tGLAD: A sparse graph recovery based approach for multivariate time series segmentation

Shima Imani & Harsh Shrivastava

Microsoft Research, Redmond, USA
{shimaimani,hshrivastava}@microsoft.com

Abstract. Segmentation of multivariate time series data is a valuable technique for identifying meaningful patterns or changes in the time series that can signal a shift in the system's behavior. We introduce a domain agnostic framework 'tGLAD' for multivariate time series segmentation using conditional independence (CI) graphs that capture the partial correlations. It draws a parallel between the CI graph nodes and the variables of the time series. Consider applying a graph recovery model uGLAD to a short interval of the time series, it will result in a CI graph that shows partial correlations among the variables. We extend this idea to the entire time series by utilizing a sliding window to create a batch of time intervals and then run a single uGLAD model in multitask learning mode to recover all the CI graphs simultaneously. As a result, we obtain a corresponding temporal CI graphs representation of the multivariate time series. We then designed a first-order and second-order based trajectory tracking algorithm to study the evolution of these graphs across distinct intervals. Finally, an 'Allocation' algorithm is designed to determine a suitable segmentation of the temporal graph sequence which corresponds to the original multivariate time series. tGLAD provides a competitive time complexity of $O(N)$ for settings where number of variables $D << N$. We demonstrate successful empirical results on a Physical Activity Monitoring data.[1]
Keywords: Multivariate time series segmentation, Conditional Independence Graphs, Sparse Graph recovery

1 Introduction

Time series segmentation is the process of dividing a time series into multiple segments, or sub-series, based on certain characteristics or patterns. Segmentation has many benefits, such as reducing a long time series into manageable sections to facilitate labeling by a human or machine annotator, and uncovering unexpected actionable patterns in data through exploration. For example, it can be used in finance to identify trends and patterns in stock prices, in marketing to analyze consumer behavior, and in healthcare to monitor patient vital signs. This helps to understand the underlying dynamics of the data and even make predictions about future events [2,21,37].

[1] *Software*: https://github.com/Harshs27/tGLAD

There are numerous algorithms available for segmenting time series with majority of them primarily designed to handle the univariate case. If N is the length of the time series, most algorithms have an expected time complexity of $O(N^2)$, however, some more recent algorithms have achieved an $O(N \log N)$ time complexity with certain limiting approximations. Some time series segmentation methods are designed for specific domains, limiting their broader application. Additionally, some methods make assumptions about the semantic segments being well-defined, but they may not always align with real-world data and thereby hinder their effectiveness. The extensions suggested for these methods to handle multivariate data are non-trivial and often do not perform well in practice. Related works section covers them in detail.

Consider a small slice of a multivariate time series consisting of D variables, say from T_1 to T_{10} which contains no crucial segmentation points. For this slice, we can expect the correlation between the D variables to be roughly the same throughout $\text{corr}^D(T_1) \sim \text{corr}^D(T_{10})$. Now, let's assume that there is a segmentation point at T_{11}. For instance, if we are monitoring the sensor data of an athlete, we can consider that at time T_{11}, the athlete switched activity from jogging to sprinting. We now expect that the correlations among the variables will change at the segmentation points, $\text{corr}^D(T_{10}) \not\sim \text{corr}^D(T_{11})$. Our proposed framework, called `tGLAD`, is designed to efficiently detect this change of correlations which indicate segmentation points.

To realize this intuition, we identified a novel cross-domain application of sparse graph recovery for time series analysis. Briefly, given input variables and their samples, sparse graph recovery methods output a graph whose edges capture the direct dependencies among the variables. In our work, we focus on recovering special type of graphs, called the conditional independence (CI) graphs [28]. The CI graphs capture the partial correlations between the variables, which can be either positive or negative. Among the many different algorithms to recover CI graphs, we choose a recently developed state-of-the-art deep model called `uGLAD` [31,32]. Its multitask learning ability enables a single instance of the model to run on batch input and recover multiple graphs simultaneously, a property that paves way for the high efficiency of `tGLAD`. Although, one can theoretically utilize any algorithm under the larger umbrella of the sparse graph recovery methods, the methods section will highlight the key reasons which justifies our choice of using the combination of CI graphs and `uGLAD`.

