies. Besides, it facilitates the interpretability and traceability for achieving better representation learning characteristic in the algorithm.

Definition 5: *Euclidean Distance* (ED) between two subsequences $T_{a,m}$ and $T_{b,m}$ is expressed by the following Equation: $ED(T_{a,m}, T_{b,m}) = \sqrt{\sum_{i=1}^{m}(t_{a+i} - t_{b+i})^2}$ where t is a record of T of index i.

Definition 6: A time series *motif* is the most similar subsequence pair of a time series T of length n. It represents the usual behaviour of time series T. Fromally, we consider $T_{a,m}$ and $T_{b,m}$ is the motif pair iff $ED(T_{a,m}, T_{b,m}) \leq ED(T_{i,m}, T_{j,m}) \forall i, j \in [1, ..., n - m + 1]$, where $a \neq b$ and $i \neq j$.

Definition 7: A time series *discord* is the subsequence of T that is appearing uniquely. It represents the unusual behaviour of time series T. Formally, we consider $T_{k,m}$ as the time series discord iff $ED(T_{k,m}, T_{i,m}) \geq ED(T_{i,m}, T_{j,m}) \forall i, j \in [1, ..., n - m + 1]$, where $k \neq i$ and $i \neq j$.

Definition 8: Distance Profile DP_k^m is a vector that stores the minimum Euclidean Distance between a given subsequence/shapelet $T_{k,m}$ in source T and all the other subsequences $T'_{p,m}$ of target T'. As a remark, T' can be the same as T (self-join) or just another time series. The distance profile can be computed efficiently by using certain methods such as MASS [23]. MASS (*Mueen's Algorithm for Similarity Search*) is the fastest algorithm for measuring the distance between two time series due to its convolution-based nature. It is based on the *Fast Fourier Transform*, which only requires time complexity of $O(nlogn)$ for whatever subsequence/shapelet length we have.

Definition 9: Matrix Profile MP^m is a vector of distance between every subsequence $T_{k,m}$ in T and its nearest neighbor $T'_{p,m}$ in target T'. In other words, $MP^m = (MP_1^m, ..., MP_{n-k+1}^m)$, where $MP_i^m = \min(DP_i^m)$, $i \in [0, n - k + 1]$, n is the length of T. Thus, MP^m is a way of representing an overview of the minimum distances between all subsequences in T. Therefore, time series motifs and discords can be allocated immediately after calculating the Matrix Profile.

III. Proposed Solution

The goal is to spot anomalies in time series streams as soon as they are met. This is a non-trivial process in the streaming context where data is coming in unbounded streams fashion. There is some existing work about how to identify the data streams and their challenges, but yet, a standard way to deal with time series streams has not been proposed. Discussion about how to compare the measurements of time series with the anomalous patterns in order to detect these anomalies is still opened. Therefore, we propose a scientific way to develop an algorithm that can deal with this problem from start to end. Our solution is composed of two main phases: (1) continuous learning patterns of anomalies as representative shapelets; and (2) detecting these patterns in realtime. The system overview of our proposed solution is shown in Figure 1.

