Seán Enis Cody

Topology Based Anomaly Detection in Time Series Data

Bachelor's Thesis

to achieve the university degree of

Bachelor of Science

Bachelor's degree programme: Computer Science

submitted to

Graz University of Technology

Supervisor

Ass.Prof. Dipl.-Ing. Dr.techn. Roman Kern

Institute for Interactive Systems and Data Science
Head: Univ.-Prof. Dipl.-Inf. Dr. Stefanie Lindstaedt

Graz, August 2020

Affidavit

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present bachelor's thesis.

14.08.2020

Date

Seán Cody

Digital unterschrieben von Seán Cody
Datum: 2020.08.14 15:42:18 +02'00'

Signature

Abstract

Anomaly detection on sequential time series data is a research topic of great relevance with a long standing history of publications. In the context of time series data, anomalies are subsequences of data that differ from the general pattern. Frequently, these specific areas represent the most interesting regions in the data, as they often correspond to the influence of external factors.
Problems which conventional anomaly detection frameworks face are the limitation to highly domain specific applications and the requirement for pre-processing steps in order to function as intended. Through the use of the Recurrence Plot, the algorithm proposed in this thesis, initially seeks to capture the pattern of recurrence found in sequential time series data. An ensuing step for vector quantization by Growing Neural Gas ensures more efficient computation of collective anomalies. Furthermore, the usual preprocessing steps for noise removal are bypassed by the topology preservation aspects the Growing Neural Gas provides. Recurrence Plot construction is done according to a sliding window approach.
The results indicate that both the noise removal by Growing Neural Gas and the pattern preservation by the Recurrence Plot, lead to highly accurate results, with the proposed Anomaly Detector finding all anomalies in a real world data set of Austria's Power Consumption in the year 2017.
Having demonstrated the applicability and potential of combining the Growing Neural Gas with the Recurrence Plot, it seems likely that these concepts could also be adapted to detect further anomalies such as contextual ones.

Contents

List of Figures

1 Introduction

Anomaly Detection in data has experienced ongoing research since the 19th century [2], and judging by the amount of recent papers, holds great relevance to this day.

This paper seeks to further explore anomaly detection for time series data, while utilizing the Recurrence Plot (RP) by Eckmann et al. [5] in combination with the Growing Neural Gas (GNG) algorithm presented by Fritzke [8].

The use of the RP aims to reduce the time series from distinct points in time to distinct state recurrences. The GNG algorithm is then applied on the resulting RP for vector quantization, upon which my approach employs a method to identify anomalous regions in the resulting network. The identification process is based on a sliding window approach, commonly found in the field of computer vision, where it is utilized for object recognition [13]. The goal of this thesis lies in demonstrating the implications the inclusion of the RP and GNG have on the ensuing anomaly detection by sliding window.

The main purpose of the GNG in the proposed approach is vector quantization, allowing for more efficient computation of similarity measures between the sliding windows. Additionally, the algorithm's topology preserving aspects are expected to remove any associated noisy measurements from the data through its Competitive Hebbian Learning (CHL) approach. In contrast to other topology preserving vector quantization methods and variations of the Kohonnen's Self Organizing Map (SOM) [9], the choice of the GNG algorithm lies in its dynamic number of centroids as well as its unsupervised characteristic. This aspect is of utmost importance to my research as the algorithm I wish to develop, must work in an unsupervised environment, so as to grant the anomaly detector (AD) the capability of dealing with unlabelled data. This choice is largely based on the fact that labelled data is generally a scarce resource and tedious to generate.

A further aspect I wish the AD to exhibit is generality. Since anomaly detection of time series data is generally speaking highly domain specific, it is non-trivial to

successfully apply the same AD developed for a certain domain to another field [2]. However through specific selection of my RP parameters, I aim to alleviate this issue and develop an algorithm that can circumvent this problem. The AD's scope would therefore be widened greatly, extending its applicability from a sole domain to multiple domains and with no requirement for domain specific knowledge. I expect my approach and the use of the RP will meet the requirements necessary for it to be considered a general purpose AD.

Under the assumption, that a greater number of recurring states in the time series display normal behaviour opposed to a fewer number of anomalous samples, I suspect this approach could outperform conventional AD systems, such as discord discovery algorithms [4, 10], that seek to identify the most anomalous subsequence within a given sequential data set.

2 Related Work

2.1 Background

Before proceeding with the AD algorithm, I will present definitions of the ensuing tools and methods used. More specifically this chapter will explain the basic concepts of both the RP and the GNG.

Recurrence Plot:
The RP, as developed by Eckmann et al. [5], can be viewed as a way of visualizing the recurrence of states over the course of time [1]. One can express the RP as a mathematical equation as follows:

$$R_{i,j} = \theta(\epsilon_i - \|\vec{x}_i - \vec{x}_j\|), \qquad \vec{x}_i, \vec{x}_j \in \mathbb{R}, \quad i, j = 1, ..., N \tag{2.1}$$

The definition presented in Equation 2.1 is provided by Recurrence Plots and Cross Recurrence Plots [1]. However it has been adapted slightly to better fit the approach proposed in this thesis. Essentially, the formula above describes the necessary steps involved in the creation of the RP. In this context i and j both refer to the locations in the sequential time series data. These markers are used to identify $\vec{x}_i$ and $\vec{x}_j$, which describe the measurements occurring at the i^{th} or respectively j^{th} location in the sequential data set.
In order to create a RP, one must iterate through all possible values for both i and j. The value N in Equation 2.1 indicates the number of measurements contained in the data.
The parameter ϵ_i, known as the threshold distance, serves to define the granularity of what qualifies as a state recurrence, by increasing the chance of the normalized

[1] http://www.recurrence-plot.tk/glance.php (Accessed on: 2020-07-13)

difference of two measurements being greater or equal to zero. For the scope of this thesis, the same ϵ value is assumed for all i data points. The result is then inserted into a heaviside function θ, resulting in all positive values equating to 1, whilst anything else is set to zero. The Heaviside function is a step function, which is defined as shown below in Equation 2.2.

$$\boldsymbol{\theta}(\boldsymbol{x}) = \begin{cases} 1, & x \geq 0 \\ 0, & else \end{cases} \tag{2.2}$$

When the binary output of the Heaviside function θ is 1, this signifies a state recurrence and a dot is placed in the corresponding location of the plot.

Figure 2.1 shows a simple artificially created sinusoidal time series with its corresponding RP for $\epsilon = 1$. The yellow highlights added to the plots, are of width ϵ and illustrate a state recurrence. Therefore, one can clearly see the origin for the variations in sizes of dots marked on the RP.

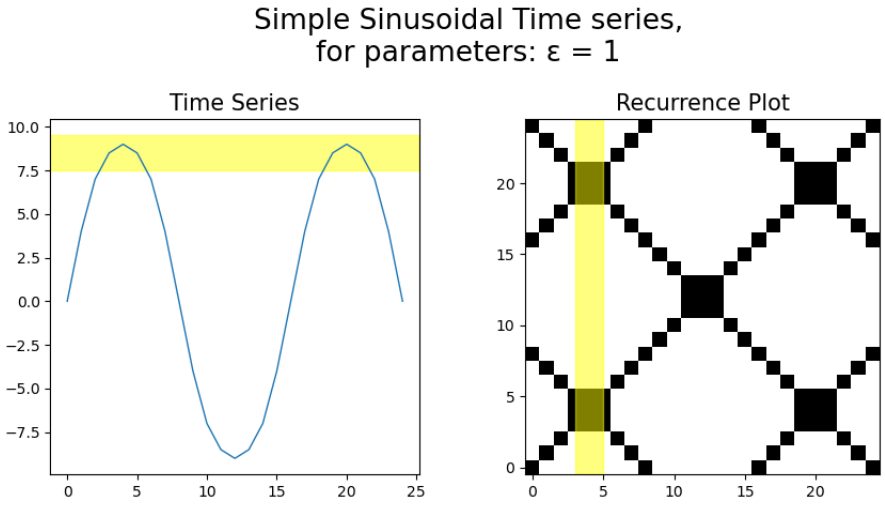


Figure 2.1: A time series and it's corresponding RP.

Growing Neural Gas

The GNG, as developed by Fritzke [8], serves the purpose of vector quantization and topology preservation. GNG achieves its topology representation, through the use of CHL, which classifies it as a Competitive Neural Network. For CHL to be applied, vector quantization must first be carried out. The algorithm uses an error counter for each neuron, which is highest for frequently adapted ones. This allows GNG to determine where more neurons must be inserted to better represent the topology.

The Network consist of two main components, nodes and edges, which have the following attributes:

1. Node - **c**
    - $w_{\mathbf{c}}$
    - $error_{\mathbf{c}}$
2. Edge - **e**
    - $node\_references_{\mathbf{e}}$
    - $age_{\mathbf{e}}$

As listed above, every node is assigned a weight $w_{\mathbf{c}}$, which can be seen as the node's positional argument in 2-dimensional space, as the AD is provided 2-dimensional input in the scope of this paper. Each node is also given an $error_{\mathbf{c}}$ counter, which as described above, helps the network determine the areas which require more detail. Edges have the $node\_references_{\mathbf{e}}$ attribute, which contains the information regarding the two nodes it connects. Additionally edges have an $age_{\mathbf{e}}$, helping to distinguish edges that can be removed.