The process followed by the `tGLAD` framework for doing multivariate time series segmentation is as follows. We divide the time series into sub-sequences or batches and then run a CI graph recovery model `uGLAD` that gives a corresponding temporal graphs. The nodes of CI graphs are the variables of the multivariate time series and the edges capture the partial correlation strength between the variables. In essence, we have distilled down some relevant information of the time series in the temporal CI graphs. As per our intuition, the instances where the consecutive CI graphs differ a lot in their correlations, those points of the temporal graphs will correspond to the segmentation points in the time series. We use this insight to develop our multi-step framework `tGLAD`. Thus, we developed

efficient algorithms to capture the dynamics or the evolution pattern of the temporal CI graphs which in turn help us identify the segmentation regions in the original time series.

Listing the key contributions of our work. Please note that we use the terms, time series and sequences, interchangeably throughout.

1. A novel cross-domain approach for multivariate time series segmentation based on using sparse graph recovery algorithms.
2. Efficient method to give linear $O(N)$ time complexity in terms sequence length for cases where the number of variables follow $D^3 << N$
3. Provide explainability and transparency by giving insights into reasons for the segmentation.
4. A domain agnostic framework that can be applied for time series from various domains.

2 Related works

Our framework is a combination of the literature from time series segmentation and sparse graph recovery. So, we discuss relevant research from both of them to provide background knowledge.

Segmentation Methods. There are several time series segmentation methods available that use different approaches to segment a time series into different classes based on changes in its temporal shape patterns. We divide the existing methods into domain specific and domain agnostic ones.

Domain specific. If one narrows down the scope for analysing time series to a specific field, specialized methods can be developed by utilizing the domain-specific insights. Survey in [17] did a collective analysis of various such methods and also highlighted one key insight that for almost all the methods some background on the nature of the domain and motion is needed. Although, the recent observed trend is to develop domain agnostic approaches and we can find interesting techniques in this category. For example, Automobile trajectories were studied in [10], electroencephalography data was analysed in [15], electrical power consumption analysis in [23], music sequence analysis in [26], biological time series in [21], human motion segmentation was investigated in [2,3,16] among others.

Domain agnostic. In attempt to design domain agnostic techniques for wider adaptability, FLOSS (Fast Low-cost Online Semantic Segmentation) [8] was developed. It is a popular method which produces an Arc Curve (AC) that annotates the original time series with information about the likelihood of a regime change at each point in the series. The AC is used to identify segments with similar temporal shape patterns that are likely to belong to the same class and occur within close temporal proximity to each other. Another method called ESPRESSO (Entropy and ShaPe awaRe timE-Series SegmentatiOn) [5], is a hybrid approach that uses both shape pattern and statistical distribution of time series to segment time series data. ESPRESSO uses a modified version of FLOSS by incorporating the Weighted Chained Arc Curve to capture the

density of pattern repetition with time. Recently proposed ClaSP (Classification Score Profile) is a self-supervised time series segmentation method that uses overlapping windows to split time series into hypothetical partitions. For each partition, a binary classifier is trained and evaluated using cross-validation. The degree of self-similarity is recorded for each offset and then the classification score profile is computed, which is ultimately used for segmenting time series data [6]. Other relevant methods include [4,12,18,19].

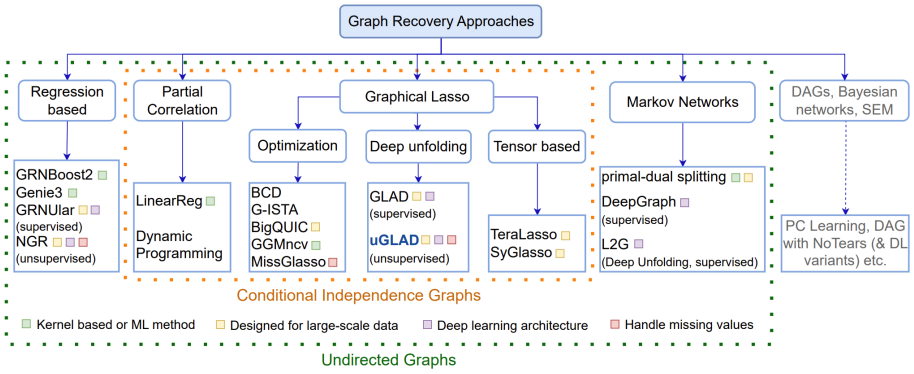

Fig. 1: *Overview of Sparse Graph Recovery methods.* We focus on methods that recover undirected graphs which capture direct dependence among their nodes or features. `tGLAD` framework utilizes a recently developed deep model, `uGLAD`, that outputs a conditional independence graph between in the features. Our framework can potentially use other methods and will be interesting topic for future explorations. (partly borrowed from [30])

Sparse graph recovery. Given data with D features and M samples as input, the aim of the sparse graph recovery methods is to obtain a probabilistic graphical model [14] that potentially shows sparse connections between the D features. We focus on methods that recover undirected graphical models, refer Fig. 1. Sparse graph recovery methods have been used for various applications like gene regulatory network discovery [1,11,20,34,35,36], understanding Digester functioning to increase Methane yield [32], extracting insights from an Infant mortality data [29,30], studying autism by analysing brain sensory signals [22] among many others.