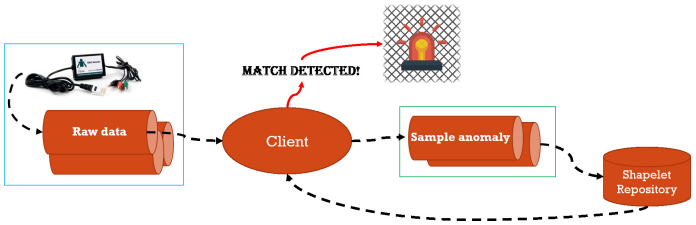

Fig. 1: System overview

A. Continuous Learning of Anomalies

In the streaming context, time series data arrive at a high speed and so we need to update the learning model incrementally based on the newly arriving data. The data is considered unsupervised with no prior labels, which leads to a non-trivial challenge to identify the patterns in the data. Matrix Profile algorithm is the fastest algorithm in allocating the discords (abnormal subsequences that deviate from the motifs) in the data [28]. Our solution will rely on Matrix Profile to allocate these discords in order to learn the anomalies as shapelets. However, these discords are not always anomalies, they can also represent outliers in the time series data. Using outliers can lead to false detection of anomalies. A way of dealing with this confusion is to consider as an anomaly a discord that is met more than once. For this we will firstly distinguish the highest discords as anomalies and deal with the repetition of these discords, and secondly, we will extract the Shapelet of these discords to be used for detection in the next phase. As both learning and detection phases are performed in parallel on the same time series streams, a windowing mechanism of different sizes will be used for each phase. Deploying a larger window for the learning phase will be more appropriate because more data leads to a model with higher accuracy. Within this window, we apply the Matrix Profile algorithm to spot the anomalies and extract their patterns as shapelets. These shapelets are saved in a repository for the next phase. Both learning and detection phases are performed on two independent windows. Due to the fact that anomalies must be detected as fast as possible, detection phase requires a smaller window size.

The main flow of the algorithm is shown in Algorithm 1, in which we show the steps needed for the learning phase. These steps mainly correspond to incrementally learning over a window of Time Series Stream to detect anomalies and extract them as shapelets. They are then stored in a shapelets repository that is used in the detection phase. The highest allocated discords are considered as anomalies since the distance between these discords and the rest of time series is distinguishable. The time complexity of the learning algorithm is $\mathcal{O}(m \log n)$ where m is the number of shapelets learned and n is the learning window's length. The challenging part was how to extract the shapelets out of these discords (in function $extractAnomaliesShapelets(...)$). Shapelets have no restriction over their length (number of points to be included), their size can be dynamic or fixed. The length of the shapelet is

Algorithm 1: Continuous Learning of Anomalies

```
Data: Time series streams
Input: Data, window's length x, shapelet's length l
Result: Storing of Top anomalies as shapelets
learningWindow ← the window containing x
 elements of Time Series Stream
if learningWindow not processed yet then
    dfRaw ← df(learningWindow.data)
      ▷ df is for transforming the data
     to dataframe
    dfMPCalculated ← calculateMatrixProfile(dfRaw)
            ▷ calculateMatrixProfile for
     calculating the MP (Definition 9)
    anomPeeks ← detectAnomalies(dfMPCalculated)
      ▷ detectAnomalies returns the top
     discords (Definition 7)
    dfAnomalies ← extractAnomaliesShapelets(dfRaw,
     dfMPCalculated, anomPeeks)
             ▷ extractAnomaliesShapelets
     returns the shapelets of length l
    storeExtractedShapelets()
       ▷ storeExtractedShapelets stores
     the shapelets in the Shapelet
     repository
end
```

determined by the number of records (points) that represent the subsequence of the shapelet. It is crucial to know the sufficient number of records to detect the anomalies (should it cover the exact anomaly points), taking into consideration that the discord's peak is the start of the anomaly. Note that these shapelets should be compared in the fastest and most accurate way in the detection phase, so it is more convenient to have two vectors of the same length (we will discuss this in detail in the next phase). There is a trade-off when it comes to speed: we can have dynamic length shapelets depending on the anomalies, or cover the main part of the anomaly and still detect it but consuming less time.

B. Detection of Anomalies

Algorithm 2 shows the followed steps. After identifying and storing the shapelets in a repository, the detection phase starts. Since we need to detect the anomalies as fast as possible, we adopt a window equal to the fixed size of the shapelets (significantly smaller than the learning window). The detection is performed immediately after the window is filled with all the required time series records. The Shapelet approach for detection provides high discrimination power, interpretability, and representability that keep the track of all the steps of the model [29]. The Shapelet representability leads to the higher precision of detection and minimizes the space for error or false positive. In order to detect the anomalies, we calculate the distance between the detection window records and the records of the shapelets stored. In order to increase the speed of computing the distance, we vectorize both sets of records and apply the Euclidean distance because of its efficiency. Besides, due to the difference in the measurements, some anomalies may contain a few different points but the same shape, which forces us to adopt a certain threshold for error tolerance. Such a threshold is application dependent. The time complexity of the detection algorithm is $\mathcal{O}(m(n-k))$ where m is the number of shapelets learned, n is the detection window's length, and k is the shapelet's length.