A GNG implementation receives a number of parameters, which are listed below. Their use will be explained in the ensuing pseudo code for the algorithm.

GNG Parameters:

- $X = \{x_1, x_2, ..., x_N\}$ input data

- ϵ_b scales the movement of the winning node
- ϵ_n scales the movement of its neighbours
- age_{max} maximum age of edges
- λ iterations before node insertion
- d error reduction constant
- α scales error after new node insertion

GNG Pseudo Code:

1. Start with two nodes a and b, with random positions $w_{\mathbf{a}}$ and $w_{\mathbf{b}}$.
2. Draw a sample x from the input data X.
3. Find the closest node s_1 and the second closest node s_2.
4. Increment the age of all of s_1's outgoing edges.
5. Add the squared distance from s_1 to the sample x to its error counter $error_{\mathbf{s1}}$
6. Move s_1 towards sample x by fraction ϵ_b. Move all direct topological neighbours of s_1 towards x by fraction ϵ_n.
7. Create a new edge between s_1 and s_2 or in the case of an edge already connecting them, set that edge's age to 0.
8. Remove all edges that have an age above age_{max}. Additionally any nodes that now have no outgoing edges will be removed.
9. If the number of iterations is a multiple of λ, a new node is inserted. The location of this node is chosen so as to minimize the error accumulated through frequent movement. Therefore the node with the highest error counter is chosen.
   a) Find node q with maximum accumulated error counter $error_q$.
   b) Insert a new node r between q and its topological neighbour f with the highest accumulated error counter $error_f$.

c) Remove the edge connecting q and f. Insert two new edges connecting the new node r to q as well as to f.
d) Multiply $error_q$ and $error_f$ by λ, so as to descrease the error. Then assign the resulting value of $error_q$ to $error_r$.

10. Decrease all error variables by multiplying them with d.

11. If a stopping criterion is not yet fulfilled, continue with step 2.

The proposed AD utilizes an implementation of the above GNG algorithm provided by Neupy[2]. This implementation introduces $min\_dist\_for\_update$ as a stopping criteria for step 11. Additionally it provides the network with a growth stunting mechanism, which caps the number of nodes the network may contain. This parameter is of utmost importance, reducing the probability of overfitting the input data, keeping the number of nodes at the bare minimum necessary for representing the topology. Neupy also provides the option of increasing the number of starting nodes presented in step 1 in the above pseudo code. My AD however did not make use of this extension.

Essentially, the GNG algorithm learns the topology by evaluating the areas that require more node insertions to model, as these are frequently moved, leading to higher error counters, due to many data samples causing signals in that area. The goal is to minimize the cumulative error of all nodes in the network.

2.2 State of the Art

Neural Networks such as Neural Gas (NG) by Martinetz et al. have been shown to aid time series analysis, through successful applications in time series predictions [11]. Since GNG can be viewed as an extension of NG by CHL, one can assume to find similar successes in GNG's topology representations and capabilities on sequential data such as those found in RP's and time series.

NG is an unsupervised learning algorithm designed for vector quantization and clustering of data. The algorithm is based on Kohonen's Self-Organizing Feature

[2] http://neupy.com/pages/home.html (Accessed on: 2020-05-02)

Map (SOM) [9], extending it by removing the constraint of requiring a grid of fixed neurons [6]. Furthermore, NG allows its neurons to move freely and create edges to neurons in their neighbourhoods. GNG expands upon this by using CHL rules to generate topology incrementally, whilst utilizing edge aging schemes to remove invalid edges resulting from the movement of neurons [8]. This removal in particular allows for the successful combination of NG with CHL.

RP, as shown by Eckmann et al., serve as a visualization technique for identifying underlying patterns [5]. These plots are characterized by setting a point in the grid only when a recurrence of a specific state in the state space of the time series occurs. An additional parameter allows specification of the degree of similarity the two distinct states must have, in order for the second state to be classified as a recurrence of the first.

Although the focus of this paper lies in the identification of anomalies in such RP, it is also important to highlight more conventional uses of GNG for outlier identification in time series.

Chandola et al. in their survey on anomaly detection, list a multitude of anomaly detection techniques, of which many can be deemed unsuitable for this paper's goal, which is the pursuit of an unsupervised learning approach, applicable to finding anomlies in sequential time series data [2]. Considering that cluster-based techniques have proven successful, a vector quantization method such as the GNG is likely to perform well. Although this paper's goal is the identification of collective anomalies in a RP, there exist a vast number of papers concerning point anomaly and contextual anomaly detection that successfully utilize GNG, which I would like to highlight. One such solution can be seen in the utilization of Merge Growing Neural Gas (MGNG), which has been shown to successfully identify noise in time series data [15]. MGNG is an extension of GNG that incorporates context as a weighted vector for deciding on the winner neuron [6]. This inclusion of context makes this an ideal technique for sequential data [1].

Further research has been conducted on GNG's capabilities in outlier detection in the field of computer vision. As shown by Waniek et al., the GNG's performance in vector quantization yields good results [17]. In their work, they demonstrated GNG's use by embedding it into their event processing pipeline for real time anomaly detection. The reduced number of points to compare per frame through GNG vector quantization improved overall efficiency. Similar results attesting to GNG's success in identifying outliers can be seen in detections of abnormalities in

changing surveillance scenes [14]. Here Qianru et al. specifically show the benefits GNG has over SOM approaches, most notably being the lack of null neurons, arising due to SOM's use of grids. Qianru et al. explained the related dangers these null neurons could potentially inflict, describing that identification would be harder if they were to perfectly align with anomalies. In my work, knowing this is an added benefit, yet the deciding factor in the choice of GNG was its ability to dynamically choose the number of clusters as opposed to NG and thereby SOM grid [6].

A problem which arises when trying to apply the approaches presented in computer vision to a time series, is the lack of ground truth data. In the above experiments, the anomalies were identified through comparison of the current frame to its previous one. In this thesis' problem specification, this would essentially translate to a comparison of time series where one is non-anomalous.

A potential solution to bypass the need for additional data is provided by Paisner et al. in their paper on symbolic anomaly detection and assessment using GNG [12]. The paper presents a GNG AD technique, that utilizes A-distance for symbolic AD on a time series. A-distance corresponds to a sliding window technique, which provided with a baseline window, compares its values to that of a sliding window, to identify deviations. This baseline for time series AD corresponds to the initial n measurements of the series and not a separate data set. The AD method developed in this thesis draws great inspiration from this approach.
Since most of the previously mentioned approaches however do not directly target the detection of collective anomalies, which will be the focus of this thesis, a few notable ADs for collective anomaly detection within recurrent time sequences must therefore be mentioned. The problem of finding collective anomalies in sequential data is closely related to another research topic. This research area's focus is on finding the solution to the problem of identifying the subsequence within a time series that is least similar to all other subsequences [10]. The process of finding these so called discords is known as discord discovery. According to Lin et al. the trivial approach, known as brute force discord discovery, is both easy to implement and provides exact results. It does however have poor runtime performance. Therefore Mooi Choo Chuah et al. have extended the BFDD to achieve higher accuracy in discerning anomalous regions from normal ones [4]. Their algorithm is known as adaptive window-based discord discovery (AWDD).

Window-based approaches are of great relevance to time series anomaly detection. What sets my method apart, however, is the use of the RP and therefore the reduction

to state space for conventional time series. Through the application of the GNG to the RP and the ensuing noise removal and vector quantization, a novel concept to anomaly detection on a time series, which has not been explored in previous research, is presented. This paper is one of the first to utilize the information contained within the RP for anomaly detection on a time series. Through the combination of the noise removal and vector quantization aspects, GNG offers, and the accuracy provided by window based discord discovery algorithms, the proposed algorithm has the potential to combine all the previously mentioned approaches' benefits.

Since beginnning the development of the proposed AD, further significant research regarding the application of the RP in an anomaly detection context has been published. Utilizing a Convolutional Neural Network (CNN) to analyze the hidden patterns presented in RPs, has been shown to be effective in distinguishing abnormal behaviours in time series data [3]. In this work, Chen et al. extracted RPs from data collected from sensors monitoring electrical current and electrode-position profiles in a flash welding process. The RPs are then used to train the CNNs, leading to very accurate results in anomaly detection of welding quality variations.

While the success of their approach attests to the applicability of the RP in anomaly detection pipelines, there are multiple drawbacks pertaining to the use of CNN in this context, the greatest being the reliance on labelled data. While not explicitly mentioned in their paper, CNN's are predominantly trained in a supervised manner. This ties in closely with the limitations in cross domain usage their algorithm presents. The AD is limited to the detection of anomalies in a certain time series domain and the trained network cannot easily be extended to provide detection on slightly different data sets. This can present a significant problem considering how scarce a resource labelled data is.
My proposed approach to anomaly detection circumvents this issue by relying on unsupervised learning, making the AD applicable in a greater number of scenarios.

3 Method

This chapter will describe the thesis' approach to anomaly detection and explain the concepts surrounding it. Firstly, definitions and assumptions taken throughout the algorithm's development are given. Following this a highly detailed look at the ensuing AD and its concepts is undertaken.