Conditional Independence graphs. The edges of a CI graph show the partial correlation between the nodes or features. The partial correlation can be considered as capturing direct dependency between the features as it is the conditional probability of the features under consideration given all the other features. Refer inner block in orange of Fig. 1. Popular formulations of recovering CI graph include optimizing the graphical lasso objective [7,25] which include deep models like [27,31,32,33] or dynamic programming based approach to directly evaluate the expression of partial correlations. Survey [28] formalizes the definition of

CI graphs, categorizes various methods that recover such graphs, describe and compares their performance, provide their implementation details and discuss their applications. It is a good entry point to understanding the umbrella of methods that recover CI graphs. The method by [9] they utilized temporal graphs to understand dynamics of systems which is similar idea as ours but was not developed for time series segmentation settings.

3 Methods

We introduce the necessary definitions and notations to facilitate our discussions followed by the steps followed by the tGLAD framework.

3.1 Definitions

A **multivariate time series** T of length N and dimension D is a sequence of real-valued vectors

$$T = t_1, t_2, \ldots, t_N, \text{ where } t_i \in \mathbb{R}^D$$

A **Subsequence** is defined as a local section of a time series that consists of a continuous subset of its values. A subsequence $T_{i,M}$ of a time series T is a continuous subset of the values from T of length M starting from position i. Formally, $T_{i,M} = t_i, t_{i+1}, \ldots, t_{i+M-1}$, where $1 \leq i \leq N - M + 1$.

In order to extract continuous subsequences from time series, we utizlize the **stride length** shifting to determine the next subsequence. In time series data, the stride length is the number of data points by which we shift the starting position of the current subsequence to extract the position of the next subsequence. For example, a stride length s means that if the current subsequence is located at $T_{i,M}$ where i where is the starting position of the subsequence from T with length M, then the next subsequence is $T_{i+s,M}$ with the starting position at $i + s$.

3.2 tGLAD framework

Figure 2 enumerates the steps followed by tGLAD to do multivariate time-series segmentation. The details for each of these steps are given below.

(A) Identifying variables and prepare batch input for sparse graph recovery

For all the variables in the given multivariate time series, basic preprocessing is done which includes missing value imputation using a forward filling algorithm. The data is now partitioned into small chunks using a fixed window size M and stride length s and runs over the entire time series. The window size determined based on the approach suggested in [13]. We now end up with $B = (N - M + 1)/s$ batches, with each having M samples for D variables. The input to the graph recovery algorithm will be the batch of samples, represented as a tensor of size