Algorithm 2: Detection of Anomalies

```
Data: Time series streams, Shapelets of Anomalies
Input: Data, window's length y
Result: Notification in case of Matched Anomaly
detectionWindow ← the window containing y elements
 of Time Series Stream
dfAnomlies ← readStoredShapelets()
       ▷ readStoredShapelets reads the
 shapelets from the Shapelet
 repository
MatchedAnomaly ← None
if detectionWindow not processed yet then
    while dfAnomlies not empty do
        dfRaw ← df(detectionWindow.data)
             ▷ df is for transforming the
         data to dataframe
        dfAnomaly ←dfAnomalies.pop()
        vectorRaw ← vectorize(dfRaw)
         ▷ vectorize is used to represent
         the data as a vector
        vectorAnomaly ← vectorize(dfAnomaly)
        distance ← ED(vectorRaw, vectorAnomaly)
              ▷ ED is for calculating the
         euclidean distance (Definition
         5)
        if distance <threshold then
            MatchedAnomaly ← dfAnomaly
        if MatchedAnomaly is not None then
            notifyAnomalyDetected()
             ▷ notifyAnomalyDetected is
             for alerting the detection of
             anomaly
    end
end
```

After being launched, as we are dealing with unsupervised time series streams, the system will face some few seconds of "cold start" meanwhile first shapelets are learned. Unlike cold starts of common clustering algorithms, this waiting time is minimal since learning is performed as soon as the window is full of data. A key point for detecting anomalies accurately is that they should be met more than once. The first time an anomaly appears, it will be learned, and from the second time it will be immediately detected. However, an anomaly that only appears once may probably represent an outlier in the data (false measurement from the sensor).

IV. Scalable and Optimized Execution Framework

The execution environment plays a major role in how the model performs. Thus, considering ECG problem in streaming context, we are dealing with very fast data that may overwhelm the receiver. If these data were not handled properly, the system would eventually run into a memory/buffer bottleneck provoking a lose of most of such data. In order to tackle the streaming nature of the data, we propose a new framework that we call **STREAMER** for continuous learning and detection in realtime. STREAMER provides our model a convenient environment to operate without reaching any bottlenecks.

The framework implements the full realtime analysis chain of ingesting data in streams, realtime processing (continuous learning and detection) and visualization. The framework is composed of 4 independent modules. These modules are connected through services provided by external tools. This degree of decoupling enables to easily replace those components without affecting the rest of the framework. Figure 2 shows the detailed workflow of the framework. It is very important to keep the framework light (with the lower possible complexity) leveraging only the specific functionalities needed out of the existing tools and technologies.

The modules of the **STREAMER** framework are presented as follows:

- M1) **Data Producer:** It is in charge of simulating the data stream sending from $1...N$ sources. Data is ingested from the source and sent to the different messaging channels (also called *topics*). These topics are then accessed by the Stream Processor module to retrieve the data. Each producer simulates one data source. There are two kind of producer types: (i) Offline -data is sent by blocks with a fixed time interval (size of the block and the time interval are configurable parameters); (ii) Online -data is sent according to its time stamp (can be scaled, advanced or delayed in time).
- M2) **Stream Processor:** This module is responsible for managing the workflow of the data through communicating with the rest of the modules using different services. Its mains tasks are collecting the data from the connected topic(s), pre-processing it, launching the update learning model service (to be executed repeatedly every T time), processing the data using the trained model stored, processing the results and visualizing these results.
- M3) **Learning Algorithms:** It offers several online and classical machine learning algorithms for solving different problems. Besides the provided algorithms, the user can add his own algorithms. Algorithms can be coded in any language since in-memory storage repository service is used to information exchange and model storage. This module can be invoked offline (batch training) or online (learning update).
- M4) **Visualization:** It provides as many *html* pages as the number of running sensors or applications (one web page per each streaming workflow). The user can choose between different available graphical interfaces.

