Machine Learning
https://doi.org/10.1007/s10994-023-06405-x

Graph similarity learning for change-point detection in dynamic networks

Déborah Sulem[1] · Henry Kenlay[2] · Mihai Cucuringu[1,3,4] · Xiaowen Dong[2]

Received: 18 May 2022 / Revised: 1 May 2023 / Accepted: 21 August 2023

Abstract
Dynamic networks are ubiquitous for modelling sequential graph-structured data, e.g., brain connectivity, population migrations, and social networks. In this work, we consider the discrete-time framework of dynamic networks and aim at detecting change-points, i.e., abrupt changes in the structure or attributes of the graph snapshots. This task is often termed *network change-point detection* and has numerous applications, such as market phase discovery, fraud detection, and activity monitoring. In this work, we propose a data-driven method that can adapt to the specific network domain, and be used to detect distribution changes with no delay and in an *online* setting. Our algorithm is based on a *siamese graph neural network*, designed to learn a graph similarity function on the graph snapshots from the temporal network sequence. Without any prior knowledge on the network generative distribution and the type of change-points, our learnt similarity function allows to more effectively compare the current graph and its recent history, compared to standard graph distances or kernels. Moreover, our method can be applied to a large variety of network data, e.g., networks with edge weights or node attributes. We test our method on synthetic and real-world dynamic network data, and demonstrate that it is able to perform online network change-point detection in diverse settings. Besides, we show that it requires a shorter data history to detect changes than most existing state-of-the-art baselines.

Keywords Dynamic networks · Change-point detection · Graph similarity learning · Siamese graph neural network

1 Introduction

The study of dynamic—or temporal, evolutionary, time-varying—networks has become very popular in the last decade, with the increasing amount of sequential data collected from structured and evolving systems, e.g., online communication platforms (Kumar et al., 2019), co-voting networks (Wilson et al., 2019), and brain fMRI data (Cribben & Yu, 2017). In general, adding a time component to network data is a richer representation which allows for a more powerful analysis of these systems (Skarding et al., 2021).

Editor: Joao Gama.

Extended author information available on the last page of the article

To analyse these networks, a variety of methodology has been proposed and recently reviewed by several survey papers, for instance, on dynamic neural networks (Han et al., 2021), dynamic higher-order networks (Majhi et al., 2022), and dynamic network embedding (Xue et al., 2022).

Modelling the temporal evolution of a network is particularly important when the latter is governed by a non-stationary underlying process, whose dynamics undergo abrupt switches or breaks, called *change-points*. For instance, social network interactions display different structures along time, which can be affected by external events such as social unrest or criminal attacks (Bourqui et al., 2009). Moreover, detecting structural breaks and finding stationary phases in dynamic networks have diverse applications, from brain connectivity state segmentation (Ondrus et al., 2021) to phase discovery in financial correlation networks (Barnett & Onnela, 2016). Besides, real-world dynamic networks are often structured around functional groups or densely connected communities (Rossetti & Cazabet, 2018). Therefore, the temporal evolution of such networks has been often analysed through changes in these substructures (Corneli et al., 2018; Cribben & Yu, 2017; Delvenne et al., 2010) such as size growths and decays, merges and splits of communities, etc.

For multivariate time series, the change-point detection task has been widely studied in various settings, e.g., nonparametric (Zou et al., 2014), high-dimensional (Wang & Samworth, 2018), and online (Wang et al., 2022). The equivalent task for dynamic networks, termed *network change-point detection* (NCPD), has recently become a popular problem. It has notably been applied to financial networks (Barnett & Onnela, 2016), brain data (Ofori-Boateng et al., 2019), and transport systems (Yu et al., 2021). In this work, we introduce the NCPD task in the discrete-time representation of temporal networks called *snapshot networks*. We denote a dynamic network $\mathcal{N}_I = \{G_t\}_{t \in I}$ to be a sequence of *graph snapshots*, where I is an ordered set (for instance, $\mathbb{N}_{>0}$), and for each $t \in I$, G_t, is a (static) graph, which can be directed, have edge weights or node attributes.

For simplicity, I is chosen as $\mathbb{N}_{>0}$ and we define a change-point for the network $\mathcal{N}$ as a timestamp $t \in I$ such that the distribution of the graphs before t, e.g., $(G_1, \dots, G_{t-2}, G_{t-1})$, is significantly different from the distribution of graphs observed from t, e.g., $(G_t, G_{t+1}, \dots)$. To better formalise the concept of change-point, we assume that each snapshot G_t of the dynamic network is independently generated from a random graph distribution $\mathcal{G}_t$. For instance, $\mathcal{G}_t$ can be an inhomogeneous Bernoulli network distribution (Wang et al., 2021; Yu et al., 2021), a stochastic block model (Bhattacharjee et al., 2020a), or a graphon model (Zhao et al., 2019). We call $\mathcal{G}_t$ the unknown *generative distribution* of G_t and $\tau \in I$ a change-point if $\mathcal{G}_\tau \neq \mathcal{G}_{\tau-1}$. In general, a dynamic network sequence may contain multiple change-points $\tau_1 < \tau_2 < \cdots$, corresponding to multiple distribution changes.

In this work, we aim at detecting and localising change-points in a online setting, i.e., as soon as the graph snapshots are collected. Note that in an offline analysis, one aims at detecting change-points a-posteriori, i.e., after the whole data sequence has been observed. For particular graph generative models, the feasibility of the NCPD task and minimax rates of estimation have been analysed in dynamic random graph models for undirected, unweighted, and unattributed graphs, e.g., Bernoulli networks (Enikeeva & Klopp, 2021; Padilla et al., 2019; Wang & Samworth, 2018; Yu et al., 2021), graphon models (Zhao et al., 2019), stochastic block models (Wang et al., 2013; Wilson et al., 2019), and generalized hierarchical random graphs (Peel & Clauset, 2015). However, many real-world dynamic networks have heterogeneous properties, such as edge weights, node attributes, and nonlinear dynamics (Li et al., 2017), which cannot be handled by existing model-based methods. Moreover, specifying the generative distributions of a dynamic network, and the type of change that can occur, can be too restrictive in practice (Wang et al., 2017).

In contrast, model-free approaches for NCPD often rely on a discrepancy measure between two subsets of graphs, leveraging a graph similarity function, a graph kernel, or a distance to perform pairwise graph comparisons (Chu & Chen, 2018; Cribben & Yu, 2017; Gretton et al., 2008; Zhao et al., 2019). Nonetheless, it is often difficult to make an adequate choice of such metric, while being agnostic to the generating mechanism or type of change-point. In fact, without any domain knowledge, an arbitrary choice can lead to poor performances (Chu & Chen, 2018; Enikeeva & Klopp, 2021; Kriege et al., 2020). Besides, most online NCPD methods require finely tuning several hyperparameters, such as detection thresholds (Yu et al., 2021) and window sizes (Huang et al., 2020).

To address these challenges, we propose a change-point agnostic method that performs online NCPD and includes learning a data-driven graph similarity function. Our approach is therefore adaptive to the network distribution and different types of change-points. In particular, it can easily incorporate general graph features such as node attributes, edge weights or attributes, and can handle sparse settings. In summary, our contributions are the following:

- We propose an online NCPD algorithm based on an efficient similarity statistic, with a data-driven graph similarity function and a short-term history of graph snapshots;
- We design a graph similarity learning algorithm using a siamese graph neural network (s-GNN) with a parsimonious architecture, and an adequate training procedure. In particular, our s-GNN is sensitive to both local and global displacements in the graph structure by leveraging Sort-k pooling layers (Zhang et al., 2018), and is able to handle any available network attributes;
- We demonstrate the advantages of our data-driven similarity method for the online NCPD task on synthetic networks with diverse types of change-points, as well as on two real-world correlation networks data sets. In particular, we show that the learnt graph similarity function can be used in an efficient online NCPD statistic avoiding detection delays and requiring little additional hyperparameter tuning. We also propose a self-supervised training procedure for data sets without ground-truth labelling of change-points.

Paper outline. In Sect. 2, we review existing work on the NCPD task, then present our general setup and methodology in Sect. 3. In Sect. 4, we describe our evaluation procedure and test our method on synthetic and real-world data sets. Finally, we conclude in Sect. 5 with a summary of our findings and discuss the current limits of our methods as well as possible future developments.