Anomalies in the context of this thesis can be seen as follows:

given a time series T containing K subsequences C as defined below ...

$$\begin{aligned} &\text{Time series T} = \{t_1, t_2, ..., t_m\}, \text{ where } t_i \text{ is the measurement at} \\ &\qquad\qquad\qquad\qquad \text{location } i \text{ in the data set} \quad [10] \\ &\text{Subsequence C} = \{t_p, t_{p+1}, ..., t_{p+n-1}\}, \text{ for } 1 \leq p \leq m - n + 1 \\ &\qquad\qquad\qquad\qquad \text{and } n \leq m \quad [10] \end{aligned} \tag{3.1}$$

... the goal is to find the subsequences that maximize the cummulative difference to all other non-intersecting subsequences of the time series T, in other words anomalies in the context of this thesis constitute to:

$$Anomalies = \underset{C_i, \forall 1 \leq i \leq K}{\arg\max} \sum_{k=1}^{K} diff(C_i, C_k), \tag{3.2}$$

where K is the number of non-intersecting subsequences of length n contained in time series T of length m.

Within this paper's scope, anomalies abide by the definition given by Chandola et al., where: "Anomalies are patterns in data that do not conform to a well defined notion of normal behaviour" [2].

On the topic of normal behaviour, there are various approaches to identify normal behaviour in anomaly detection. A trivial approach is to define a region of data as normal behaviour and search for anything that deviates from it [2]. However, for non-labelled time series data and therefore this thesis, this approach is of little use, as no prior knowledge on normality of the data is assumed. The definition of normal behaviour in sequential data used in this thesis, is based on the assumption that more normal data is present opposed to the amount of anomalous samples.

Out of the three main classes of anomalies - point anomalies, contextual anomalies and collective anomalies - presented in Chandola et al.'s survey "Anomaly Detection: A Survey", [2], the focus of this paper lies in the detection of collective anomalies. Such anomalies in periodic time sequences would correspond to distinct periods which display deviations from the behaviour of a greater number of similar periods. Based on the previous definition of normal behaviour, one can therefore label the fewer periods of the time series data, that display deviating patterns, as anomalous.

For the identification of the above mentioned period of recurrence in time series data, I assume the period's length is known. The process of finding the period, season length estimation, is a large research topic in itself and does not lie within the scope of this paper. In this thesis it is therefore assumed, that the period length has been previously extracted from the data and that it's length does not change.

As defined by Lin et al. the subsequence of a time series that is least similar to all other subsequences is known as a discord [10]. This definition closely relates to this paper's goal, which is the identification of periods in time series data, which differ most from a larger number of periods.

Since labelled data is a scarce resource, algorithms that function in unsupervised learning environments were chosen. The underlying assumption is, that no prior knowledge on the data is known.

Furthermore, my paper's AD utilizes a sliding window procedure, commonly used for object detection in the domain of computer vision [13], in order to identify anomalous behaviour. This technique has been shown to perform quite well in its respective field, which is why I have chosen it to be a part of the anomaly detection pipeline.

The AD should consist of the following steps:

1. Construction of RP
2. Application of GNG
3. Calculation of Similarity
4. Identification of Anomalous Windows

Step 1 and 2 of the above mentioned AD approach offer multiple ways in which one can apply the sliding window procedure. In this paper, three of these approaches were studied more closely.

The first is the application of the sliding window procedure to the result obtained by applying GNG to a RP of the entire data set. The advantage of constructing the RP for the entire data set is the preservance of all possible states. In other words, if one were to construct a RP based solely on a section of the time series, only states occurring in that specific section would be known. An example of how this could cause a problem is shown below in Figure 3.1, where an anomalous period, marked in blue resembles the pattern of a valid state sequence in the corresponding RP.

This issue can lead to false anomaly detection under certain conditions and is best avoided by constructing a RP of the entire time series.
A downside to this approach, however, is the slow runtime resulting from large input data for the GNG. As one certainly requires more nodes to model a more complex topology, this also leads to much greater runtimes. The original GNG algorithm is known for its poor runtime performance on larger data sets, with 48% of the algorithm's runtime being allocated to the search of the nearest neighbour, and 51% to the search for the node of largest error [7]. As Fišer et al. have shown this can be improved greatly through more efficient GNG design. However, since this thesis' choice was to evaluate the implications, which the original GNG algorithm has in an anomaly detection context, an algorithm that can bypass this issue without the need for refactoring procedures was developed.

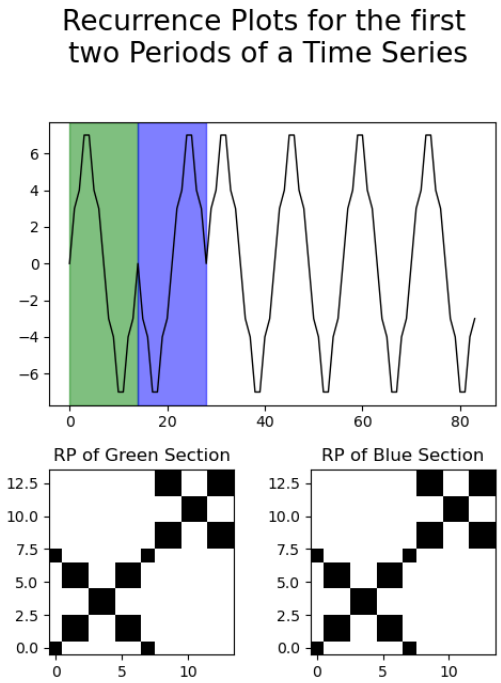


Figure 3.1: Two distinctly different data sequences can have the same RP.

The second approach studied, utilizes the sliding window technique prior to applying the GNG. In this approach, initially, the creation of a RP containing all data samples takes place, from which windows are extracted to provide the GNG with input features. However this resulted in slow runtime performance due to inefficient window construction. The trivial application of the sliding window can lead to a grid approach, splitting a RP into R^2 RPs, where R is the number of windows which fit into the time series. A more sophisticated is the use of symmetry, which allows us to limit the comparison of either the right of left of the RP's diagonal (symmetry is given since ϵ value is the same for each sample). However this algorithm cannot deal with is a non-stationary time series such as the one provided in Figure 3.2. This is a common pitfall in time series anomaly detection and is often alleviated through pre-processing steps. To avoid the reliance on further pre-processing steps, an approach was devised, which can bypass this issue without the need for additional tools.

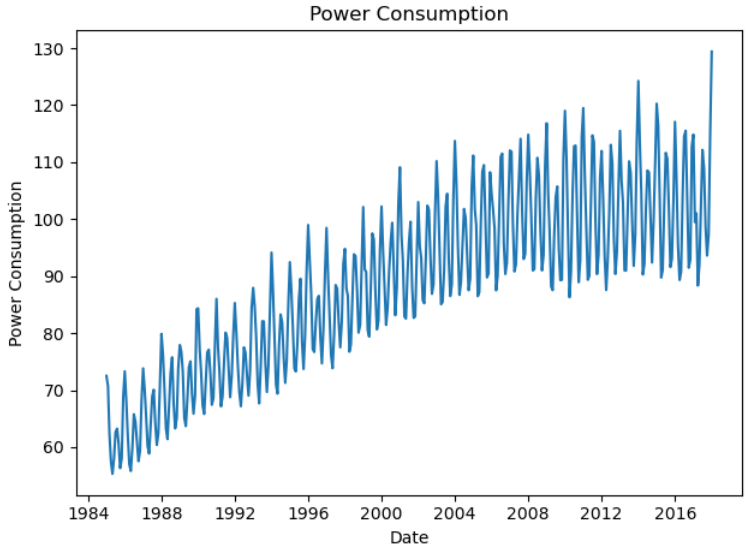


Figure 3.2: A non-stationary Time Series[1]

This leads to the third and best performing method, which is to apply the sliding window technique directly to the time series data, thereby obtaining a partitioning of the data set. The GNG is then applied to the RPs extracted from the window partitions. This approach abuses the missing state information shown in Figure 3.1, while using a low enough step size, such as the value 1, when sliding the window, to avoid the erroneous window remaining undetected. Essentially, this method allows the examination of the relationship between the states within a period and then compares this to other periods' behaviours.

In order to assess the similarity between windows, the average pairwise minimum distance between the nodes of two different windows was chosen as the representative value of the similarity. This was done in both direction, as false similarity can arise if few nodes are compared to many. In this case it is likely that the few nodes will have a short distance to the many nodes, while the overall networks are clearly dissimilar