Therefore, our proposed solution is implemented in **STREAMER** framework. Most of the framework is based upon scalable technologies that provide a faster and fault-tolerance mechanisms to ensure the proper execution of the model. The Shapelet repository mentioned in the previous section can be in fact represented by the in-memory lookup storage to store the shapelets of the anomalies in the framework. Thus, the updated model (new shapelets) in the detection phase is utilized in the earliest time.

V. Experimentation and Evaluation

Throughout our experiments, we considered the electrocardiogram (ECG) time series data problem. We extensively evaluated our proposed solution in terms of its anomaly detection performance by comparing it with three state-of-art methods: SH-BASE [29], AUTO [30] and RC [31].

A. Experimental Setup

The dataset used is **MIT-BIH database** [32], which contains a set of 48 ambulatory ECG recordings with about 30 minutes each. These recordings were digitized at 360 samples per second per channel with 11-bit resolution over a 10mV range. Every heartbeat is represented by a different sequence in the dataset. These time series are annotated with 5 class labels: *Normal beat* (N), *Supra-ventricular ectopic beat* (S), *Ventricular ectopic beat* (V), *Fusion beat* (F) and *Unclassified Beat* (Q). For comparison purposes with other works, in the evaluation section we will consider the recordings of 8 patients from the data which are: 100, 103, 105, 113, 121, 202, 221 and 228 numbered from 0 to 7.

The metrics used for evaluating the performance of the proposed solution are:

- *Accuracy*: it is the degree to which the results conforms to the correct value. In other words, accuracy means how correctly the data is processed [33]. Higher value means better results.
- *Sensitivity*: is a measure of the proportion of actual positive cases that got predicted as positive (or true positive) [34]. The sensitivity represents how much we predicted positive with respect to the total number of positive records.
  Formally, we can write the accuracy and sensitivity respectively as follows:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

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

  where TP means true positive, TN means true negative, FP means false positive and FN means false negative.

B. Experimental Results

After we presented the model and the framework used throughout our approach, we tested it in a real-world-like scenario for making sure of its applicability and insight-full vision in detecting anomalies and raising warnings. The model