2 Related works

The network change-point detection problem (NCPD) is a relatively recent area of research that has largely incorporated principles from change-point detection in time series, especially in high-dimensional settings. Most existing NCPD methods are model-based, for instance, they estimate the parameters of a network model, e.g., the generalised hierarchical random graph (Peel & Clauset, 2015), a stochastic block model (De Ridder et al., 2016), or the preferential attachment model (Bhamidi et al., 2018), and conduct hypothesis tests to detect changes in the estimated parameters. Other methods maximize a penalized likelihood function, e.g., based on a non-homogeneous Poisson point process model (Corneli

et al., 2018) or a dynamic stochastic block model (Wilson et al., 2019; Bhattacharjee et al., 2020b).

To relax the model assumptions, many model-free approaches for NCPD extract graph features and use classical discrepancy measures to quantify the amount of change between two subsets of snapshots. For instance, Miller and Mokryn (2020) use the degree distribution as the snapshots' features, while Wang et al. (2017) and Huang et al. (2020) respectively choose the joint distribution of a set of edges, and the Laplacian eigenvectors as graph features. For directly comparing pairs of snapshots, several graph similarity functions, (pseudo)-distances, and kernels, have been used, such as the DeltaCon metric (Koutra et al., 2016), the Hamming distance (Donnat & Holmes, 2018), the Frobenius distance (Barnett & Onnela, 2016), the Laplacian spectral distance (Cribben & Yu, 2017; Hewapathirana et al., 2020), the ℓ_2 or ℓ_∞ distances (Zhao et al., 2019), and graph kernels (Desobry et al., 2005; Gretton et al., 2008; Harchaoui et al., 2009). Nevertheless, these graph metrics suffer from intrinsic limitations and can be sensitive to the graph density (Donnat & Holmes, 2018). Furthermore, Barnett and Onnela (2016) underline that the choice of graph distance can significantly affect the output of a method, and therefore this choice requires a-priori knowledge on the network distribution.

Another widely popular method in change-point detection problems is the cumulative sums (CUSUM) statistic, which has been used in different time series contexts, e.g., in the offline and high-dimensional setting (Wang et al., 2022), and more recently, in the online setting (Wang et al., 2022). Several NCPD methods have adapted this efficient statistic to dynamic network sequences, e.g., with sparse snapshots (Wang & Samworth, 2018) or missing links (Dubey et al., 2021; Enikeeva & Klopp, 2021), in offline (Padilla et al., 2019) and online (Yu et al., 2021) settings. In some specific network generative models, CUSUM methods can achieve minimax rates of estimation for the overall false alarm probability and the detection delay. However, these algorithms necessitate a *forward* window to detect a change at a given timestamp, and often require to tune several hyperparameters, e.g., one or several detection thresholds.

In addition to the aforementioned limitations, most previously cited methods do not provide a principled way to incorporate node attributes or even edge weights. To the best of our knowledge, no prior work has ever considered graph neural networks (GNNs) for the NCPD problem, despite the fact that such architectures can easily handle different types of networks (e.g., signed or directed), and can inherently account for any available node attributes (Kipf & Welling, 2016). In fact, dynamic graph neural networks have been leveraged in a varied range of tasks in dynamic network modelling, e.g., sequence predictions (Manessi et al., 2020; Seo et al., 2018), dynamic link prediction task (Rossi et al., 2020; Sankar et al., 2020; Trivedi et al., 2019), and anomalous edge detection (Cai et al., 2021). Interestingly, Zhang et al. (2020) incorporate GNN layers in a deep-learning method for change-point detection in multivariate time series to encode the cross-covariances between the time series dimensions. In this context, the GNN is only one part of a complex neural network architecture where the temporal dependencies are encoded by recurrent neural network layers.

In contrast, we propose to leverage a static GNN in an online NCPD method. In fact, GNNs can be designed to learn graph similarity functions in a data-driven way and for particular tasks in an end-to-end fashion. This type of methods, called *graph metric learning* or *graph similarity learning (GSL)* (Ma et al., 2021), has notably been shown to improve performance in graph classification tasks (Ktena et al., 2017; Liu et al., 2019; Yoshida et al., 2021; Zhao & Wang, 2019). Common types of models for GSL are siamese graph neural networks (Ma et al., 2019) and graph matching networks (Li et al., 2019; Ling et al.,

2021), and allow to learn flexible and adaptive similarity functions for downstream tasks. In our method, we build a GSL model for the online NCPD task, which notably avoids the need for choosing a-priori a particular graph distance, kernel, or embedding.

Finally, we note that the network change-point detection task shares some links with graph anomaly detection. However, the latter is often considered in the static setting, and consists in finding anomalous nodes, edges, or subgraphs in a single graph. In the dynamic network setting, an *event or change detection* can be considered as a dynamic graph anomaly (Ranshous et al., 2015), although it is most often the case when the change affects only one snapshot and is not sustained over a time period. In comparison, the network change-point detection tasks aims at finding the *timestamps* at which the sequence of snapshots undergoes a *sustained* change.

3 General set-up and method

In this section, we present our NCPD method based on a graph similarity learning algorithm. We start in Sect. 3.1 by introducing our general inference set-up and our network change-point detection statistic. The latter leverages a graph similarity function learnt by a s-GNN model described in Sects. 3.2 and 3.3. Finally, our training and validation procedures are described in Sect. 3.4. Before presenting our methodology, we introduce some useful notation.

Notation. We denote $G = (\boldsymbol{A}, \boldsymbol{X}) \in \mathbb{G}$ a graph with $n \geq 1$ nodes, adjacency matrix $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ and node attributes (or features) matrix $\boldsymbol{X} \in \mathbb{R}^{n \times d} \cup \{\emptyset\}$, with $d \geq 1$ attributes. We say that the graph is *attributed* if $\boldsymbol{X} \neq \emptyset$, and *unattributed* otherwise. If $\boldsymbol{A} \in \mathbb{R}_{\geq 0}^{n \times n}$, we also say that the graph is unsigned.

Let $\boldsymbol{I}_n$ and $\mathbb{1}_n$ be respectively the $n \times n$ identity matrix and the all-one vector of size n. For a matrix $\boldsymbol{M}$, we denote $\boldsymbol{M}_{ij}$ an entry, $\boldsymbol{M}_{i:}$ its i-th row and $\boldsymbol{M}_{:j}$ its j-th column. We also denote $\|\boldsymbol{M}\|_F$ and $\|\boldsymbol{M}\|$ respectively the Frobenius norm and operator norm (i.e., the largest singular value). For a vector $\mathbf{v}$, we denote by $\|\mathbf{v}\|$ its Euclidean norm. For any positive integer J, let $[J]$ denote the set $\{1, 2, \ldots, J\}$.

3.1 Similarity-based network change-point detection

We consider a dynamic network $\mathcal{N}_T = \{G_i\}_{1 \leq t \leq T}$ with $T \geq 1$ snapshots, and an unknown number of change-points $\tau_0 = 1 < \tau_1 < \cdots < \tau_K < T$, $K \geq 1$, such that, for any $k \in [K]$ we have

$$G_i \overset{i.i.d.}{\sim} \mathcal{G}_{k-1}, \quad \tau_{k-1} \leq i < \tau_k, \tag{1}$$

where $(\mathcal{G}_0, \ldots, \mathcal{G}_K)$ are distinct graph generating distributions. Moreover, we assume that the set of nodes in each graph snapshot G_t is fixed, of size n, and its ordering in the adjacency matrices $(\boldsymbol{A}_t)_{1 \leq t \leq T}$ (and thus, the node attributes matrices $(\boldsymbol{X}_t)_{1 \leq t \leq T}$) is kept unchanged along the sequence. We also assume a minimal spacing between two consecutive change-points, i.e., $\forall k \geq 1, \tau_k - \tau_{k-1} \geq L_0$, with $L_0 > 0$ a known constant. We note that the i.i.d. assumption in (1) is a strong hypothesis, which, in practice, may not be verified, since consecutive snapshots of real-world dynamic networks are often correlated. However, this set-up is standard for deriving theoretical results on NCPD methods in dynamic random graph models (see for instance Bhattacharjee et al., 2020b; Wang & Samworth,

2018; Yu et al., 2021; Zhao et al., 2019). In this work, we consider this set-up for clarity of exposition, nevertheless, our method partially accounts for the possibly existing correlations between the snapshots in the design of the snapshot sampling scheme (see Sect. 3.4).