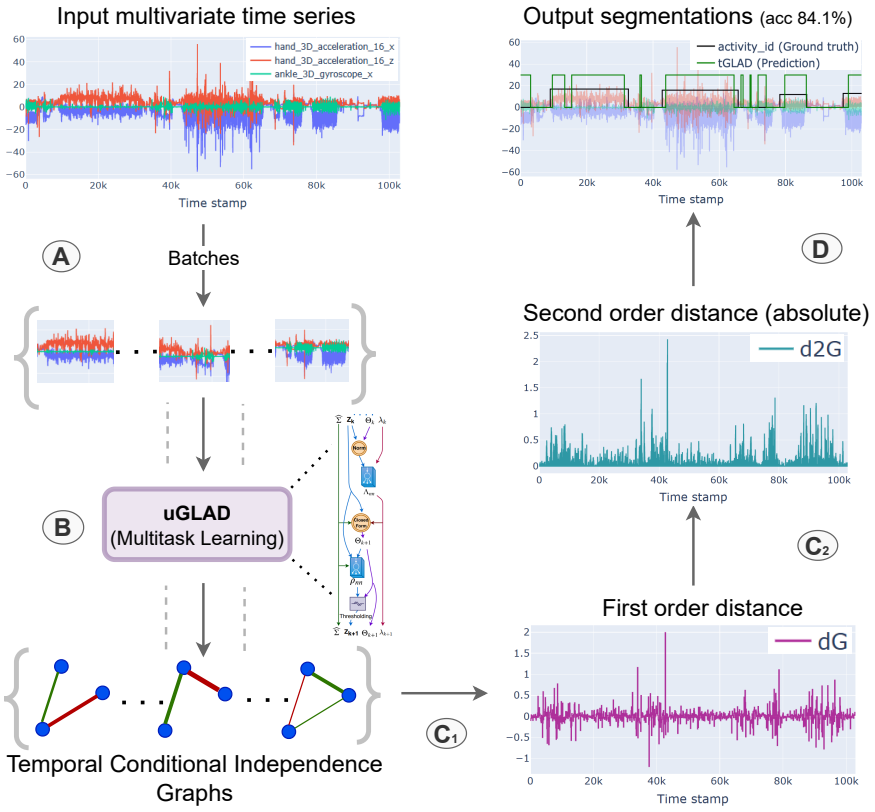

Fig. 2: *tGLAD framework*. (**A**) The time series is divided into multiple intervals by using a sliding window to create a batch of intervals. (**B**) Run a single uGLAD model in multitask learning (or batch) mode setting to recover a CI graph for every input batch. This gives a corresponding set of temporal CI graphs. The entire input is processed in a single step as opposed to obtaining a CI graph for each interval individually. ($\mathbf{C}_1$) Get the first order distance, dG sequence, of the temporal CI graphs which captures the distance between the consecutive graphs. This is supposed to give higher values at the segmentation points. ($\mathbf{C}_2$) Again take a first order distance of the sequence in the previous step and then its absolute value to get $d2G$ sequence, which further accentuates the values at the segmentation points. (**D**) Apply a threshold to zero out the smaller values of $d2G$ and identify the segmentation blocks using an 'Allocation' algorithm.

$X \in \mathbb{R}^{B \times M \times D}$.

(B) Obtaining the temporal Conditional Independence graphs

The aim of the sparse graph recovery algorithm is to run on the input from step (A), denoted by X and output corresponding set of graphs, whose adjacency matrix is represented here by the tensor $\mathrm{P} \in \mathbb{R}^{B \times D \times D}$. There are 2 key requirements from any such method, namely (1) The resultant graph should capture direct dependencies between the features (2) The method should be efficient. We chose a combination of CI graphs and uGLAD model keeping in mind the desiderata desired.

Algorithm 1: Allocating segments

```
Function get-segments(d2G, Z=5):
    B ← len(d2G)
    labels ← [1] × B
    /* Removing noise                */
    d2G < 0.5 = 0
    /* The window size is M          */
    For i ← 0 to B do
        If d2G[i] > 0 then
            start = max(0, i − M · Z)
            end = min(B − 1, i + M · Z)
            labels[start : end] ← 0
    return labels
```

Why CI graphs? CI graphs capture partial correlations between the features which model direct dependencies between them. The nodes are the features and the edge weights carry the partial correlation value that lies in the range $[-1, 1]$. This additionally provides us with the positive or negative correlation information, which later help us in determining the relevant features that result in a segmentation prediction as well as provide explainability and transparency to our framework.

Why uGLAD? Introduced in [31], uGLAD is a deep-unfolding (or unrolled algorithm) based model which is an unsupervised extension of the GLAD [33] model. These models are based on the optimization of the graphical lasso objective which assumes that the observed data comes from an underlying multivariate Gaussian distribution. Owing to the deep-unfolding done based on the Aternating Minimization updates and then expressiveness provided by the neural network based parameterization, these models are shown to better capture the tail-distribution points and also improve sample complexity results. Apart from the theoretical advantages and performance improvements over the other CI graph recovery methods, uGLAD is efficient as well. The tensor based implementation of uGLAD allows it to do multitask learning. This enables a single model to recover the entire batch of data simultaneously. We want to point out that we consider the sample data within a window size follow i.i.d. setting for the multivariate Gaussian assumption to work.