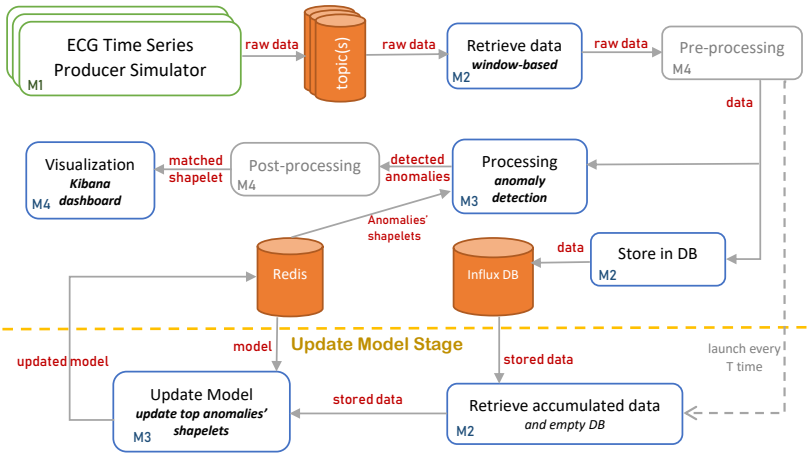

Fig. 2: The Workflow of **STREAMER** Framework

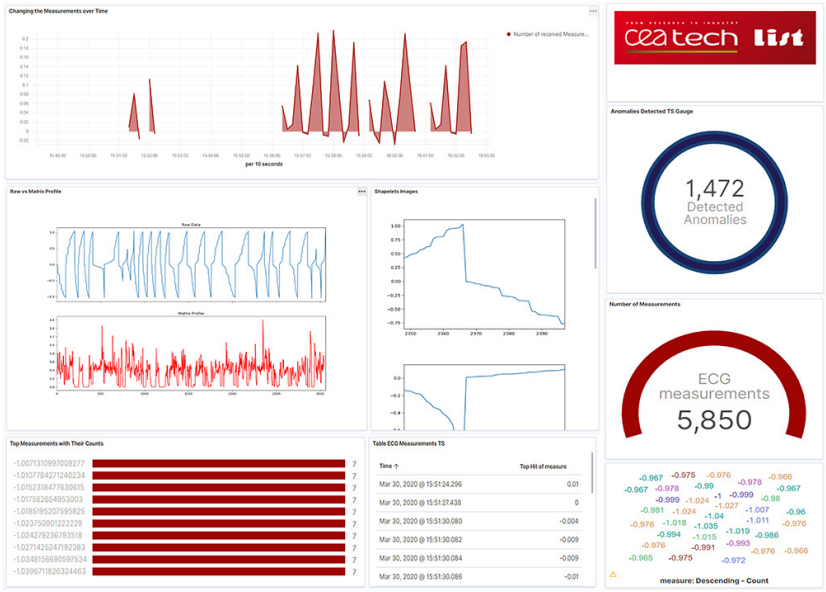

Fig. 3: The Dashboard displaying the results of our TS Anomaly Detection Model at a point of time *t*

was able to highly detecting any anomalous subsequence in the data processed in realtime. As data is ingested into the system, the learning and detection phases start processing these data (as explained in section III). The learning phase extracts the shapelets that were identified as anomalies based on our continuous learning model. The detection phase will be performed before any new learning occurs. Information extracted from raw data, processed data, learned shapelets, and detection of these anomalies is used to build a dashboard. It contains widgets that help the user monitor the execution of all process cycles, with a special focus on having an interpretable and traceable representation learning algorithm as shown in figure 3. It also allows the user to monitor the performance of the model in a transparent way as shown in figure 4.

The proposed solution is of unsupervised nature and so evaluating it is of critical importance. Thus, we conducted an experiment with a labelled dataset in order to measure its performance and compare it with other existing approaches. In this experiment, we consider the *Ventricular ectopic beat* (V) as our V-type anomaly. The benchmark of our proposed solution with the other state of the art approaches is presented in figures 5 and 6. This comparison is based in two metrics already defined in section V-A: *accuracy* and *sensitivity*.

The results presented show that our solution achieved slightly better accuracy in patients 1,3,4 and 6 and an average accuracy in the rest. Due to the fact that the dataset is highly imbalanced in terms of heartbeats categories, normal heartbeats dominate the testing heartbeat collection. For this reason, evaluating the model by only studying its accuracy may not reveal its full performance. Therefore, to get a full reliable evaluation, we also introduce a complementary metric that measures the sensitiveness of the model. Based on the

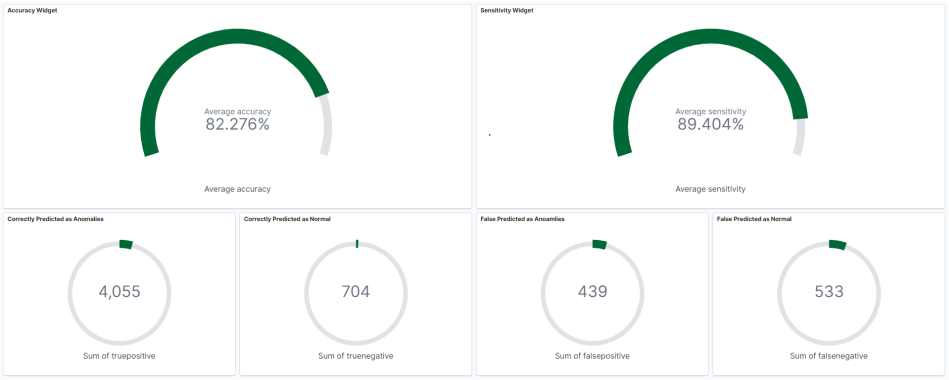

Fig. 4: The Dashboard displaying the *accuracy* and *sensitivity* our TS Anomaly Detection Model at a point of time t