Assume for now that we have at our disposal a graph similarity function $s : \mathbb{G} \times \mathbb{G} \to [0, 1]$ that verifies, for any $t_1, t_2 \in [T]$, with $\mathcal{G}_{i_1}, \mathcal{G}_{i_2} \in \{\mathcal{G}_i\}_{i=0}^K$ and $G_{t_1} \sim \mathcal{G}_{i_1}, G_{t_2} \sim \mathcal{G}_{i_2}, i_1, i_2 \in \{0, 1, \dots, K\}$,

$$s(G_{t_1}, G_{t_2}) \begin{cases} > 0.5 & \text{if } \mathcal{G}_{i_1} = \mathcal{G}_{i_2}, \\ \leq 0.5 & \text{otherwise.} \end{cases} \tag{2}$$

Then, we can use the similarity function s to classify pairs of snapshots from $\mathcal{N}_T$ as not being separated by at least one change-point (label '1'), or not (label '0'). In other words, the pair similarity score $s(G_{t_1}, G_{t_2})$ can be interpreted as a pseudo-probability of G_{t_1} and G_{t_2} of belonging to the same segment $[\tau_i, \tau_{i+1})$, for some $i \leq K$, and therefore, of having the same generating distribution. To detect change-points in $\mathcal{N}_T$, we can then monitor the following average similarity statistic

$$Z_t(s, L) = \frac{1}{L} \sum_{i=1}^{L} s(G_t, G_{t-i}), \quad t \geq L, \tag{3}$$

where $L < L_0$ is a hyperparameter that controls the length of the past (or *backward*) window. Note that for $L = 1$, $1 - Z_t(s, L)$ corresponds to a dissimilarity score that can be used in the context of dynamic graph anomaly detection (Ranshous et al., 2015). However, for the NCPD task, using a window size $L > 1$, and thus averaging over multiple evaluation of s, allows to better estimate the similarity between the current snapshot and the ones in its recent past.

From (2) and (3), we define the following online detection rule. For any timestamp t, if

$$Z_{t'}(s, L) > 0.5, \quad t - L \leq t' < t, \quad \text{and} \quad Z_t(s, L) \leq 0.5, \tag{4}$$

then, we state that t is a change-point. Note that in (4), the detection threshold is chosen as 0.5 to match the classification property (2) of s. Nonetheless, if the similarity function does not fully verify this property, and in order to have more flexibility on the detection rate of the algorithm, one could replace 0.5 by a threshold θ in (4). The latter would then be a hyperparameter of our method, requiring a suitable validation method such as the ones used by Ranshous et al. (2015) or Cribben and Yu (2017). In our experiments, we consider the simpler version, which avoids any additional hyperparameter tuning procedure.

Using (4), we can then detect an arbitrary number of change-points in an online setting, and without any detection delay, i.e., as soon as the snapshots are observed. However, the properties and performance of such approach heavily depend on the similarity function s and its discriminative power. Our main contribution consists of *learning* the graph similarity function from a training sub-sequence of the dynamic network using a siamese graph neural network model, which we describe in the next section.

Remark 1 The hyperparameter L tuning the window length of the statistic (3) controls the minimal spacing between two change-points that can be detected by our method. We note that a too small value is likely to provide a noisy estimate of the similarity between the current graph G_t and the previous snapshots. Therefore, the choice of this hyperparameter implies a trade-off between the temporal granularity of the detection algorithm and its robustness for estimating an average similarity score. In practice, its value can be chosen

using domain knowledge regarding the frequency of change-points in the data, or validated like other hyperparameters of our method. In our numerical experiments, we test different values and observed on simulated data that our method is not very sensitive to this hyperparameter, as soon as the graph similarity function has good discriminative power. Therefore, a relatively small L (and history of data) can be used in practice, e.g., $L = 6$ in our synthetic experiments in Sect. 4.3.

Remark 2 Our learning approach for the graph similarity function can also be employed in the *offline* setting of NCPD, with a slight change of the detection rule. In this context, one aims at localising changes in a dynamic network $\mathcal{N}_T$ after the whole sequence has been collected. For instance, for a network with a single change-point, one can localise the latter at $\hat{\tau}$, such that

$$\hat{\tau} = \arg\min_{L \le t \le T} Z_t(s, L), \quad \text{or} \quad \hat{\tau} = \arg\max_{L+1 \le t \le T} |Z_t(s, L) - Z_{t-1}(s, L)|. \tag{5}$$

Additionally, our method could be adapted to a setting where a small detection delay (e.g., of order L) may be tolerated. In this case, we could replace (3) by a change-point statistic that also uses the snapshots in the *forward* window, e.g., $(G_t, G_{t+1}, \dots G_{t+L})$, for instance, a two-sample test statistic such as the maximum kernel Fisher discriminant ratio (Harchaoui et al., 2009) or the maximum mean discrepancy (Gretton et al., 2008). In comparison to (3), these statistics are likely to be more robust to any noise in the observations.

3.2 Graph similarity learning via siamese graph neural networks

In this section, we describe our siamese graph neural network model for learning the pairwise similarity function s. In this deep-learning model, the input is a pair of graphs, say (G_1, G_2), and, in the first model block, G_1 and G_2 are encoded with the same graph encoder (or equivalently, two *siamese* encoders sharing the same weights). Then, the two graphs' embeddings are combined in the second block, the symmetric similarity module, which output is the pair similarity score. The variability of s-GNN models mainly lies in the design of the graph encoder and similarity module [see for instance different approaches in Ktena et al. (2017), Ling et al. (2021), and Ma et al. (2019)]. Our proposed architecture for NCPD is represented in Fig. 1. Note that its input will consist of pairs of graph snapshots from the dynamic network.

For the sake of simplicity, we use a simple graph convolutional network (GCN) (Kipf & Welling, 2016) for undirected and unsigned graphs as the graph encoder in our s-GNN. However, in our architecture, this block is generic and can be replaced by any ad-hoc graph encoder, e.g., a graph attention network (Veličković et al., 2018), a GraphSage network (Hamilton et al., 2017), or a graph isomorphism network (Xu et al., 2019). In a GCN, the embedding of a graph $\boldsymbol{H}^{(j)}$ at each layer $j \in [J], J \ge 1$ is computed as follows

$$\boldsymbol{H}^{(j)} = \sigma\left(\tilde{\boldsymbol{A}}\boldsymbol{H}^{(j-1)}\mathbf{W}^{(j)} + \mathbf{B}^{(j)}\right), \tag{6}$$

where $\mathbf{W}^{(j)} \in \mathbb{R}^{h_{j-1} \times h_j}$ is a weight matrix, h_j, h_{j-1} are the numbers of hidden units of layers j and $j-1$, $\mathbf{B}^{(j)} \in \mathbb{R}^{h_j}$ is a bias vector, $\tilde{\boldsymbol{A}} = \tilde{\boldsymbol{D}}^{-1/2}(\boldsymbol{A} + \boldsymbol{I}_n)\tilde{\boldsymbol{D}}^{-1/2}$ is the normalized augmented adjacency matrix with degree matrix $\tilde{\boldsymbol{D}} = \mathrm{Diag}\left((\boldsymbol{A} + \boldsymbol{I}_n)\mathbb{1}_n\right)$, and σ is the pointwise ReLU activation function, i.e., $\sigma(x) = \max(x, 0)$. Note that each row $\boldsymbol{H}_i^{(j)}$ corresponds to the embedding of node i at layer j. The input of the first layer of the GCN, denoted $\boldsymbol{H}^{(0)}$, is either the node attributes matrix $\boldsymbol{X} \in \mathbb{R}^{n \times d}$ if the input graph is attributed or a positional

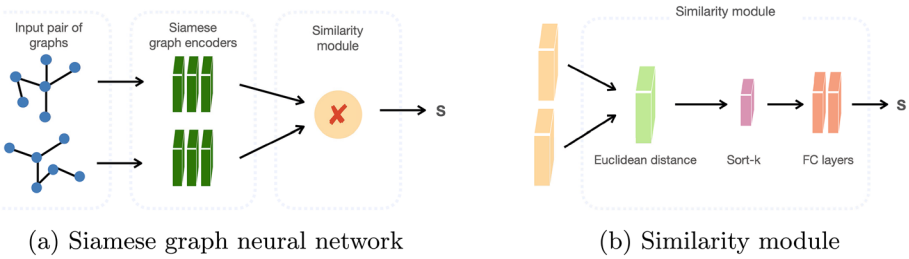


Fig. 1 Architecture of our graph similarity learning model. The general pipeline (**a**) is a siamese GNN, which second block is a similarity module (**b**). The latter uses node-wise Euclidean distances, a Sort-k pooling operation, and fully-connected layers, for computing the similarity scores of snapshots in dynamic networks

encoding matrix (see Sect. 3.3). Moreover, the block's output is a node-level embedding matrix $\boldsymbol{H}^J \in \mathbb{R}^{n \times h_J}$ from the last layer. Therefore, for any pair of snapshots (G_{t_1}, G_{t_2}), the graph encoder computes a pair of embeddings, $(\boldsymbol{H}_1, \boldsymbol{H}_2) := (\boldsymbol{H}^J(G_{t_1}), \boldsymbol{H}^J(G_{t_2}))$, where each row vectors $(\boldsymbol{H}_1)_{i:}$ and $(\boldsymbol{H}_2)_{i:}$ correspond to the representations of node $i \in [n]$, respectively in G_{t_1} and G_{t_2}.