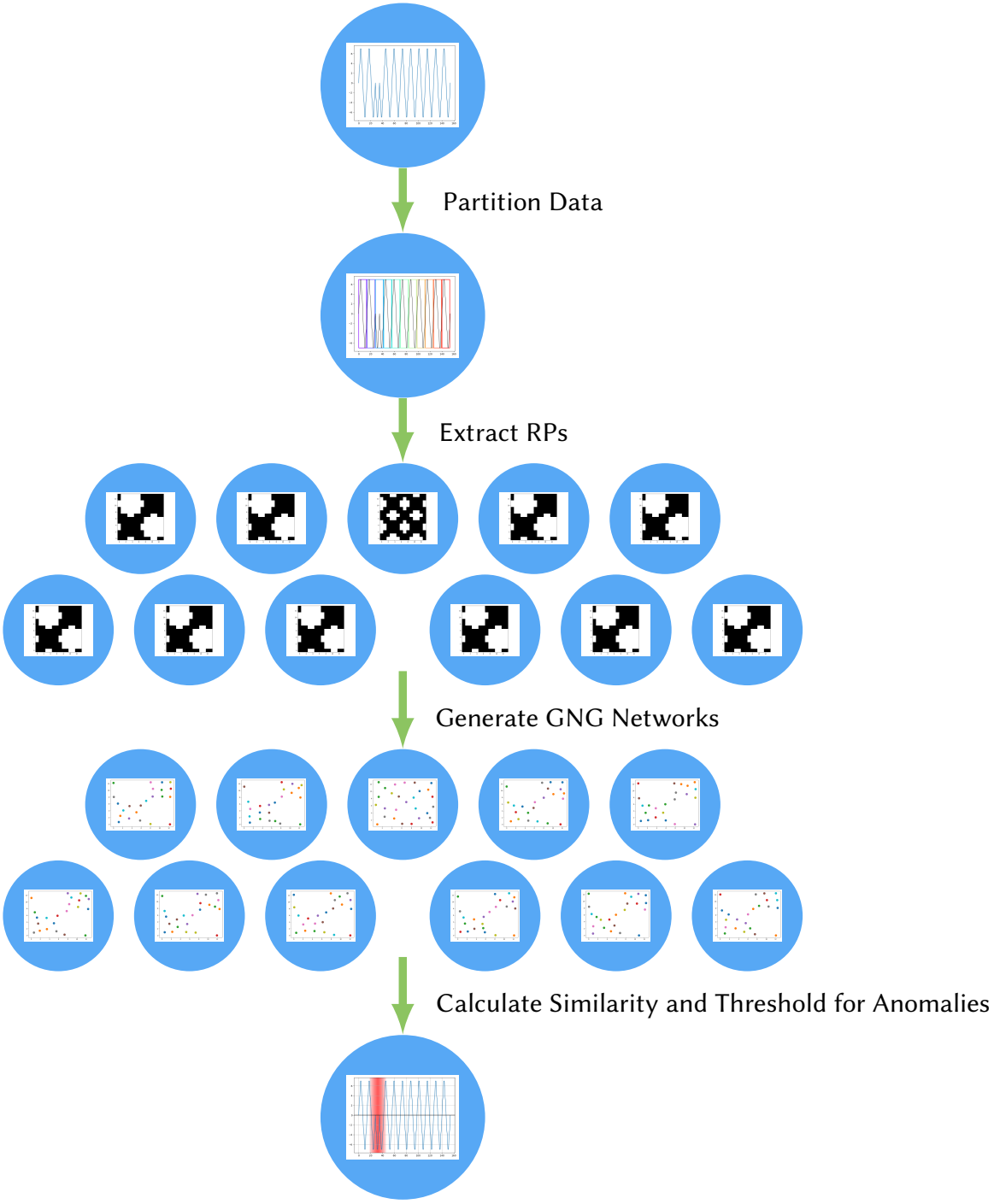
[1] https://towardsdatascience.com/predict-electricity-consumption-using-time-series-analysis-4650284e40aa (Accessed on: 2020-07-11).

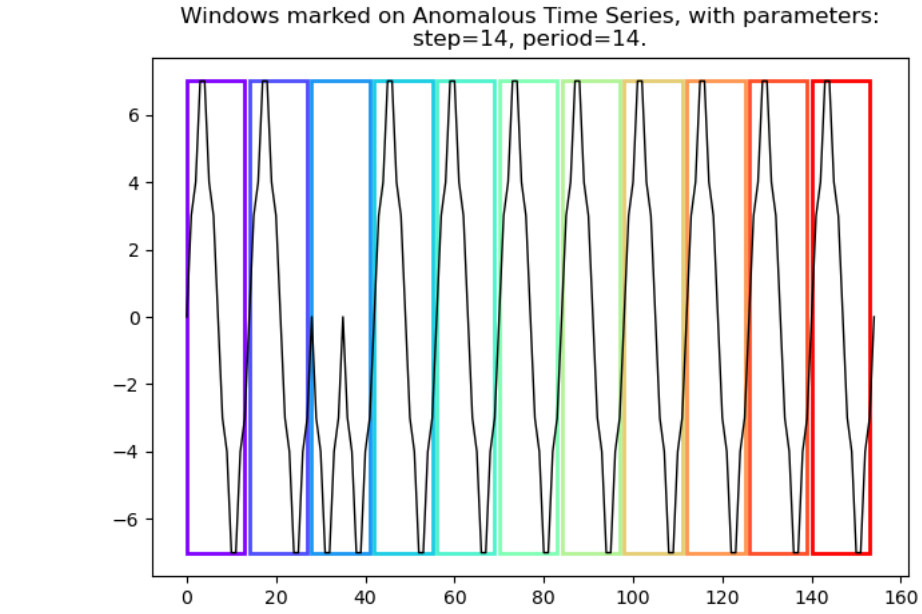
The corresponding pseudo code, Algorithm 1, is listed below, providing more detailed insight into the implementation of the four previously mentioned steps.

An example of the final sliding window technique, the ensuing RP creations and vector quantizations by GNG are shown in Figure 3.3. During the creation of RPs, unless a specific parameter is passed, an automated $epsilon$ choice is taken, in order to allow for 50% state recurrence to be plot. This is done by taking the median of the amplitudes as $epsilon$. The algorithm's steps are illustrated in the following flow chart. Furthermore a short summary of the main characteristics the proposed AD posseses and the assumptions taken throughout this chapter, is provided in Table 3.1.

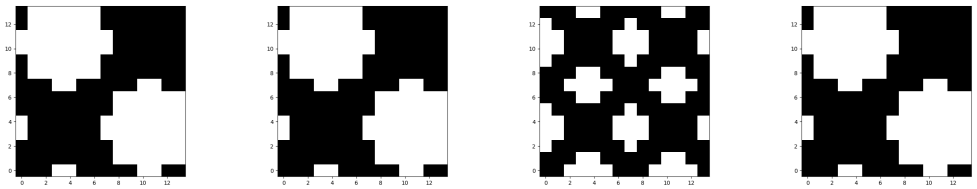
Key Assumptions of the proposed AD
A greater number of normal data must exist opposed to anomalous samples
Period length must be known
Period length must remain constant

Table 3.1: Summary of the key assumptions for successful anomaly detection.

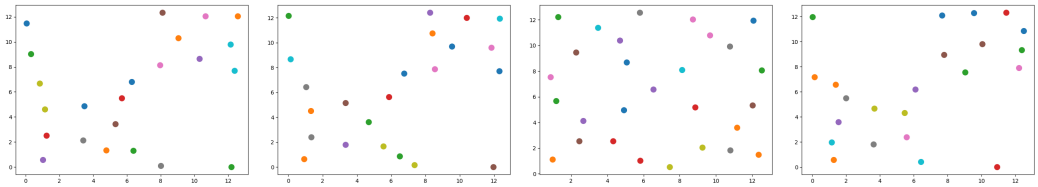

Partition Data
Extract RPs
Generate GNG Networks
Calculate Similarity and Threshold for Anomalies



(a) Grid-based window partitioning.



(b) RP of Window 1 (c) RP of Window 2 (d) RP of Window 3 (e) RP of Window 4



(f) GNG of Window 1's RP (g) GNG of Window 2's RP (h) GNG of Window 3's RP (i) GNG of Window 4's RP

Figure 3.3: RPs and GNG nodes for the first 4 partitions created by the sliding window approach on the data set.

Algorithm 1: Topology-Based Anomaly Detector

Input: time_series_data, period, window_size, step_size, similarity_threshold
Output: Anomalous windows within the data

all_RPs = $empty\_list$

Step 1: Construct RPs:
$window\_start \leftarrow 0$
while *window fits in time series* **do**
    Construct RP from data in range $window\_start$ to $window\_start$ + $window\_size$
    Save RP to all_RPs
    Increment $window\_start$ by $step\_size$
end while

Step 2: Apply GNG:
foreach *rp in all_RPs* **do**
    apply Neupy's [a] GNG implementation with the following parameters to rp:
    $max\_nodes \leftarrow 500, step \leftarrow 0.2, n\_start\_nodes \leftarrow 2,$
    $max\_edge\_age \leftarrow 50, neighbour\_step \leftarrow 0.005, n\_inputs \leftarrow 2,$
    $n\_iter\_before\_neuron\_added \leftarrow 100,$
    $after\_split\_error\_decay\_rate \leftarrow 0.5,$
    $error\_decay\_rate \leftarrow 0.995, min\_distance\_for\_update \leftarrow 0.01,$
    $shuffe\_data \leftarrow True, verbose \leftarrow False$
end foreach

Step 3: Calculate Similarities:
foreach *nodes1 in every GNG network constructed in Step 2* **do**
    foreach *nodes2 in every GNG network constructed in Step 2* **do**
        $similarity \leftarrow 0$
        foreach *node in nodes1* **do**
            $similarity \leftarrow similarity$ + distance from $node$ to closest node in nodes2
        end foreach
    end foreach

[a] http://neupy.com/pages/home.html Version: 0.8.2 (Accessed on: 2020-05-02)

```
    foreach node in nodes2 do
        similarity ← similarity + distance from node to closest node in
          nodes1
    end foreach
    similarity ← similarity/(length(nodes1) + length(nodes2))
    save similarity for both windows
end foreach

Step 4: Find Anomalous Windows:
foreach win1 in GNG window do
    similarity_counter ← 0
    foreach win2 in GNG window do
        if euclidean_distance(win1, win2) is period then
            if similarity(win1, win2) ≤ similarity_threshold then
                increment similarity_counter
            else
                decrement similarity_counter
            end if
        end if
    end foreach
    if similarity_counter < 0 then
        add win1 to anomalous regions to return
    end if
end foreach
```

4 Evaluation

To evaluate the performance of the proposed AD described in chapter 3, two distinct baseline algorithms were chosen, the results of which will be compared to the AD's output. The time series data sets used to extract the results consist of both synthetic and real measurements. The synthetic data sets are used to better target the shortcomings and benefits of the algorithms, while the real data set is used to showcase the ability of my algorithm to function in real world scenarios.