We run uGLAD in 'batch mode' to obtain all the underlying precision matrices at once, $\theta \leftarrow$uGLAD(X), where $\theta \in \mathbb{R}^{B \times M \times D}$. The calculation of the partial correlation matrix P is straightforward from Θ, refer [28]. The parameter sharing across these different tasks helps maintain robustness against noisy data and facilitates transfer learning. We thus obtain a series of temporal CI graphs,

represented by the adjacency matrices $\mathbf{G} = [G_1, G_2, \cdots, G_B] \in \mathbb{R}^{B \times D \times D}$ using P. Each entry of the adjacency matrix is equal to the partial correlation value, $G_b[p, q] = \rho(D_p, D_q)$ for the b^{th} batch and D_k represent the k^{th} time series variable. The temporal graphs can be seen as distilling some relevant information from the original multivariate time series data in form of graphs.

(C) Towards segmentation of the corresponding temporal CI graphs

Our formulation is based on the assumption that the key signals needed to successfully segment the original time series are captured in the corresponding temporal graphs and that the correlation among the features are informative enough for the task. So, if we are able to segment the temporal graphs, we can map the segmentation to the original time series.

(C_1) We compute the first-order distance sequence $dG \in \mathbb{R}^B$ by finding the distance of the consecutive graphs in the temporal graph series $\mathbf{G}$. For each entry $b \in B$ of dG, we measure the distance between its recovered graph and the next neighbor as

$$dG[b] = \text{distance}(G_b, G_{b+1}) = \sum_{p,q} (G_b[p, q] - G_{b+1}[p, q]) \quad \forall p, q \in \{1, \cdots, D\}$$

where weights are the partial correlation values of the edges of the CI graphs under consideration.

(C_2) Given the sequence dG, next we compute the second-order distance sequence $d2G$ by applying the following distance operation

$$d2G[b] = \text{abs}\left(dG[b] - dG[b-1]\right), \quad \forall b \in (1, B)$$

The first-order distance measures the change between each recovered graph and its next neighbor, while the second-order distance highlights potential segmentation points. While there are other distance metrics that can potentially be used, in our experiments, we found that the first-order and second-order distances described above worked well for detecting segmentation points. The output of this trajectory tracking step is the $d2G$ sequence.

D. Allocation algorithm for obtaining the final segmentation

We develop an 'Allocation' algorithm to obtain the final segmentation points from the $d2G$ sequence. We first filter out small noises in $d2G$ by applying a conservative threshold. We then traverse the sequence $d2G$ sequentially and mark the start of a segmentation a new block if we observe a non-zero value. We also disregard any changes in behavior or segmentation points that occur in less than Z times the window size (M), otherwise the segmentation size will be significantly smaller than the window size and we will not be able to catch it. We usually choose $Z \sim 5$ in our experiments. The allocation process (Alg. 1) reads the $d2G$ sequence and predict the `tGLAD` segmentation scores.

3.3 Time complexity analysis of tGLAD

We analyse the time complexity of each of the steps followed by the tGLAD framework below.

- *(A)* Creation of batches will require a single full scan of the time series, so complexity is $O(N)$.
- *(B)* the time complexity of this step will consist of the input covariance matrix creation $O(N \cdot D^2)$, and then running uGLAD in batch mode. For a single input, uGLAD runs in $O(D^3 \cdot E)$, so for B batches the sequential runtime will be $O(B \cdot D^3 \cdot E)$. Since, we can process batches in parallel with uGLAD batch mode, in practice we observe significantly less runtime. The worst case scenario will be when $B \to N$, giving time complexity as $O(N \cdot D^3 \cdot E)$.
- *(C_1)* The first order distance function goes through the entire length of the temporal graph sequence and each time enumerate all possible edges between graphs having D nodes. So, it has a time complexity of $O(N \cdot D^2) \sim O(N)$.
- *(C_2)* Creation of $d2G$ will require a single full scan of the dG, so complexity is $O(N)$.
- *(Allocation algorithm)* Scans the $d2G$ array once, so complexity is $O(N)$.

The overall time complexity of the tGLAD framework in cases where the number of variables are not high, $D^3 << N$, is $O(N) + O(N \cdot D^3 \cdot E) + O(N \cdot D^2) \sim O(N)$. The worst case time complexity, where the number of variables are so large that we cannot leverage the power of the multitask learning in batch mode of uGLAD, is $O(N) + O(N \cdot D^3 \cdot E) + O(N \cdot D^2) \sim O(N \cdot D^3)$.

4 Experiments

We evaluate the tGLAD framework on a real world body sensor dataset. Since, it is a novel framework, we conduct several design choices experiments to understand their impact on tGLAD's performance.