Then, the pair of embeddings $(\boldsymbol{H}_1, \boldsymbol{H}_2)$ is processed by our similarity module, described in Fig. 1b. This module consists of a Euclidean distance operation, a *Sort-k* pooling layer (Zhang et al., 2018), and two fully-connected layers. The *Sort-k* pooling operation selects and orders the k largest entries of the input, where $k \geq 1$ is a hyperparameter tuning the *receptive field* of this layer. Note that the pooling layer is applied to the n-dimensional vector of row-wise Euclidean distances between the embeddings $\boldsymbol{H}_1$ and $\boldsymbol{H}_2$, therefore, it implies a subset selection of nodes having the largest displacement between their representations in the two snapshots. More precisely, the output of the pooling operation is given by

$$\boldsymbol{P} = (f_{r_1}, \ldots, f_{r_k}), \quad f_i = \|(\boldsymbol{H}_1)_{i:} - (\boldsymbol{H}_2)_{i:}\|_2, \quad 1 \leq i \leq n,$$

where $r_1, \ldots, r_k$ correspond to the indices of the (sorted) k largest elements in $\{f_i\}_{i \in [n]}$. Intuitively, a large distance f_i can indicate that the i-th node plays distinct structural roles in G_{t_1} and G_{t_2}. Besides, we note that the Euclidean distance operation in our similarity module could be also replaced by another standard distance, similarity, or kernel function such as the cosine similarity or a Gaussian kernel.

Finally, the pooled vector $\boldsymbol{P}$ is processed by two fully connected layers, each of them containing an affine transformation, a batch normalisation layer, and a ReLU activation function. Furthermore, the output of the second fully-connected layer is pooled using sum-pooling and transformed into a non-negative similarity score $s(G_{t_1}, G_{t_2}) \in [0, 1]$ by a sigmoid activation function. This score can be transformed into a binary label using a classification threshold of 0.5, i.e.,

$$\hat{y}(G_{t_1}, G_{t_2}) = \begin{cases} 1 & \text{if } s(G_{t_1}, G_{t_2}) > 0.5 \\ 0 & \text{otherwise.} \end{cases} \tag{7}$$

Note that a label 1 can be interpreted as the two graphs being similar, or having the same generative distribution, or, more specifically in our context, not separated by a change-point in the dynamic network sequence.

We find that using a Sort-k pooling layer in the design of the similarity module has two main advantages in our context.

• First, as previously noted, it selects the nodes that have the largest distances between their embeddings in the two graphs. Therefore, if a structural change in the dynamic network affects only a few nodes, this change can be picked up by this pooling operation, without being diminished by the absence of change in the rest of the network. In other words, if the change between two snapshots affects n_1 nodes, then, when $k < n_1$, each entry of P has a large magnitude, even if the other distances f_i may not be very large, and this should lead to a low final similarity score. In the opposite case when $k > n_1$, the output score may remain high if P does not contain many large entries. Nonetheless, in real-world networks, choosing a hyperparameter k too small may result in detecting occasional graph anomalies instead of sustained change-points. Therefore, there is a trade-off in our method between the size of the subgraph directly affected by or causing a change that can be detected, and the robustness to anomalous nodes. In practice, we tune this hyperparameter k on a validation set, in order to be flexible to the typical "size" of the change-point in the network. In fact, the change-points of interest may be localised onto a subset of nodes, for instance, if a few "important" nodes (e.g., nodes with high centrality) experience sustained change to their connectivity pattern, while the large majority of nodes and edges remain almost unchanged. Besides, this component of our similarity module could be further built upon for identifying which part of the network is mainly driving the change-point, thus enhancing the explainability of the proposed pipeline.

• Second, Sort-k pooling reduces the number of parameters in our architecture, while preserving the most important information for measuring potential and local graph changes. More generally, replacing max or sum pooling by Sort-k pooling has been proven to increase the accuracy and generalization power of neural networks, in particular in settings with limited data availability, such as one-shot learning (Horváth, 2020). Note that it can also be used for downsampling large graphs (Lee et al., 2019). In our change-point-agnostic method, this pooling layer may thus mitigate our lack of information on the change-points.

Remark 3 It is often a desirable property of GNN models with graph-level (resp. node-level) output to be invariant (resp. equivariant) to nodes' permutations, i.e., permutation of the rows (resp. rows and columns) of the node attributes (resp. adjacency) matrices of the graphs. This is due to the fact that the order of the node set in these matrices is generally arbitrary. In our method, since the s-GNN model takes as input pairs of graph snapshots from a dynamic network sequence, every graphs contain the same set of nodes with the same ordering. Therefore, in our set-up, the invariance property corresponds to the fact that the s-GNN is invariant to any permutation of the nodes that is applied on *both* inputs. More precisely, for any permutation of the node set $\sigma : [n] \to [n]$, denoting $\sigma * G$ the resulting transformation of a graph G under σ (i.e., permutation of the rows and columns of the adjacency and node attributes matrices), the invariance property writes $s(\sigma(G_1), \sigma(G_2)) = s(G_1, G_2)$. This is indeed the case for our method since the node-wise operations, i.e, the graph encoder and the Euclidean distance, are equivariant, and the Sort-k pooling layer is permutation-invariant, i.e. $\boldsymbol{P}(\boldsymbol{H}) = \boldsymbol{P}(\sigma(\boldsymbol{H}))$, therefore, so is the final similarity score.

3.3 Node encodings for unattributed dynamic networks

In this section, we present our methodology for constructing synthetic node attributes when the dynamic network is unattributed, i.e., each graph snapshot contains only structural information $G_t = (\boldsymbol{A}_t, \emptyset)$. We recall that, for attributed dynamic networks, our s-GNN described in Sect. 3.2 uses the snapshots' node attributes matrices $(\boldsymbol{X}_{t_1}, \boldsymbol{X}_{t_2})$ to initialise the features matrices $(\boldsymbol{H}^{(0)}(G_{t_1}), \boldsymbol{H}^{(0)}(G_{t_2}))$. In fact, for unattributed networks, we need to resort an appropriate initialisation of the features matrices, a choice that can be critical for the expressivity of the model (Dwivedi et al., 2022). We propose different variants of our method with several types of node encodings, i.e., synthetic node attributes that capture their relative positions in the graph structure or their specific identity. The first three types below correspond to existing techniques for general graph learning settings, and the last type is one that we believe may also be appropriate for certain NCPD tasks.