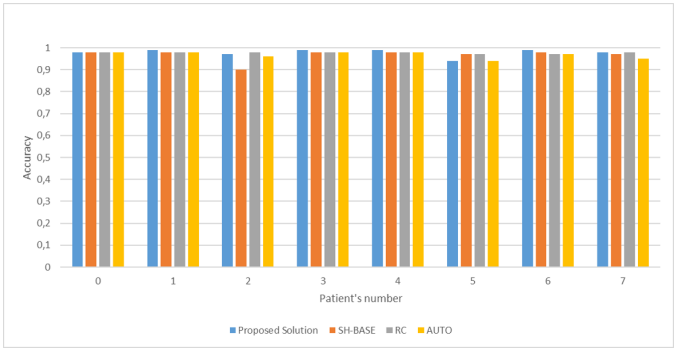

Fig. 5: Evaluation in terms of **accuracy** of our proposed solution with respect to the existing approaches

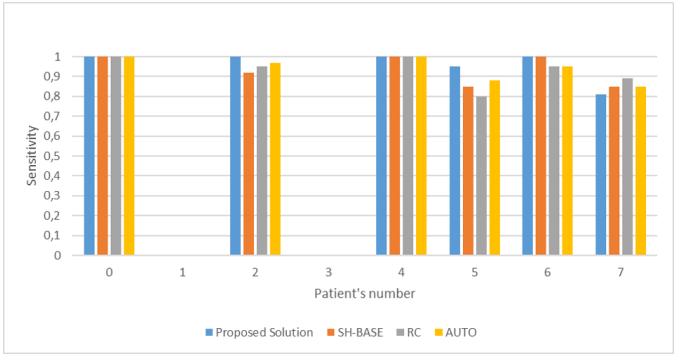

Fig. 6: Evaluation in terms of **sensitivity** of our proposed solution with respect to the existing approaches

identification of the V-type anomaly, the results show that our solution compete the other approaches in patients 2 and 5 in terms of sensitivity. Although it showed a similar or slightly better results, it was slightly behind the other approaches in patient 7. We think it is due to the fact that patient 7 presented, besides (N) and (V), other categories which may have noise and so interfere in the identification of few V-type records.

This evaluation proved how our solution can compete with the existing approaches and even improves them in certain cases. In addition to that, our approach was evaluated in terms of the execution speed by measuring the execution time needed by both phases (training and testing phases) to be completed. This evaluation is shown in figure 7. It was noticed that the execution time for the testing phase is less than half a second which looks suitable for the streaming nature of the problem. Moreover, the execution time of the learning phase, is slightly increasing with time due to the fact that the learning is repeated taking into consideration all the records from start until the moment. Although time increases, it still is well feasible to be applied to the streaming context as it did not exceed the 15 seconds in total for all of the records of the dataset mentioned above. More data will lead to degradation in terms of learning time and that is the reason why we set certain threshold for data freshness performing the learning on the recent N records where N is configurable. Thus, our approach goes one step forward since it is adapted to the real-world context. Following the streaming nature of the problem, our system manages the stream analysis by continuously learning, detecting and evolving (learning new shapelets) ECG anomalies in realtime.

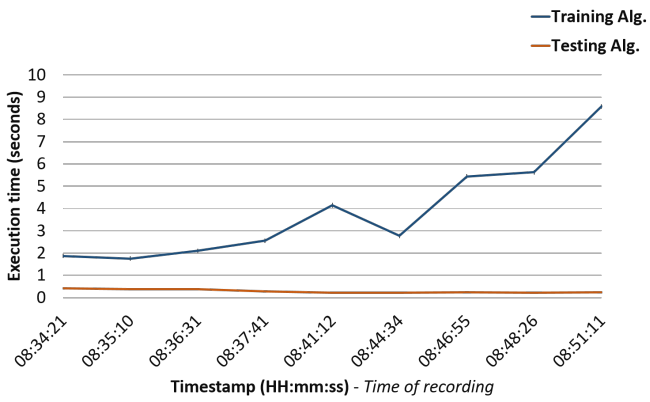

Fig. 7: Evaluation in terms of execution speed of our proposed solution showing the execution time of both algorithms.