The chosen baseline algorithms for evaluation are (1) a Median Average Deviation (MAD) based approach as the lower end baseline method and (2) the Brute Force Discord Discovery (BFDD), which is a state-of-the-art discord detection algorithm. The second algorithm was chosen in order to develop a better understanding of true applicability in the field.

Lin et al. present in their paper "Approximations to magic: finding unusual medical time series", a method which outperforms BFDD in runtime while the anomaly detection results remain the same [10]. However for the purpose of this thesis BFDD was chosen due to its simplicity in implementation.

In the algorithm proposed in this thesis, the chosen $window\_size$ parameter has a large influence on runtime. This can be seen in Figure 4.1, where multiple different values for $window\_size$ have been chosen and the runtime for the algorithm to finish computing anomalies on the same data is plotted.
Although the exact runtime is of little importance for this thesis, the general trend must be evaluated, in order to identify the applicability of the proposed AD to larger data sets.

Clearly, one can see that keeping the $window\_size$ to a minimum is beneficial to improving the algorithm's runtime. Additionally, the result will be more fine grained, as a smaller window encompasses less valid data points surrounding the anomalous ones. As mentioned previously, the lower bound for $window\_size$ should be the

period, as anything below could result in inaccuracies.

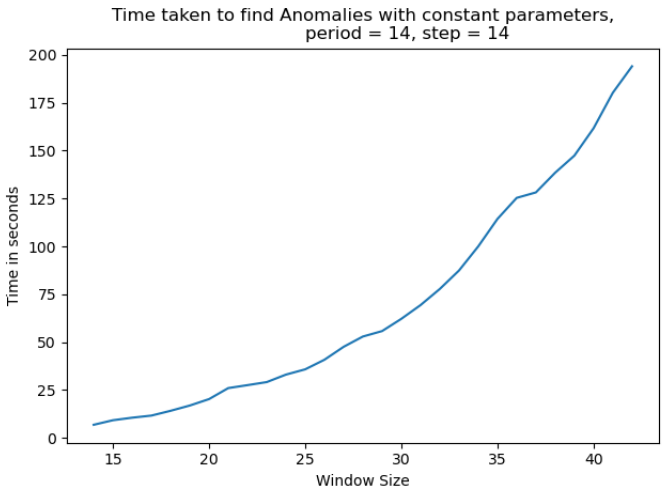


Figure 4.1: Runtime for AD with varying $window\_size$ and constant step size.

It can be argued that the above results are skewed as parts of the data will be evaluated multiple times, due to the $step\_size$ being lower than the $window\_size$. By setting them equal however, the runtime depicts a fluctuating pattern. Figure 4.2, shows these slight fluctuations. The general trend has not changed, although the runtime is greatly reduced. A possible explanation for the frequent drops shown in runtime is as follows: when windows do not fit exactly into the time series, this results in non-computation of portions of the data.
Clearly increasing the $step\_size$ leads to faster results. One must however keep in mind that, as mentioned in chapter 3, smaller $step\_sizes$ are safer for finding anomalous regions.

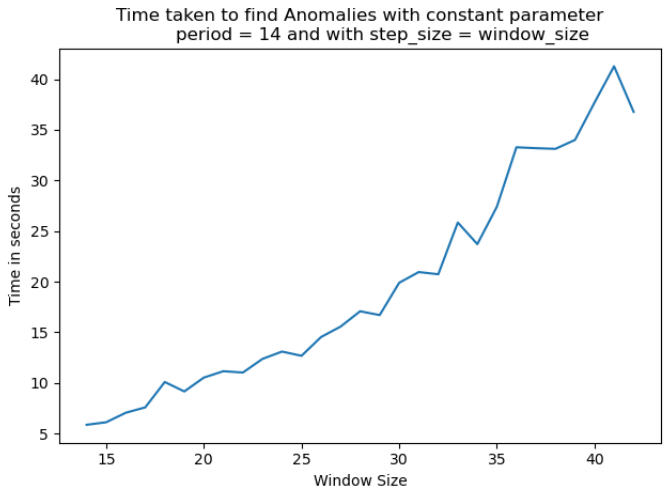
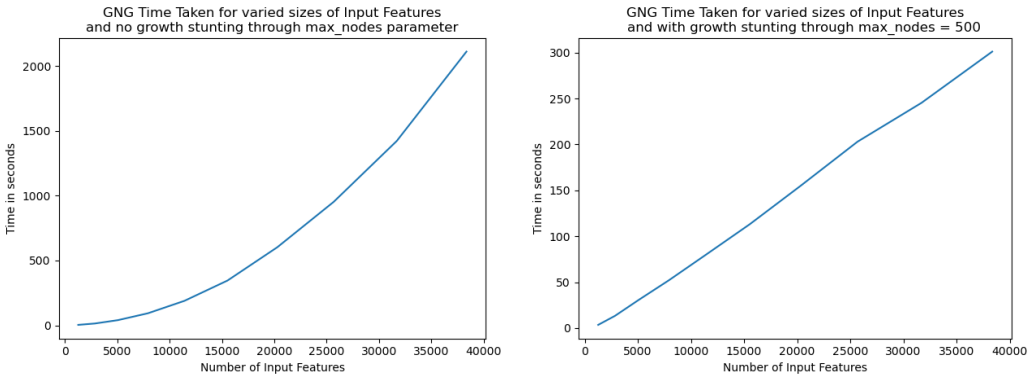


Figure 4.2: Runtime for AD with varying $window\_size$ and equal step size.

Chapter 3 mentioned the inefficiency of the original GNG's runtime. This theory is further supported by Figure 4.3a, where one can see the rapid growth in runtime with increasing input feature sizes. As the behaviour closely resembles that of my algorithm's runtime, a logical conclusion is that the runtime performance is greatly dependent on the execution time of the GNG algorithm. The maximum node count is uncapped in the following plot.



(a) Uncapped number of nodes.

(b) Growth stunting through max_nodes parameter.

Figure 4.3: Effect of growth stunting mechanisms on GNG.

The proposed AD method however has an upper bound for node count, which is set to 500. Figure 4.3b shows a near linear relation between runtime and input feature lengths for GNG. For all GNG experiments, 20 epochs of training were used.

The result supports my choice for introducing growth stunting within the anomaly detction pipeline. A problem that could arise and would require intervention, however would be a scenario where the topology is highly complex and requires a greater number of nodes to model. Under the assumption that a larger RP is more complex in nature and therefore requires more nodes to model, the conclusion is that keeping the $window\_size$ to a minimum is vital for the AD's success.

As previously mentioned, the runtime on big data sets is of relevance to the topology-based AD. Figure 4.4, shows how the algorithm behaves for increased time series lengths. The steep increase in runtime likely stems from the nature of the original GNG as a $window\_size$ of 24 results in a RP of approximately 288 state recurrences. Applying the GNG for 20 epochs and adding nodes after every 100^{th} iteration, results in 59 representative nodes. Therefore, in this specific scenario, the benefits

of equating the $max\_nodes = 500$ cannot be accrewed and the steep increase in GNG runtime depicted in Figure 4.3a occurs.

A possible further explanation for significant runtime incurred could be the similarity measure, which is is trivially implemented leading to $O(n^2)$ runtime. This could be improved greatly by the introduction of geometry based algorithms for finding the closest pair in a data set, such as minpair in the Minimum Pairwise Average Distance. Optimally, a more suitable algorithm could be chosen to evaluate the similarity between windows, such as computer vision's SIFT or similar feature detection methods. These algorithms traditionally extract important features and describe patches of data in images with great success. Since the scenarios are similar, one can assume that the methods could be applied with reasonable success in finding similarities between windows.

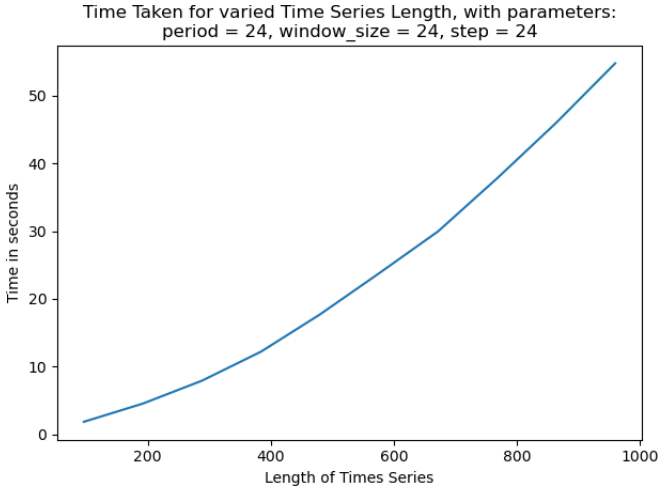


Figure 4.4: Runtime for AD with varying input time series lengths.