1. **Degree encoding (s-GNN-D)** (Bruna & Li, 2017) In this encoding, the attribute of a node is a scalar equal to its degree in the graph, i.e., $\boldsymbol{H}^{(0)} = \boldsymbol{A}\mathbb{1}_n \in \mathbb{R}^{n\times 1}$.
2. **Random-Walk encoding (s-GNN-RW)** (Dwivedi et al., 2022; Li et al., 2020) With $l \geq 1$, the vector of attributes of a node i is a l-dimensional vector $\boldsymbol{H}^{(0)}_{i:} = [\boldsymbol{R}_{ii}, \boldsymbol{R}^2_{ii}, \dots \boldsymbol{R}^l_{ii}]$, where $\boldsymbol{R} = \boldsymbol{A}\boldsymbol{D}^{-1}$ is the random-walk operator.
3. **Laplacian (or positional) encoding (s-GNN-PE)** (Dwivedi & Bresson, 2021) The node attributes are constructed from the principal eigenvectors of the symmetric normalised Laplacian matrix $\boldsymbol{L} = \boldsymbol{I}_n - \boldsymbol{D}^{-1/2}\boldsymbol{A}\boldsymbol{D}^{-1/2}$. More precisely, using the decomposition of the Laplacian $\boldsymbol{L} = \boldsymbol{U}^T\boldsymbol{\Lambda}\boldsymbol{U}$, where $\boldsymbol{U}, \boldsymbol{\Lambda}$ respectively contain the ordered set of eigenvectors and eigenvalues of $\boldsymbol{L}$, the Laplacian encodings are defined as $\boldsymbol{H}^{(0)} = [\boldsymbol{U}^T_{:1}, \boldsymbol{U}^T_{:2}, \dots, \boldsymbol{U}^T_{:l}] \in \mathbb{R}^{n\times l}$, where $l \geq 1$ is a chosen number of eigenvectors. Note that these attributes are similar to the ones used in spectral clustering algorithms.
4. **Identity encoding (s-GNN-I)** In this variant, we define the initial feature matrix as $\boldsymbol{H}^{(0)} = \boldsymbol{I}_n$, corresponding to a *one-hot* encoding of each node.

   We claim that this is an appropriate choice for the graph siamese encoder in our setting, although, in general, this type of encoding can break the equivariance or invariance properties of GNN models. However, this is not the case here since our s-GNN is applied to the snapshots of a dynamic network, where we have assumed that the set of nodes is constant and its global ordering, although arbitrary, is common to all graphs—note moreover that the equivariance property has a modified definition in our setting (see Remark 3) In fact, taking into account the nodes' identities in graph learning tasks can be beneficial in some tasks (Donnat & Holmes, 2018), and we believe it is particularly the case for real-world dynamic networks with a small number of nodes (see Sect. 4.4.2).

Finally, we note that there exist more complex strategies for computing node encodings, including learning procedures during the training phase of the s-GNN (Dwivedi et al., 2022). However, these approaches significantly increase the model complexity, therefore, we do not consider them in our method.

3.4 Training and validation procedures

Our s-GNN model described in Sect. 3.2 requires to be trained on a sub-sequence of the dynamic network, in order to learn the weights matrices and bias vectors of the GCN and fully-connected layers. This model can be trained in a supervised setting, when

a labelled data set of pairs of snapshots is available. In fact, if we can construct a set $\mathcal{D} = \{(G_1^i, G_2^i, y_i)\}_{1 \le i \le N}$ of N training pairs where each label $y_i \in \{0, 1\}$ indicates if the graphs G_1^i and G_2^i are similar (or *not* separated by a change-point), then, we can learn the model parameters by minimising, via gradient descent, the binary cross-entropy loss function

$$\mathcal{L}_{BCE}(\mathcal{D}, s) = \frac{1}{N} \sum_{i=1}^{N} -y_i \log s(G_1^i, G_2^i) - (1 - y_i) \log(1 - s(G_1^i, G_2^i)).$$

We now describe our constructions of such training and validation sets, from sub-sequences of the dynamic network containing ground-truth change-points.

To construct a set of labelled pairs of graphs $\mathcal{D} = \{(G_1^i, G_2^i, y_i)\}_{1 \le i \le N}$, we first divide the sequence of graph snapshots into training, validation and test sub-sequences, e.g., using consecutive windows of respectively 60%, 20% and 20% timestamps. We assume that the resulting sub-sequences contain at least one ground-truth change-points. Then, we propose sampling and labelling schemes of pairs of snapshots from the training and validation sub-sequences.

1. **Random scheme (training set)** we consider the set of all (non-ordered) pairs of graphs in the training sequence and label each pair (G_1^i, G_2^i) with $y_i = 1$ if there is no change-point between t_1 and t_2, and $y_i = 0$ otherwise. Then we uniformly sample a fixed number of pairs with label 1 (the *positive* pairs) and the same number of pairs with label 0 (the *negative* pairs), without replacement. The number of pairs N is chosen heuristically between T and $10 \times T$ in our experiments.
2. **Windowed scheme (validation set)** we consider the set of all (non-ordered) pairs of graphs in the network sequence that are not distant from each other by more than L timestamps, and label them with the same procedure as in the **Random scheme**.

We note that the different sampling mechanisms for the pairs in the training and validation sets are designed to satisfy a double objective of our learning procedure. In fact, we simultaneously aim to learn an adequate graph similarity function and to detect change-points in a dynamic network sequence using the latter.

For the first objective, the **Random scheme** allows to sub-sample pairs of graph snapshots that are further away in the sequence. This design can mitigate two possible undesired effects in real-world dynamic networks: on the one hand, the possible temporal correlations between the snapshots in each pair and between the pairs themselves; on the other hand, transition phenomena, i.e., gradual changes, between two generative distributions. Additionally, the **Random scheme** avoids label imbalance in the training set, since we can sub-sampling the same number of positive and negative pairs, which we assume is favorable for the learnt similarity function.

For the second objective, the **Windowed scheme** builds a validation set of pairs that is more similar to the test setting of our model. In fact, in our change-point statistic (3) and detection rule (4), the graph similarity function s is only evaluated on pairs of graphs within a sliding window of size L. In particular, these pairs of graphs may be highly correlated, and are more likely to be *positive* pairs than *negative* ones, since there are generally only few change-points in the dynamic network. We finally note that in both the **Random** and **Windowed schemes**, the sampled pairs have in common at most one graph snapshot.

Therefore, in a supervised NCPD setting, we can sample training and validation sets to learn the (hyper) parameters of our s-GNN model using the previous sampling strategies.

In an unsupervised NCPD setting, i.e., when the dynamic network does not contain any ground-truth label of change-point, we need to resort to a novel ad-hoc self-supervised learning technique (Liu et al., 2021). In this case, we first pre-estimate a set of change-points in the training and validation sequences using a spectral algorithm, then apply the previous sampling schemes to draw training and validation pairs of graphs (see more details in Sect. 4.4.1 where this strategy is applied to the financial network data set).

4 Numerical experiments

In this section, we test and evaluate the performances of our s-GNN method in the online NCPD task, first, in a controlled setting of synthetic dynamic networks (Sect. 4.3), then, on two real-world correlation networks (Sects. 4.4.1 and 4.4.2).[1]

4.1 Performance metrics

For dynamic network data sets with ground-truth labels of change-points, we evaluate the performance of NCPD methods using the following metrics.

- **Localisation error (single change-point settings)**. In the case of a unique ground-truth change-point τ and estimate $\hat{\tau}$ (see the synthetic settings in Sect. 4.3.1), the localisation error metric is defined as $\text{Error}_{\text{CPD}} = |\hat{\tau} - \tau|$.
- **Adjusted F1-score (multiple change-points settings)** . In this setting, we use a classification metric of timestamps as change-point (label 1) or not change-point (label 0), with a tolerance level t. Note that in the context of change-point detection, the labels of timestamps differ from our labelling scheme of pairs of graphs in Sect. 3.4, where 1 corresponds to the label for "similar" pairs. Note moreover that for a value t of the tolerance level, all the timestamps within a window of length $2t + 1$ centered at the ground-truth change-points are considered as ground-truth label 1 for this metric, and a valid detection occurs whenever one of these timestamps is classified as change-point (Xu et al., 2018).

For the the financial data set which has no ground-truth labels, we qualitatively discuss our findings in Sect. 4.4.1, frame them in a financial context, and compare them with previous analysis of similar data. Additionally, in the synthetic data experiments in Sect. 4.3, we also evaluate the accuracy of our graph similarity function s for discriminating between graphs sampled from the same or different distributions, i.e., for classifying pairs of graphs generated from either the same or different random graph models.

Remark 4 In the multiple change-point setting, other performance metrics could be used, for instance, metrics taking into account the distance to the ground-truth change-points such as the Adjusted Rand Index (ARI). The latter measures the similarity between the partition of the observation window $[1, T]$, using respectively the detected and ground-truth change-points. We have also tested the ARI metric in our synthetic experiments and obtained similar performances than with the adjusted F1-score (see Appendix A.4).

[1] The implementation of our method is available at https://github.com/dsulem/DyNNet.git.