VI. Conclusions and Future Work

This paper proposed an unsupervised approach to detect anomalies in streaming time series from the same data source (ECG sensor placed on the chest of the patient). This solution

can be used for a wide range of applications in which unlabeled time series data are being generated. The approach is composed of the anomaly detection algorithm and its streaming implementation within STREAMER framework. The algorithm, which does not require labeled data for training, is able to detect anomalies in a very fast and efficient way. The framework was used to improve the scalability of the anomaly detection algorithm by optimizing the execution environment to handle any fast time series data. The experimentation carried out showed that our approach, based on both the model and the framework, was of major importance for detecting anomalies in streaming time series. Results showed that our solution is competent and improved in some cases the already existing approaches. Future lines of work will focus on the evaluation of our solution by including more patients and considering other datasets. In addition to that, the presented approach will be applied on multivariate time series data. Also, the streaming adaptation of more machine learning algorithms for classification and clustering of streaming time series is pursued. Moreover, setting the adequate N for data freshness will be of interest to get the best results in the minimum time spent.

REFERENCES

[1] G. E. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

[2] S. Ahmad, A. Lavin, S. Purdy, and Z. Agha, "Unsupervised real-time anomaly detection for streaming data," *Neurocomputing*, vol. 262, pp. 134–147, 2017.

[3] J. Grabocka, N. Schilling, M. Wistuba, and L. Schmidt-Thieme, "Learning time-series shapelets," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2014, pp. 392–401.

[4] M. J. Breteler, E. Huizinga, K. van Loon, L. P. Leenen, D. A. Dohmen, C. J. Kalkman, and T. J. Blokhuis, "Reliability of wireless monitoring using a wearable patch sensor in high-risk surgical patients at a step-down unit in the netherlands: a clinical validation study," *BMJ open*, vol. 8, no. 2, p. e020162, 2018.

[5] R. Sassi, S. Cerutti, F. Lombardi, M. Malik, H. V. Huikuri, C.-K. Peng, G. Schmidt, Y. Yamamoto, D. Reviewers:, B. Gorenek *et al.*, "Advances in heart rate variability signal analysis: joint position statement by the e-cardiology esc working group and the european heart rhythm association co-endorsed by the asia pacific heart rhythm society," *Ep Europace*, vol. 17, no. 9, pp. 1341–1353, 2015.

[6] J. Zuo, K. Zeitouni, and Y. Taher, "Exploring interpretable features for large time series with se4tec." in *EDBT*, 2019, pp. 606–609.

[7] L. Ye and E. Keogh, "Time series shapelets: a novel technique that allows accurate, interpretable and fast classification," *Data mining and knowledge discovery*, vol. 22, no. 1-2, pp. 149–182, 2011.

[8] V. Chandola, "Anomaly detection for symbolic sequences and time series data," 2009.

[9] Y. Fu, C. Aggarwal, S. Parthasarathy, D. S. Turaga, and H. Xiong, "Remix: Automated exploration for interactive outlier detection," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2017, pp. 827–835.

[10] S. Forrest, S. A. Hofmeyr, A. Somayaji, and T. A. Longstaff, "A sense of self for unix processes," in *Proceedings 1996 IEEE Symposium on Security and Privacy*. IEEE, 1996, pp. 120–128.

[11] M. V. Mahoney and P. K. Chan, "Trajectory boundary modeling of time series for anomaly detection," Tech. Rep., 2005.

[12] S. Salvador and P. Chan, "Learning states and rules for detecting anomalies in time series," *Applied Intelligence*, vol. 23, no. 3, pp. 241–255, 2005.

[13] R. J. Hyndman, E. Wang, and N. Laptev, "Large-scale unusual time series detection," in *2015 IEEE international conference on data mining workshop (ICDMW)*. IEEE, 2015, pp. 1616–1619.

[14] J. Ma and S. Perkins, "Time-series novelty detection using one-class support vector machines," in *Proceedings of the International Joint Conference on Neural Networks, 2003.*, vol. 3. IEEE, 2003, pp. 1741–1745.

[15] E. Keogh, J. Lin, and A. Fu, "Hot sax: Efficiently finding the most unusual time series subsequence," in *Fifth IEEE International Conference on Data Mining (ICDM'05)*. Ieee, 2005, pp. 8–pp.