Further potential pitfalls of the algorithm to be highlighted are the following.

Firstly, the AD requires an exact period to function properly. If this is not provided, the detection fails to identify anomalies. While this can be alleviated to some extent using pre-processing steps to identify the period of the data, it is not a trivial task and deviations can cause failure of the algorithm.

Secondly, relating back to the choice of $window\_size$, whilst having identified that keeping it as low as possible is a necessity, it must still be large enough to extract a reasonable RP from it. As mentioned in chapter 3, the period can be considered to be the optimal choice, due to the fact that the unfolding of entire state sequences within a given period can be modelled.
Essentially, the $window\_size$ must encompass enough measurements to define the topology. For example, a window size of 2 coupled with the 50 percent heuristic epsilon choice, may have no more than 2 dots in the RP, which lie in the diagonal. This would lead to identical behaviour within all windows. Additionally applying GNG would cause no vector quantization or noise removal. Taking this issue into account, the choice of window size clearly has a lower bound. The best and safest choice for this lower bound is the period itself, as the goal is to compare the periods and anything below that does not represent all states contained within them. On the other hand, some values below the period have been shown to perform reasonably well, see Figure 4.5. The areas marked in red correspond to the anomalous windows detected by the algorithm. A suitable upper bound would certainly be largely dependent on the number of anomalous periods as more normal ones must exist. Essentially, one cannot pick window sizes that cause partitions of the data in ways that contradict the previous assumption of normal behaviour.

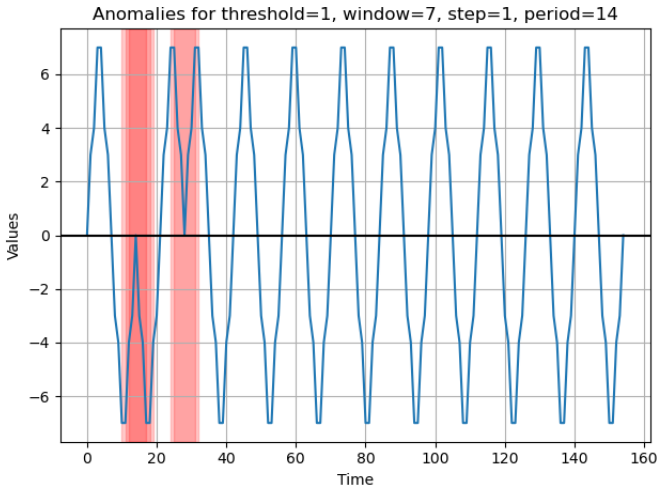


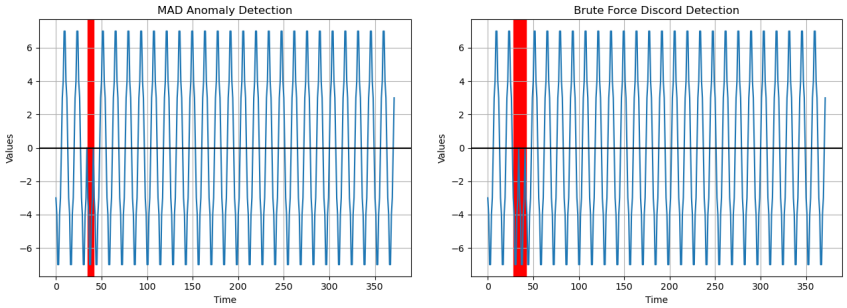
Figure 4.5: Anomaly detection for $window\_size$ below period, shows only a part of the collective anomaly being detected and marked red.

Figure 4.5 does however miss the anomaly found between the two marked regions, due to the nature of the recurrence plot and its epsilon value. The plot indicates that values hovering around amplitude 0 are considered the same and lead to similarities lying below the chosen threshold.

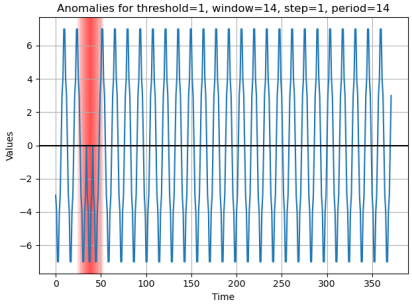
The choice of similarity thresholds leading to good results can be seen as another shortcoming. On the surface, one can solve this through passing a range of thresholds to the AD and manually narrowing down the choice based on the results. This however, becomes more difficult as the period and its $window\_size$ increases, leading to the averaging performed in Minimum Average Pairwise Distance becoming less distinctive. As previously mentioned, a potential solution would be a more refined similarity measure.

Concerning detection performance, the topology-based AD performed very well on all synthetic data sets, outperforming the baseline algorithm utilizing MAD, while producing identical results to the BFDD.

In Figure 4.6 one can observe successful identification of the discord in a simple time series. The plots illustrate that all three methods can achieve desired results on easy to identify anomalies.



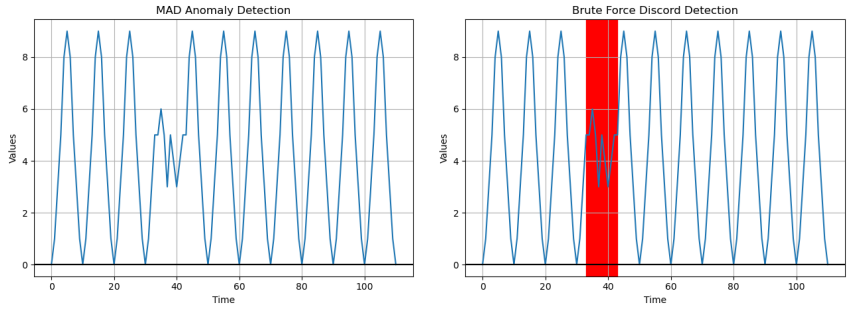
(a) Median Average Deviation (b) Brute Force Discord Discovery



(c) Topology-Based Anomaly Detection

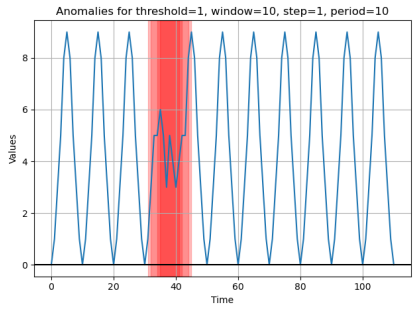
Figure 4.6: The baseline methods and proposed algorithm identify simple anomalies successfully. Anomalous windows are marked red.

The MAD detector however fails on the data set found in Figure 4.7. The underlying reason for this failure is the detector's simple thresholding, where anything deviating from the expectation by more than the MAD value of the data set is considered an anomaly. Clearly, the regions found by the BFDD and topology-based AD methods are anomalous, however not anomalous enough for the MAD-detector to identify.



(a) Median Average Deviation

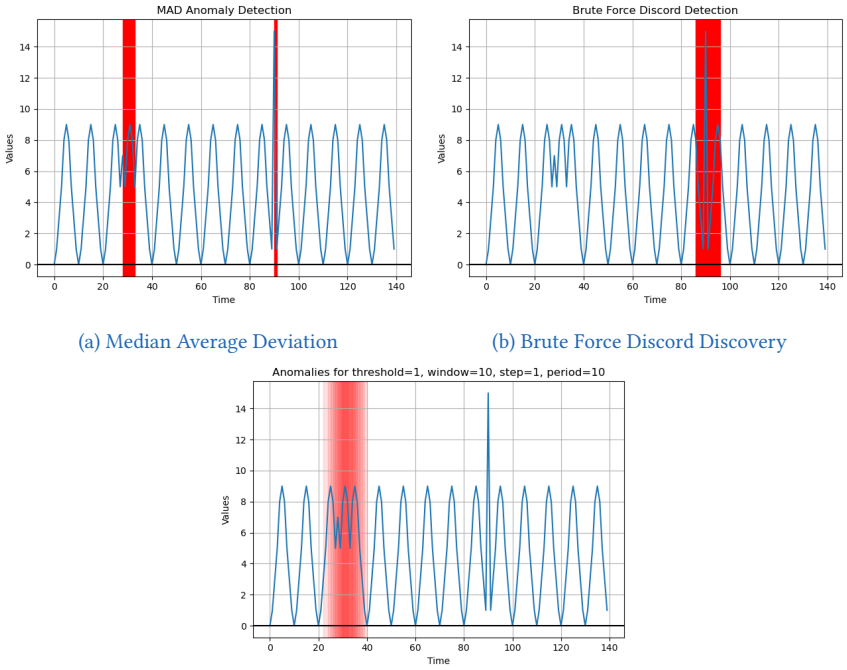
(b) Brute Force Discord Discovery



(c) Topology-Based Anomaly Detection

Figure 4.7: Topology-Based AD outperforms MAD approach. Anomalous windows are marked red.

After establishing the applicability of both the Topology AD and BFDD for discord discovery, one must still evaluate their ability to deal with false and noisy measurements. Figure 4.8 shows BFDD's inability to deal with a potentially high false measurement. This lies in the algorithms nature of simply computing the sum of differences between values of two windows, leading to outliers having a large influence on the results.