4.1 PAMAP2 Dataset

To get a realistic sense about the effectiveness of our approach, we conducted experiments on the PAMAP2 Physical Activity Monitoring dataset [24]. This dataset captures sensor data from multiple participants engaging in a variety of physical activities, making it a valuable resource for activity recognition and algorithm development. Our analysis was primarily based on multi-dimensional time series with the following three signals: the hand acceleration signal in the x-axis and z-axis, and the ankle gyroscope signal in the x-axis, which allowed us to examine the movements and rotations of the hand and ankle during physical activity.

Fig. 2 (A & D) shows a three-hour segment of this data collected from one of the participants, highlighting their physical activities such as ironing (44 minutes), vacuum cleaning (42 minutes), and stair activity (ascending 15

minutes, descending 10 minutes), as well as periods of inactivity (transient). Fig. 2 additionally shows all the steps of the `tGLAD` framework followed in order to segment the data.

4.2 Results

We chose accuracy as the metric to evaluate the segmentation performance. The accuracy is measured as the penalty for mislabeling the segmentation. For the ground truth time series, we put label=1 whenever an activity occurs and at every segmentation point where there is no activity, we switch the label=0. For the prediction labels, we consider the $d2G$ sequence obtained from Fig. 2(D) and use the Allocation technique describe in Alg. 1, with parameter $Z = 5$.

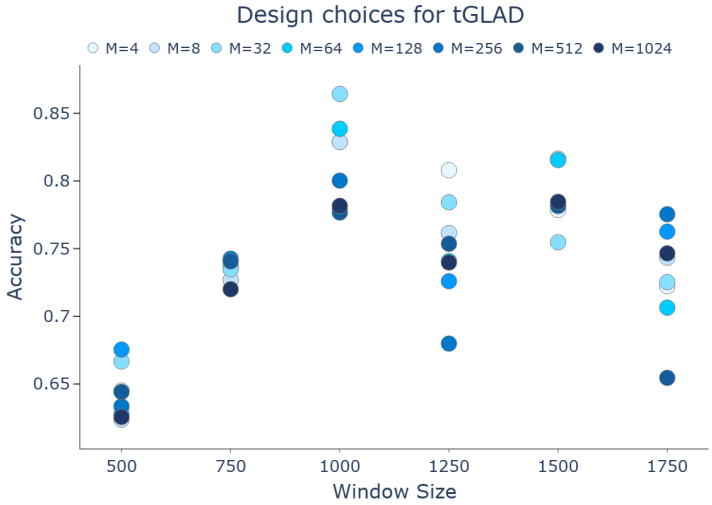

Fig. 3: *Design choices for `tGLAD`.* Examining the segmentation accuracy on the PAMAP2 dataset which records body sensor data. We vary the window size on the x-axis and for each window size, we evaluate the performance for varying batch sizes (M). The stride length was fixed at 100 for all the experiments.

We achieved an accuracy of 84.1% for the PAMAP2 dataset using a window size of 1000 , batch size of 64 and stride length of 100, indicating that the `tGLAD` framework is effective for physical activity monitoring based time series. The window size is the chunk of the time series considered at a time for processing CI graphs, so it is an important parameter to be chosen while running our framework. The batch size is the number of graphs that are recovered by a single `uGLAD` model. As we are using multitasking, the parameters of the model are shared among the graphs within a batch, hence this is also an important parameter that can affect `tGLAD`'s performance. Small batch size will lead to

increased runtime as more number of batches to process, less robust to noise but more accurate graph recovery, while on the other hand, higher batch size will be efficient in term of runtime and robustness to anomalies, but since it has to recover graphs that are potentially sampled from different underlying distributions, the accuracy might take a hit. So, it is imperative that we do a study on effect of these design choices. Thus, in order to gain insights into the performance of tGLAD with respect to the batch size and window size, we explored the impact on the segmentation accuracy by doing a grid plot over a range of size choices, as illustrated in Fig. 3.

Analysing the results indicate that changes in batch size and window sizes do not significantly affect the accuracy of the segmentation. If we consider any fixed window size, we do not see much variance in the performance over different batch sizes, that suggests a good graph recovery performance of the uGLAD model. Thus, we can potentially increase the batch size for faster runtimes, without compromising much on the accuracy. Lots of research has been done on the choice of window size, with some methods being more sensitive than others. We do see variance in the performance of tGLAD with change in window size, still the results suggest that a reasonable window size can be chosen to achieve a satisfactory segmentation label. The choice of the window size also depends considerably on the type of data as well.