[16] A. W.-C. Fu, O. T.-W. Leung, E. Keogh, and J. Lin, "Finding time series discords based on haar transform," in *International Conference on Advanced Data Mining and Applications*. Springer, 2006, pp. 31–41.

[17] C. Chatfield and H. Xing, *The analysis of time series: an introduction with R*. CRC press, 2019.

[18] G. S. Na, D. Kim, and H. Yu, "Dilof: Effective and memory efficient local outlier detection in data streams," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 1993–2002.

[19] G. Pang, L. Cao, L. Chen, and H. Liu, "Learning representations of ultrahigh-dimensional data for random distance-based outlier detection," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 2041–2050.

[20] L. Ye and E. Keogh, "Time series shapelets: a new primitive for data mining," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2009, pp. 947–956.

[21] J. Lines, L. M. Davis, J. Hills, and A. Bagnall, "A shapelet transform for time series classification," in *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2012, pp. 289–297.

[22] L. Beggel, B. X. Kausler, M. Schiegg, M. Pfeiffer, and B. Bischl, "Time series anomaly detection based on shapelet learning," *Computational Statistics*, vol. 34, no. 3, pp. 945–976, 2019.

[23] 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 *2016 IEEE 16th international conference on data mining (ICDM)*. Ieee, 2016, pp. 1317–1322.

[24] C.-C. M. Yeh, "Towards a near universal time series data mining tool: Introducing the matrix profile," *arXiv preprint arXiv:1811.03064*, 2018.

[25] A. Mueen, K. Viswanathan, C. Gupta, and E. Keogh, "The fastest similarity search algorithm for time series subsequences under euclidean distance," *url: www. cs. unm. edu/˜ mueen/FastestSimilaritySearch. html (accessed 24 May, 2016)*, 2015.

[26] W. Luo, H. Tan, H. Mao, and L. M. Ni, "Efficient similarity joins on massive high-dimensional datasets using mapreduce," in *2012 IEEE 13th International Conference on Mobile Data Management*. IEEE, 2012, pp. 1–10.

[27] Y. Ma, X. Meng, and S. Wang, "Parallel similarity joins on massive high-dimensional data using mapreduce," *Concurrency and Computation: Practice and Experience*, vol. 28, no. 1, pp. 166–183, 2016.

[28] C.-C. M. Yeh, Y. Zhu, L. Ulanova, N. Begum, Y. Ding, H. A. Dau, Z. Zimmerman, D. F. Silva, A. Mueen, and E. Keogh, "Time series joins, motifs, discords and shapelets: a unifying view that exploits the matrix profile," *Data Mining and Knowledge Discovery*, vol. 32, no. 1, pp. 83–123, 2018.

[29] H. J. Sun, L., "An extensible framework for ecg anomaly detection in wireless body sensor monitoring systems," *International Journal o Sensor Networks*, vol. 29, no. 2, pp. 101–110, 2019.

[30] P. De Chazal, M. O'Dwyer, and R. B. Reilly, "Automatic classification of heartbeats using ecg morphology and heartbeat interval features," *IEEE transactions on biomedical engineering*, vol. 51, no. 7, pp. 1196–1206, 2004.

[31] M. A. Escalona-Morán, M. C. Soriano, I. Fischer, and C. R. Mirasso, "Electrocardiogram classification using reservoir computing with logistic regression," *IEEE Journal of Biomedical and health Informatics*, vol. 19, no. 3, pp. 892–898, 2014.

[32] G. B. Moody and R. G. Mark, "The impact of the mit-bih arrhythmia database," *IEEE Engineering in Medicine and Biology Magazine*, vol. 20, no. 3, pp. 45–50, 2001.

[33] M. Alshaer, "An efficient framework for processing and analyzing unstructured text to discover delivery delay and optimization of route planning in realtime," Ph.D. dissertation, 2019.

[34] "ML Metrics: sensitivity vs. specificity," https://dzone.com/articles/ml-metrics-sensitivity-vs-specificity-difference, accessed: 2020-04-02.