4.2 Baselines

In our experiments, we compare our NCPD method to existing algorithms in two ways. On the one hand, we compare our data-driven graph similarity function to graph distances, similarity function, and graph kernels previously used in the context of NCPD and graph two-sample-test, when used within our change-point statistic (3).

- **Frobenius distance** (Barnett & Onnela, 2016; Dubey et al., 2021; Nie & Nicolae, 2021), defined as $d_F(\boldsymbol{A}, \boldsymbol{B}) = \|\boldsymbol{A} - \boldsymbol{B}\|_F$, for two matrices $\boldsymbol{A}, \boldsymbol{B}$ with equal dimensions. We apply this distance to the adjacency matrices of two graph snapshots. We note that Bao et al. (2018) apply the Frobenius distance on the graph Laplacian matrices. Moreover, this distance has also been used in a minimax testing perspective between two graph samples by Ghoshdastidar et al. (2020).
- **Procrustes distance** (Hewapathirana et al., 2020). This distance corresponds to the Frobenius distance between the matrices of k principal eigenvectors of the symmetric graph Laplacian $\boldsymbol{L} = \boldsymbol{I}_n - \boldsymbol{D}^{1/2}\boldsymbol{A}\boldsymbol{D}^{1/2}$, after performing an alignment step. The number of eigenvectors k can be pre-specified or chosen by finding the optimal low-rank approximation of $\boldsymbol{L}$.
- **DeltaCon similarity** (Koutra et al., 2016). This graph similarity function is based on the Matusita distance applied to the Fast Belief Propagation graph operators, defined for a graph as $\boldsymbol{S} = [\boldsymbol{I}_n + \epsilon^2\boldsymbol{D} - \epsilon\boldsymbol{A}]^{-1}$ with $\epsilon > 0$. We use the implementation of this similarity function provided in the Python package `netrd`.[2]
- **Weisfeiler–Lehman (WL) kernel** (Shervashidze et al., 2011).
  This graph kernel is notably used in the two-sample-test problem for sets of graphs (Gretton et al., 2008). We use the implementation from the GraKel Python package (Siglidis et al., 2020), and fix the number of iterations of the WL kernel algorithm to 5 in our experiments.

We note that for the previous baselines, we tune the threshold of our change-point detection rule on the training and validation sub-sequences. On the other hand, we compare our NCPD algorithm to methods that do not rely on an explicit graph metric for detecting change-points.

- **Network change-point detection with spectral clustering (SC-NCPD)** (Cribben & Yu, 2017). This method first partitions the node set of each snapshot with a spectral clustering algorithm, then computes an inner product between averages of spectral features across a backward and a forward windows. In this method, the number of clusters and the lengths of the windows are pre-specified.
- **Laplacian anomaly detection (LAD)** (Huang et al., 2020). This method applies both to the anomaly detection and change-point detection tasks for dynamic networks, and is based on the anomaly score

$$Z_t = 1 - |\tilde{\sigma}_t \sigma_t|,$$

  where $\sigma_t \in \mathbb{R}^\ell$ is the vector of ℓ largest singular values of the unormalized Laplacian of the graph G_t and $\tilde{\sigma}_t \in \mathbb{R}^\ell$ aggregates the ℓ largest singular values of each snapshots

[2] https://netrd.readthedocs.io/.

in a past window of size L, i.e., $(\sigma_{t-L}, \dots, \sigma_{t-1})$The number of singular values k and the length of the window are pre-specified hyperparameters.

- **Network cumulative sums statistic (CUSUM)** (Yu et al., 2021). This method uses a backward and a forward windows of sizes L' to compute a sequence of CUSUM matrices

$$\boldsymbol{C}_t = \frac{1}{\sqrt{2L'}}\left(\sum_{s=t-L'+1}^{t} \boldsymbol{A}_s - \sum_{s=t+1}^{t+L'} \boldsymbol{A}_s\right), \quad L' \le t \le T - L'. \tag{8}$$

  Following the methodology in Yu et al. (2021), we divide the dynamic network into two samples, $N_A = \{G_{2t}\}_{1 \le t \le T/2}$ and $N_B = \{G_{2t-1}\}_{1 \le t \le T/2}$, containing the snapshots respectively at even and uneven timestamps. This algorithm monitors two statistics based on the CUSUM matrices (8) of these samples: the Frobenius norm of the Universal Singular Value Threshold (USVT) estimator $\widetilde{B}(t)$ of the CUSUM matrix computed from N_B, and the dot product between $\widetilde{B}(t)/\|\widetilde{B}(t)\|$ and the CUSUM matrix computed from N_A. To avoid tuning the additional threshold parameters, we do not apply the USVT step (or equivalently choose $\tau_1 = 0$ and $\tau_2 = 1$ in USVT). Moreover, we only use the second statistics since the first one is very close to the next baseline.
- **Operator norm of network CUSUM (CUSUM 2)** (Enikeeva & Klopp, 2021). We adapt this offline method to the online problem by computing the CUSUM matrix over a past and future windows of size L'.

  The NCPD statistics is then $z_t = \|\boldsymbol{C}_t\|$.

For these baselines, we fix the number of clusters or singular values to $k = 6$ and the size of windows to $L' = L/2$ when both the past and future are used in the NCPD statistic. We also note that these methods are applied to non-attributed dynamic networks and therefore only use the sequence of adjacency matrices $(\boldsymbol{A}_t)_{t \in [T]}$. Nonetheless, in our synthetic experiments, we generate unattributed networks and only one of the real-world data set is attributed. In this case, the node attributes are ignored by the baseline methods (see Sect. 4.4.1).

4.3 Synthetic data

In this section, we evaluate our method on synthetic dynamic networks generated from dynamic stochastic block models (Bhattacharjee et al., 2020b; Padilla et al., 2019; Yu et al., 2021; Zhao et al., 2019) with one or multiple change-points, of several types.

4.3.1 Single change-point detection

In this experiment, we construct dynamic network sequences with $T = 100$ snapshots with $n = 400$ nodes, generated from a dynamic stochastic block model. In this sequence, we introduce a single change-point τ, uniformly sampled in $[T/4, 3T/4]$ such that, for each $t \in [T]$, each graph is unattributed and

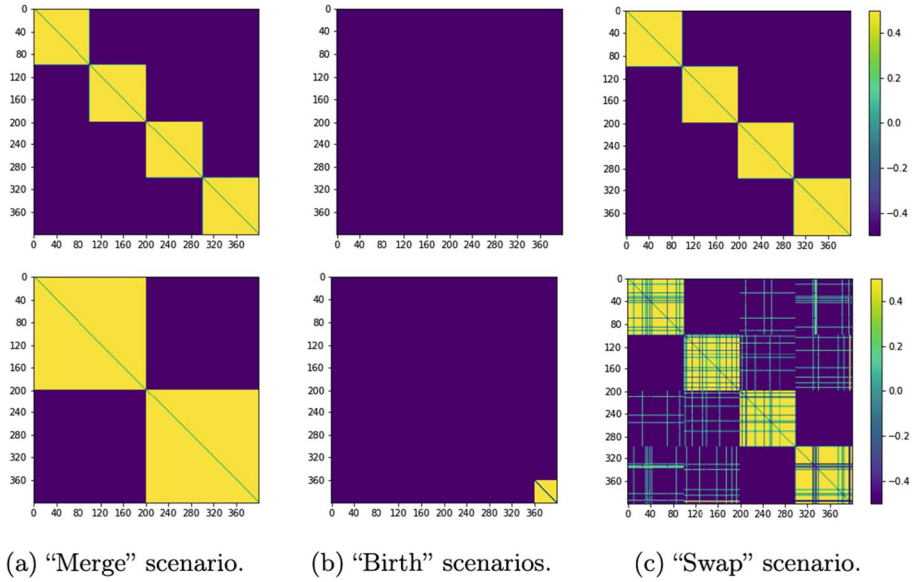


Fig. 2 Heatmaps of the adjacency matrices of the expected graph in the two stochastic block models $\mathcal{G}_1$ (first row) and $\mathcal{G}_2$ (second row), with $n = 400$ nodes, in the three main scenarios of our single change-point synthetic experiments, i.e., "Merge" (**a**), "Birth" (**b**) and "Swaps" (**c**). We recall that $\mathcal{G}_1$ and $\mathcal{G}_2$ correspond to the generative distributions of the snapshots before and after the change-point

$$G_t \overset{i.i.d}{\sim} \mathcal{G}_1, \quad \text{if } 1 \leq t < \tau,$$
$$G_t \overset{i.i.d}{\sim} \mathcal{G}_2, \quad \text{if } t \leq \tau \leq t \leq T,$$

where $\mathcal{G}_1, \mathcal{G}_2$ are the distributions of two distinct stochastic block models (SBM). We define an SBM with $K \geq 1$ communities with a connectivity matrix $\boldsymbol{C} = (p - q)\boldsymbol{I}_K + q\mathbb{1}_K\mathbb{1}_K^T$ with intra- and inter-cluster connectivity parameters $p, q \in [0, 1]$, and a cluster membership matrix $\Theta \in \{0, 1\}^{n \times K}$. The parameter p (respectively q) corresponds to the probability of existence of an edge between two nodes in the same community (respectively in two different communities), while each row Θ_i of the membership matrix has a single entry equal to 1, indicating the cluster to which the node i belongs.