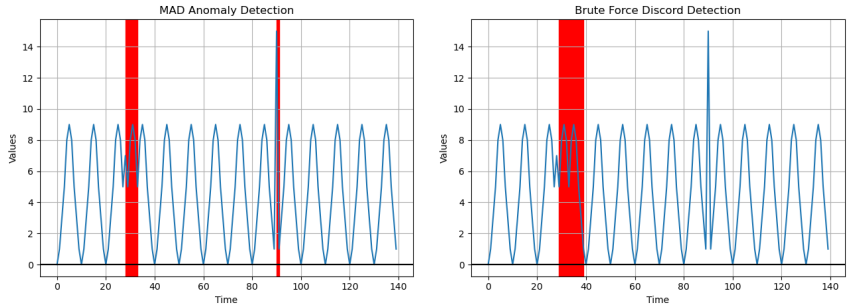
(a) Median Average Deviation

(b) Brute Force Discord Discovery

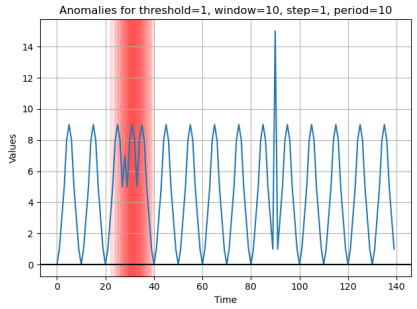
(c) Topology-Based Anomaly Detection

Figure 4.8: Topology-Based AD outperforms the baseline. Anomalous windows are marked red.

Mooi Choo Chuah et al. have, during their exploration of a more advanced method, mentioned the execution of a normalization step prior to BFDD's application to the data set. Figure 4.9 shows the effect of normalization of the data within each window on pattern preservation through the reduction of outlier influence.



(a) Median Average Deviation (b) Brute Force Discord Discovery



(c) Topology Anomaly Detection

Figure 4.9: Topology-Based AD matches normalized BFDD approach. Anomalous windows are marked red.

The regions highlighted above differ slightly, due to the fact that the proposed AD method finds more collective anomalies, opposed to BFDD's discord focused approach. The reason for choosing a discord approach over a conventional AD for evaluation, lies in the similarity of problem specification. Since a discord is the most anomalous collective anomaly in a time series, discord discovery is one of the closest approaches to collective anomaly detection of periods in a time series.

To prove the algorithm's applicability in real world scenarios, a data set was chosen containing hourly measurements of the power consumption in Austria for the year 2017. The data set was extracted from data provided by Open Power Supply [1]. A power consumption time series was chosen due to the ease of prediction of the period of recurrence as well as in the ease of explaining anomalies. The period

[1] https://open-power-system-data.org/data-sources (Accessed on: 2020-07-11)

chosen for the experiments, was 168, as this corresponds to the hours in a week. In order to achieve more reasonable runtimes, the $step\_size$ parameter was set equal to the chosen period. The underlying assumption regarding the period's length is that every normal week behaves the same in regards to power consumption. Being unlabelled, the expectation is that weeks containing bank holidays have differing power consumption patterns and must therefore be identified as anomalous.

4.1 Results

As the real world data set has no labels, the expectation is that anomalies will correspond to bank holidays in Austria for the year 2017. Intuitively, the energy consumption pattern should differ from regular working days if workers, were to stay at home and large factories, firms and offices remain closed.

Similarity thresholds above 2.8 failed to detect any anomalous windows, therefore the lowest detected values lie in the regions below. Tables presenting the results obtained through thresholding with various values are listed below.

Results for Austrian Power Consumption in 2017, for $sim\_threshold$ = 2.7 - 2.8, period = $step\_size$ = $window\_size$ = 168		
Detected Week	Bank Holiday	Bank Holiday Contained in Detection
-	6.1.2017	False
-	17.4.2017	False
-	1.5.2017	False
-	25.5.2017	False
-	5.6.2017	False
-	15.6.2017	False
-	15.8.2017	False
-	26.10.2017	False
-	1.11.2017	False
-	8.12.2017	False
24.12 - 30.12	25.12.2017	True
24.12 - 30.12	26.12.2017	True

Table 4.1: For thresholds 2.7 & 2.8 1 out of the 11 weeks have been identified.

Results for Austrian Power Consumption in 2017, for $sim\_threshold$ = 2.6, period = $step\_size$ = $window\_size$ = 168		
Detected Week	Bank Holiday	Bank Holiday Contained in Detection
-	6.1.2017	False
16.4 - 22.4	17.4.2017	True
30.4 - 6.5	1.5.2017	True
-	25.5.2017	False
-	5.6.2017	False
-	15.6.2017	False
-	15.8.2017	False
-	26.10.2017	False
-	1.11.2017	False
-	8.12.2017	False
24.12 - 30.12	25.12.2017	True
24.12 - 30.12	26.12.2017	True

Table 4.2: For threshold 2.6 4 out of the 11 weeks have been identified.

Results for Austrian Power Consumption in 2017, for $sim\_threshold$ = 2.5, period = $step\_size$ = $window\_size$ = 168		
Detected Week	Bank Holiday	Bank Holiday Contained in Detection
-	6.1.2017	False
16.4 - 22.4	17.4.2017	True
30.4 - 6.5	1.5.2017	True
21.5 - 27.5	25.5.2017	True
-	5.6.2017	False
-	15.6.2017	False
13.8 - 19.8	15.8.2017	True
22.10 - 28.10	26.10.2017	True
29.10 - 4.11	1.11.2017	True
-	8.12.2017	False
24.12 - 30.12	25.12.2017	True
24.12 - 30.12	26.12.2017	True

Table 4.3: For threshold 2.5 all but 4 weeks have been successfully found.

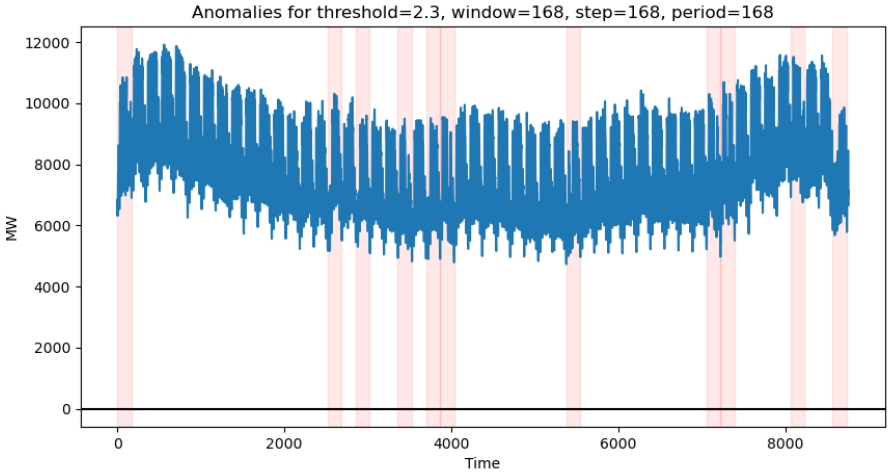
Results for Austrian Power Consumption in 2017, for $sim\_threshold$ = 2.4, period = $step\_size$ = $window\_size$ = 168		
Detected Week	Bank Holiday	Bank Holiday Contained in Detection
1.1 - 7.1	6.1.2017	True
16.4 - 22.4	17.4.2017	True
30.4 - 6.5	1.5.2017	True
21.5 - 27.5	25.5.2017	True
4.6 - 10.6	5.6.2017	True
11.6 - 17.6	15.6.2017	True
13.8 - 19.8	15.8.2017	True
22.10 - 28.10	26.10.2017	True
29.10 - 4.11	1.11.2017	True
-	8.12.2017	False
24.12 - 30.12	25.12.2017	True
24.12 - 30.12	26.12.2017	True

Table 4.4: For threshold 2.4 all but one week has been successfully found.

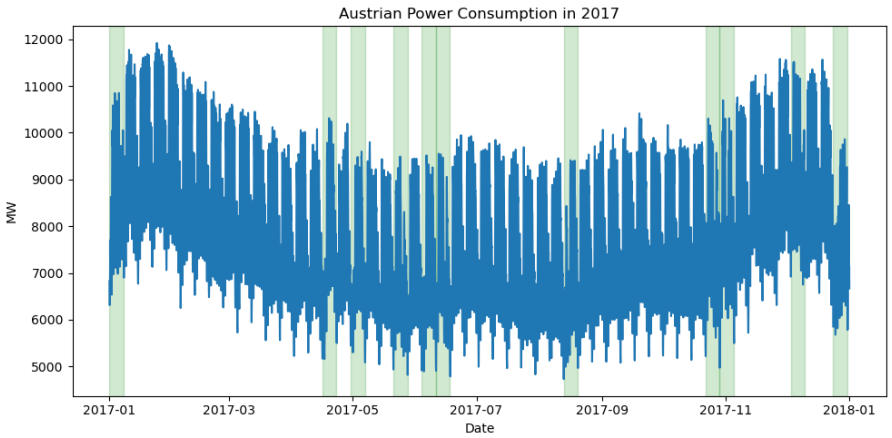
Results for Austrian Power Consumption in 2017, for $sim\_threshold$ = 2.2 - 2.3, period = $step\_size$ = $window\_size$ = 168		
Detected Week	Bank Holiday	Bank Holiday Contained in Detection
1.1 - 7.1	6.1.2017	True
16.4 - 22.4	17.4.2017	True
30.4 - 6.5	1.5.2017	True
21.5 - 27.5	25.5.2017	True
4.6 - 10.6	5.6.2017	True
11.6 - 17.6	15.6.2017	True
13.8 - 19.8	15.8.2017	True
22.10 - 28.10	26.10.2017	True
29.10 - 4.11	1.11.2017	True
3.12 - 9.12	8.12.2017	True
24.12 - 30.12	25.12.2017	True
24.12 - 30.12	26.12.2017	True

Table 4.5: For thresholds 2.2 & 2.3 all 11 anomalous regions have successfully been identified.