5 Conclusions

We introduce a domain agnostic multivariate time series framework called tGLAD. It is a novel cross-domain approach that maps the original time series to a corresponding temporal graph representation which makes the problem of finding segmentation easier and efficient. The choice of a recently developed deep model uGLAD for recovering conditional independence graphs gives the much needed efficiency to our framework. We identified a unique use of the multitask learning ability of uGLAD model which also makes the case of batch learning in sparse graph recovery models more lucrative. Additionally, from the plethora of graph choices available, this work also narrowed down the type to conditional independence graphs. The CI graphs capture the intuition that correlation among the multivariate timeseries variables will change significantly at the segmentation points. We demonstrate successful segmentation results on the challenging PAMAP2 dataset, with achieving an accuracy of 84.10% along with performing a parameter exploration study.

5.1 Future Work

We have plans to pursue two directions of research for expanding tGLAD. Firstly, we will investigate the potential for segmenting univariate time series data using the tGLAD framework. our approach consists of 'smartly' converting the 1D sequence to multidimensional time series and then use the tGLAD framework. This approach seems promising due to its high efficiency in terms of time complexity

and hopefully good segmentation accuracy. Secondly, we aim to extend the `tGLAD` framework to work in real-time or online settings. This will require adapting the framework and evaluating the trade-offs between computational efficiency and segmentation accuracy. The results of this research could have significant implications for fields such as finance, healthcare, and industrial monitoring.

5.2 Ethical Concerns

Our method does not introduce new ethical issues, but ethical considerations would be important if it were to be applied to sensitive data.

References

1. Aluru, M., Shrivastava, H., Chockalingam, S.P., Shivakumar, S., Aluru, S.: Engrain: a supervised ensemble learning method for recovery of large-scale gene regulatory networks. Bioinformatics (2021)
2. Aminikhanghahi, S., Cook, D.J.: A survey of methods for time series change point detection. Knowledge and information systems **51**(2), 339–367 (2017)
3. Aoki, T., Lin, J.F.S., Kulić, D., Venture, G.: Segmentation of human upper body movement using multiple imu sensors. In: 2016 38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC). pp. 3163–3166. IEEE (2016)
4. Castellini, A., Bicego, M., Masillo, F., Zuccotto, M., Farinelli, A.: Time series segmentation for state-model generation of autonomous aquatic drones: A systematic framework. Eng. Appl. Artif. Intell. **90**, 103499 (2020)
5. Deldari, S., Smith, D.V., Sadri, A., Salim, F.D.: Espresso: Entropy and shape aware time-series segmentation for processing heterogeneous sensor data. Proc. ACM Interact. Mob. Wearable Ubiquitous Technol. **4**, 77:1–77:24 (2020)
6. Ermshaus, A., Schäfer, P., Leser, U.: Clasp–parameter-free time series segmentation. arXiv preprint arXiv:2207.13987 (2022)
7. Friedman, J., Hastie, T., Tibshirani, R.: Sparse inverse covariance estimation with the graphical lasso. Biostatistics **9**(3), 432–441 (2008)
8. Gharghabi, S., Yeh, C.C.M., Ding, Y., Ding, W., Hibbing, P., LaMunion, S., Kaplan, A., Crouter, S.E., Keogh, E.: Domain agnostic online semantic segmentation for multi-dimensional time series. Data mining and knowledge discovery **33**(1), 96–130 (2019)
9. Hallac, D., Park, Y., Boyd, S., Leskovec, J.: Network inference via the time-varying graphical lasso. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 205–213 (2017)
10. Harguess, J., Aggarwal, J.: Semantic labeling of track events using time series segmentation and shape analysis. In: 2009 16th IEEE International Conference on Image Processing (ICIP). pp. 4317–4320. IEEE (2009)
11. Haury, A.C., Mordelet, F., Vera-Licona, P., Vert, J.P.: TIGRESS: trustful inference of gene regulation using stability selection. BMC systems biology **6**(1) (2012)
12. Imani, S., Abdoli, A., Keogh, E.: Time2cluster: Clustering time series using neighbor information
13. Imani, S., Keogh, E.: Multi-window-finder: Domain agnostic window size for time series data (2021)