We consider four different change-point scenarios, related to three possible types of events affecting the snapshots' community structure, namely the "Merge", "Birth" and "Swaps" change-points. These events imitate respectively the fusions of two communities, the appearance of a single community, and cluster switches from a subset of nodes. These scenarios are illustrated in Fig. 2, where we plot the heatmaps of the expected adjacency matrices in the two models $\mathcal{G}_1$ and $\mathcal{G}_2$.

- **Scenario 1 ("Merge").** In this scenario, the two SBMs $\mathcal{G}_1$ and $\mathcal{G}_2$ have respectively four and two equal-size clusters, and parameters $q = 0.02$ and $p \in [0.25, 0.1]$. We note that the larger p is, the easier the change-point detection problem.
- **Scenario 2 ("Birth 1").** In this scenario, $\mathcal{G}_1$ is the distribution of an Erdös–Renyi model with edge probability $q = 0.03$ and $\mathcal{G}_2$ is a SBM with two communities of size $n - s$ and s, $s \in [40, 100]$, and connectivity matrix

$$C = \begin{pmatrix} q & q \\ q & p \end{pmatrix},$$

with $p = 0.1$.

We note that the bigger the size s of the denser cluster is, the easier the change-point detection problem.

- **Scenario 3 ("Birth 2").** This scenario uses the same type of change-point as Scenario 2 but in this case, we fix the size of the denser cluster $s = 100$ and we vary the intra-cluster edge probability $p \in [0.05, 0.2]$.

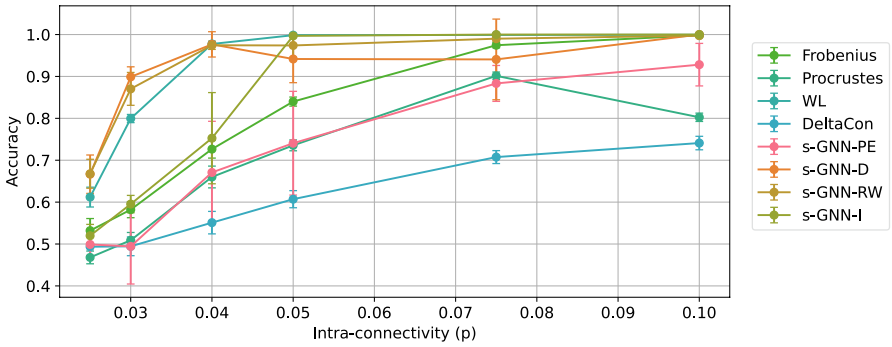
  We note that similarly to Scenario 1, the larger p is, the easier the change-point detection problem.
- **Scenario 4 ("Swaps").** In this scenario, the two SBMs $\mathcal{G}_1$ and $\mathcal{G}_2$ have the same connectivity matrices with $K = 4$ equal-size clusters, $p = 0.1$ and $q = 0.05$, but their cluster membership matrices differ. In the second SBM, a proportion $h \in [0.02, 0.2]$ of pairs of nodes from the first SBM exchange their community memberships, i.e., "swap" their corresponding row-vectors in θ. We note that the bigger the proportion h, the easier the change-point detection problem.

We note that in all settings, the graph snapshots are relatively sparse, and that Scenario 1 can be considered as a *global* change of the network structure, while the other scenarios correspond to a *local* topological change, i.e., localised on a subset of nodes. In each scenario and set of parameters, we test our change-point detection method on 50 dynamic network sequences. Moreover, to train, validate, and evaluate our s-GNN model on the classification task, we independently generate 1000 labelled pairs of graphs (G_1^i, G_2^i, y_i), where for each $i \in [1000]$, $G_1^i \sim \mathcal{G}_k, G_2^i \sim \mathcal{G}_l$, with $k, l \in \{1, 2\}$ and $y_i = 1$ if $\mathcal{G}_k = \mathcal{G}_l$ and $y_i = 0$ otherwise. In the latter data sets, we use respectively 60%, 20% and 20% of the pairs for training, validating and testing. Additional details on this experimental setting can be found in Appendix A.

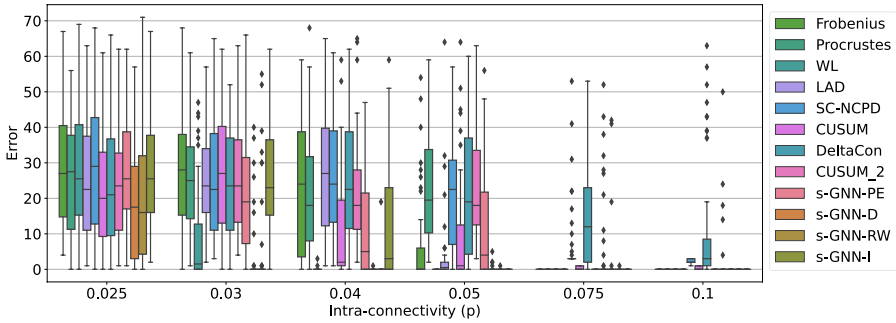
We also test four variants of our s-GNN model, with **Degree (s-GNN-D)**, **Random Walk (s-GNN-RW)**, **Laplacian (s-GNN-PE)**, and **Identity (s-GNN-I)** node encodings, as defined in Sect. 3.2. Moreover, in the NCPD task, for all methods including the baselines, we estimate the unique change-point with the detection rule (5), and use a window size $L = 6$. Our results for the classification and change-point detection tasks in each scenario are reported Figs. 3, 4, 5, and 6.

We note that in almost all scenarios and settings, all variants of our method, except s-GNN-PE, outperform the (non-trained) baselines, in both the classification and NCPD tasks. For s-GNN-PE, the worse performance could be attributed to the sign ambiguity in Laplacian eigenvectors (Dwivedi et al., 2022), which may result in some inconsistencies in the synthetic node attributes of the snapshots. Moreover, the Degree and Random Walk node encodings generally seem to be better than the Identity encodings, except for the Scenario 4. We conjecture that this is due to the fact that in the first three scenarios, nodes belonging the same cluster are exchangeable in the SBM model, while in the last scenario, this symmetry is broken by the membership exchange mechanism. For networks with a lot of symmetry, the **Identity** encodings might introduce additional noise.

Consequently, these experiments show that in various change-point scenarios, using a data-driven graph similarity function leads to better performances than existing baselines. We additionally observe that the performance of the strongest baselines, i.e., the **CUSUM** and **CUSUM 2** methods, improve when larger window sizes L are used, while our method



(a) Classification accuracy vs intra-connectivity parameter p.



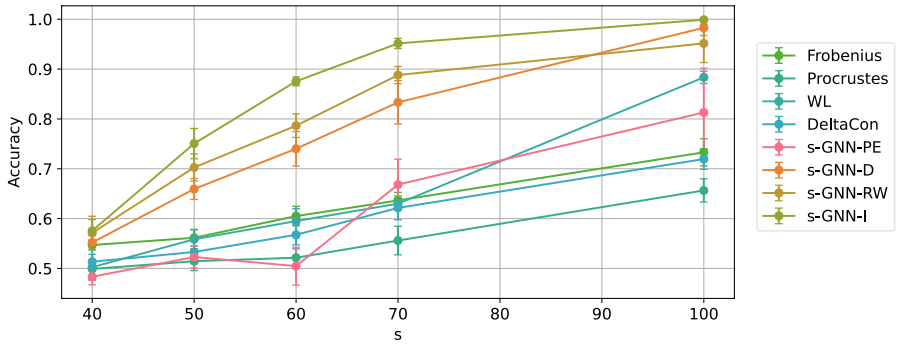
(b) Change-point localisation error for different intra-connectivity parameters p.