A visual representation of the results obtained in Table 4.5 can be seen in Figure 4.10a and can be compared to the true bank holidays marked in Figure 4.10b. All anomalous regions are identified correctly for thresholds 2.2 and 2.3 and no false positives were returned. Any threshold values below 2.2 result in false positive detections, since all weeks containing bank holidays have already been identified.



(a) Detected Anomalous Weeks are marked in red.



(b) Weeks containing Austrian Bank Holidays are marked in green.

Figure 4.10: Detected anomalies versus true anomalies.

4.2 Discussion

The results show, that the algorithm clearly has the capability to perform exceptionally well on real world data sets such as the power consumption in Austria for the year 2017. The results indicate that both the GNG and the RP can be utilized in an anomaly detection pipeline.

The results on synthetic data sets have further demonstrated the method's ability to match state-of-the-art discord discovery algorithms in identifying discords, while clearly beating the lower baseline algorithm, a MAD based AD.

Furthermore, one can deduce that the GNG has positively influenced the detection procedure through the removal of noise and outliers. This, coupled with the RP's capabilities, shows that in tandem the two methods can successfully preserve data patterns, ignoring the negative influences of outliers and bypassing pre-processing steps such as rolling the mean and frequency of data.

The results on this specific data set show the precision the proposed algorithm presents, through the detection of every bank holiday in the year 2017, whilst demonstrating no false detections.

The results further indicate that the RP with vector quantization can be made applicable to time series anomaly detection. Furthermore, the use of RP introduces generality aspects, made apparent as the detector was not tweaked in any way prior to computing results on the data set. Clearly, pattern preservation is aided through transformation of the data to state space, removing the reliance on the amplitudes of the time series.

Setting the $window\_size$ equal to the period of the data has also been shown to provide reliable results, if specific conditions such as shown in Figure 3.1 are not fulfilled.

Taking the above into consideration as well as the success the algorithm shows in matching state-of-the-art discord discovery methods results, it can be concluded that both the GNG as well as the RP can be incorporated into time series anomaly detection pipelines, and extended to detection of more anomaly classes such as the contextual ones. As for extensions that seek to identify point anomalies, the GNG could potentially cause problems, as the topology preservation has been found to

remove point anomalies, with the assumption of them being faulty measurements or noise.

5 Conclusions

The algorithm presented in this thesis has been shown to achieve state-of-the-art anomaly detection on sequential time series data, by utilizing a sliding window approach to partitioning the data, allowing for the extraction of reasonably sized RPs from the data set. Vector quantization through GNG is thereafter performed on the resulting RPs. The individual GNG networks are then compared to each other, using a similarity measure such as the Pairwise Average Minimum Distance, in order to evaluate the anomalous regions.

The proposed algorithm incorporating the GNG and RP methods was tested on various synthetic data sets and also applied to a real world data set of Austria's power consumption for the year 2017.

The algorithm matched the state-of-the-art discord discovery method BFDD and surpassed a MAD approach to anomaly detection on all synthetic data sets.

Furthermore, when applied to the real world data set of power consumption in Austria for the year 2017, the method successfully identified all weeks containing a bank holiday as anomalous. The results contained all expected anomalies and no false detections were made.

To conclude, having demonstrated very accurate results with no pre-processing steps required, it has been shown that the algorithm can successfully extract the data pattern contained within sequential time series data. The GNG has proven to not only perform vector quantization, but also to rid the data of noise and outliers, as these specific point anomalies are not within the scope of the collective anomalies to be detected.

The RP has demonstrated great potential in transferring data to state space, making the AD invariant to specific amplitudes, instead capturing only the relationship between the measurements. This aids the AD's generality component and allows for anomaly detection in various domains without the need for domain specific

knowledge related to amplitude fluctuations. This aspect was achieved by a 50 percent recurrence heuristic for the choice of the RP's $epsilon$ parameter value.

Through the use of GNG's vector quantization, the RP has been shown to be applicable to anomaly detection of time series data of reasonable sizes. The AD was however not developed with the intention of optimizing the runtime performance and can therefore likely still be improved upon, as very large data sets would results in very lengthy runtimes.

After the evaluation of the importance of both the GNG and RP and demonstration of their success in a collective anomaly detection pipeline, a similar approach could likely be derived for the remaining anomaly class of contextual anomalies.

5.1 Future Work

The proposed algorithm's assumption in requiring prior-knowledge of the period, could in future work be replaced by a dedicated season length estimation method, such as SAZED [16]. This step could potentially increase the ease of applying the algorithm with even less prior knowledge requirements.

Another related research topic could be the extension of the algorithm to contextual anomalies. The use of GNG for noise removal has already been explored, the use of RP for state space transformations however has not.

Lastly, an open problem is presented in the proposed algorithm's runtime performance. Since this paper sought to evaluate the performance of the results and disregard the runtime to a large extent, a runtime optimized approach could be developed. Especially, for larger data sets this could be of great importance.

In the event that the algorithm is not capable of functioning in reasonable time on larger data, an interesting approach would be to develop an online method based on the RP and GNG. Since small $window\_sizes$ have been shown to benefit the proposed approach, the slower input stream provided to online method's could present opportunities for GNG and RP approaches to shine. Clearly the GNG would have to compute much smaller portions of data within a given time period.

The potential shown by the GNG and the RP in an anomaly detection environment is vast, and further research must be conducted in order to find applications within

other anomaly classes' domains such as contextual anomalies. Variations of the RP and more efficient GNG implementations must be tested and evaluated in future works in order to provide a broader understanding of their joint capabilities.

Bibliography

[1] A. Andreakis, N. v. Hoyningen-Huene, and M. Beetz. Incremental unsupervised time series analysis using merge growing neural gas. In J. C. Príncipe and R. Miikkulainen, editors, *Advances in Self-Organizing Maps*, pages 10–18, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[2] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM Comput. Surv.*, 41(3), July 2009.

[3] Y. Chen, S. Su, and H. Yang. Convolutional neural network analysis of recurrence plots for anomaly detection. *International Journal of Bifurcation and Chaos*, 30:2050002–13, 01 2020.

[4] M. C. Chuah and F. Fu. Ecg anomaly detection via time series analysis. In P. Thulasiraman, X. He, T. L. Xu, M. K. Denko, R. K. Thulasiram, and L. T. Yang, editors, *Frontiers of High Performance Computing and Networking ISPA 2007 Workshops*, pages 123–135, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[5] J.-P. Eckmann, S. O. Kamphorst, and D. Ruelle. Recurrence plots of dynamical systems. *Europhysics Letters (EPL)*, 4(9):973–977, Nov 1987.

[6] J. Ehrensperger and J. Conradt. Growing neural gas for time series evaluation. 2015.

[7] D. Fišer, J. Faigl, and M. Kulich. Growing neural gas efficiently. *Neurocomputing*, 104:72–82, 03 2013.

[8] B. Fritzke. A growing neural gas network learns topologies. In G. Tesauro, D. S. Touretzky, and T. K. Leen, editors, *Advances in Neural Information Processing Systems 7*, pages 625–632. MIT Press, 1995.

[9] T. Kohonen. Self-organized formation of topologically correct feature maps. *Biological cybernetics*, 43(1):59–69, 1982.

[10] J. Lin, E. Keogh, Ada Fu, and H. Van Herle. Approximations to magic: finding unusual medical time series. In *18th IEEE Symposium on Computer-Based Medical Systems (CBMS'05)*, pages 329–334, 2005.

[11] T. M. Martinetz, S. G. Berkovich, and K. J. Schulten. 'neural-gas' network for vector quantization and its application to time-series prediction. *IEEE Transactions on Neural Networks*, 4(4):558–569, 1993.

[12] M. Paisner, M. Cox, and D. Perlis. Symbolic anomaly detection and assessment using growing neural gas. pages 175–181, 11 2013.

[13] P. M. Roth, S. Sternig, H. Grabner, and H. Bischof. Classifier grids for robust adaptive object detection. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2727–2734, 2009.

[14] Q. Sun, H. Liu, and T. Harada. Online growing neural gas for anomaly detection in changing surveillance scenes. *Pattern Recognition*, 64:187 – 201, 2017.

[15] M. Tambos. Online anomaly detection in time series using merge growing neural gas, 06 2015.

[16] M. Toller, T. Santos, and R. Kern. Sazed: parameter-free domain-agnostic season length estimation in time series data. *Data Mining and Knowledge Discovery*, 33(6):1775–1798, 2019.

[17] N. Waniek, S. Bremer, and J. Conradt. Real-time anomaly detection with a growing neural gas. In S. Wermter, C. Weber, W. Duch, T. Honkela, P. Koprinkova-Hristova, S. Magg, G. Palm, and A. E. P. Villa, editors, *Artificial Neural Networks and Machine Learning – ICANN 2014*, pages 97–104, Cham, 2014. Springer International Publishing.

Appendix