14. Koller, D., Friedman, N.: Probabilistic graphical models: principles and techniques. MIT press (2009)
15. Kozey-Keadle, S., Libertine, A., Lyden, K., Staudenmayer, J., Freedson, P.S.: Validation of wearable monitors for assessing sedentary behavior. Medicine & Science in Sports & Exercise **43**(8), 1561–1567 (2011)
16. Lan, R., Sun, H.: Automated human motion segmentation via motion regularities. The Visual Computer **31**, 35–53 (2015)
17. Lin, J.F.S., Karg, M., Kulić, D.: Movement primitive segmentation for human motion modeling: A framework for analysis. IEEE Transactions on Human-Machine Systems **46**(3), 325–339 (2016)
18. Lu, S., Huang, S.: Segmentation of multivariate industrial time series data based on dynamic latent variable predictability. IEEE Access **8**, 112092–112103 (2020)
19. Machné, R., Murray, D.B., Stadler, P.F.: Similarity-based segmentation of multi-dimensional signals. Scientific Reports **7** (2017)
20. Moerman, T., Aibar Santos, S., Bravo González-Blas, C., Simm, J., Moreau, Y., Aerts, J., Aerts, S.: Grnboost2 and arboreto: efficient and scalable inference of gene regulatory networks. Bioinformatics **35**(12), 2159–2161 (2019)
21. Omranian, N., Mueller-Roeber, B., Nikoloski, Z.: Segmentation of biological multivariate time-series data. Scientific reports **5**(1), 1–6 (2015)
22. Pu, X., Cao, T., Zhang, X., Dong, X., Chen, S.: Learning to learn graph topologies. Advances in Neural Information Processing Systems **34** (2021)
23. Reinhardt, A., Christin, D., Kanhere, S.S.: Predicting the power consumption of electric appliances through time series pattern matching. In: Proceedings of the 5th ACM Workshop on Embedded Systems For Energy-Efficient Buildings. pp. 1–2 (2013)
24. Reiss, A., Stricker, D.: Introducing a new benchmarked dataset for activity monitoring. In: 2012 16th international symposium on wearable computers. pp. 108–109. IEEE (2012)
25. Rolfs, B., Rajaratnam, B., Guillot, D., Wong, I., Maleki, A.: Iterative thresholding algorithm for sparse inverse covariance estimation. Advances in Neural Information Processing Systems **25**, 1574–1582 (2012)
26. Serra, J., Müller, M., Grosche, P., Arcos, J.L.: Unsupervised music structure annotation by time series structure features and segment similarity. IEEE Transactions on Multimedia **16**(5), 1229–1240 (2014)
27. Shrivastava, H.: On Using Inductive Biases for Designing Deep Learning Architectures. Ph.D. thesis, Georgia Institute of Technology (2020)
28. Shrivastava, H., Chajewska, U.: Methods for recovering conditional independence graphs: A survey. arXiv preprint arXiv:2211.06829 (2022)
29. Shrivastava, H., Chajewska, U.: Neural graphical models. arXiv preprint arXiv:2210.00453 (2022)
30. Shrivastava, H., Chajewska, U.: Neural graph revealers. arXiv preprint arXiv:2302.13582 (2023)
31. Shrivastava, H., Chajewska, U., Abraham, R., Chen, X.: A deep learning approach to recover conditional independence graphs. In: NeurIPS 2022 Workshop: New Frontiers in Graph Learning (2022), https://openreview.net/forum?id=kEwzoI3Am4c
32. Shrivastava, H., Chajewska, U., Abraham, R., Chen, X.: uGLAD: Sparse graph recovery by optimizing deep unrolled networks. arXiv preprint arXiv:2205.11610 (2022)
33. Shrivastava, H., Chen, X., Chen, B., Lan, G., Aluru, S., Liu, H., Song, L.: GLAD: Learning sparse graph recovery. arXiv preprint arXiv:1906.00271 (2019)

34. Shrivastava, H., Zhang, X., Aluru, S., Song, L.: Grnular: Gene regulatory network reconstruction using unrolled algorithm from single cell rna-sequencing data. bioRxiv (2020)
35. Shrivastava, H., Zhang, X., Song, L., Aluru, S.: Grnular: A deep learning framework for recovering single-cell gene regulatory networks. Journal of Computational Biology **29**(1), 27–44 (2022)
36. Vân Anh Huynh-Thu, A.I., Wehenkel, L., Geurts, P.: Inferring regulatory networks from expression data using tree-based methods. PloS one **5**(9) (2010)
37. Yeh, C.C.M., Zhu, Y., Ulanova, L., Begum, N., Ding, Y., Dau, H.A., Silva, D.F., Mueen, A., Keogh, E.: 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). pp. 1317–1322. Ieee (2016)