Fig. 3 Performance of our s-GNN method and baselines on the classification (**a**) and detection (**b**) tasks in the "Merge" scenario. In the first task, pairs of graphs sampled from the same or different SBM distributions are classified using a graph similarity function or a graph distance, therefore, the set of baselines only consists of the latter type of algorithms. In the second task, a single change-point needs to be localised in a dynamic SBM sequencem and the set of baselines include graph distance- (or kernel-) based methods and network change-point detection methods. We remark that for very large values of p, many methods attain zero error and our method achieves a smaller error for all values of p

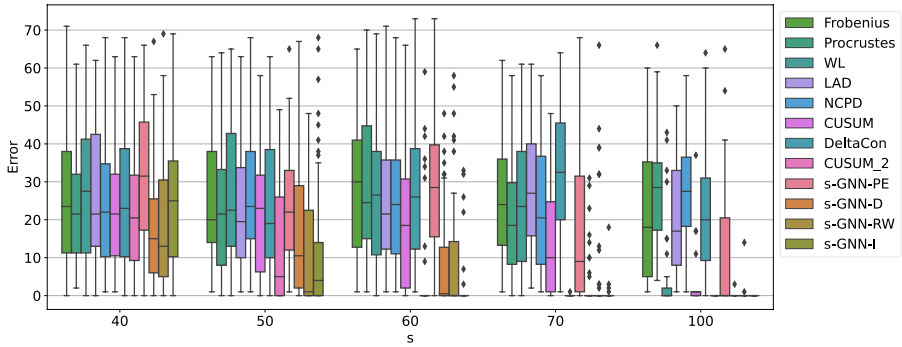
is not sensitive to this hyperparameter, as can be seen from a sensitivity analysis reported in Appendix A.2. In particular, our method performs well even when using a short history of data, and therefore could also detect change-points that are close to each other in a multiple change-point setting. In the next section, we test our method in the latter set-up.

4.3.2 Multiple change-point detection

We now consider synthetic dynamic networks with multiple change-points, corresponding to "Birth" and "Death" events of a more densely connected cluster of nodes. More



(a) Classification accuracy of pairs vs the community size s.



(b) Change-point localisation error for different community sizes.

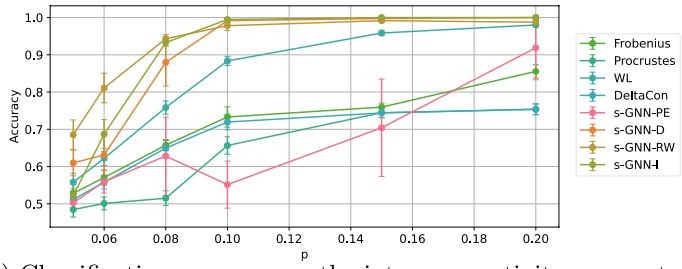
Fig. 4 Performances on the classification (**a**) and detection (**b**) tasks in the "Birth 1" scenario

precisely, we construct dynamic network sequences of length $T = 2400$, with $n = 400$ and $K = 9$ change-points $\tau_1, \ldots, \tau_K$, such that, for each timestamp $t \in [T]$, we have

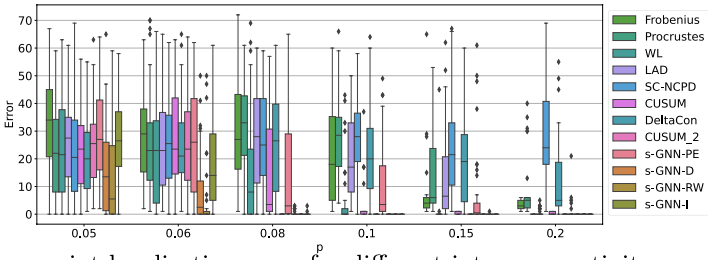
$$G_t \overset{i.i.d}{\sim} \mathcal{G}_1, \quad \text{if } 1 \leq t < \tau_i,$$
$$G_t \overset{i.i.d}{\sim} \mathcal{G}_2, \quad \text{if } \tau_i \leq t < \tau_{i+1},$$

for τ_i, $i \in \{1, 3, 5, 7\}$ and with $\tau_{K+1} = T$. In this case, $\mathcal{G}_1$ is an Erdös–Renyi model with edge probability $q = 0.02$, and $\mathcal{G}_2$ is the SBM of Scenario 2 from Sect. 4.3.1, with $p = 0.08$ and single cluster size $s \in [20, 80]$. We note that, similarly to the single change-point Scenario 2, the larger the subset size s is, the easier this multiple change-point problem becomes.

We divide each dynamic network sequence into training, validation, and test sub-sequences, with respectively 1000, 1000, and 400 snapshots. Besides, we sample $N = 9000$ pairs of snapshots for training and validation of our s-GNN. Note that the training and validation sub-sequences are also used to tune the detection threshold of the baselines, using the best Adjusted F1-score. For this experiment, we use the Identity encodings in our s-GNN and the detection rule (4) with $L = 6$. Moreover, since here, the size s tunes the

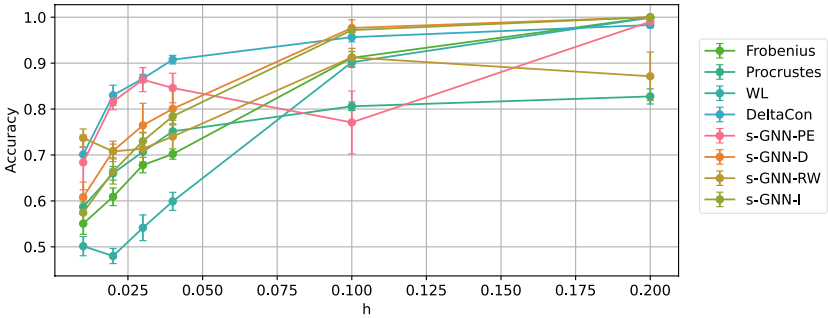


(a) Classification accuracy vs the intra-connectivity parameter p.

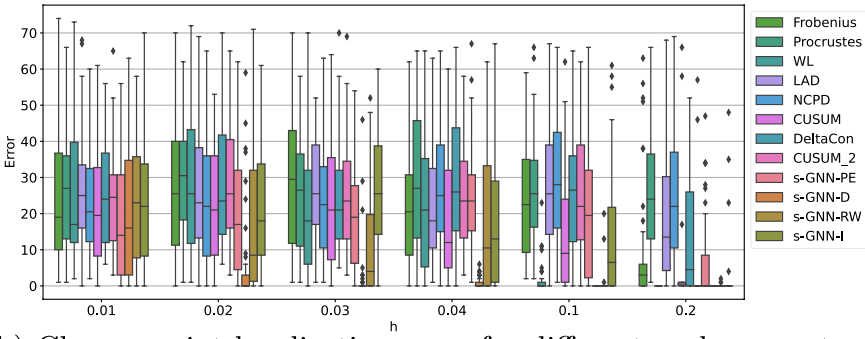


(b) Change point localisation error for different intra-connectivity parameters p.

Fig. 5 Performances on the classification (**a**) and detection (**b**) tasks in the "Birth 2" scenario



(a) Classification accuracy vs the exchange rate h.



(b) Change point localisation error for different exchange rates h.

Fig. 6 Performances on the classification (**a**) and detection (**b**) tasks in the "Swaps" scenario

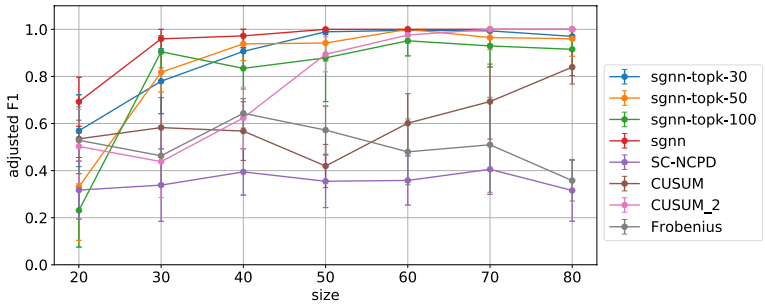


Fig. 7 Adjusted F1-score of our method (**sgnn**), its variants with fixed hyperparameter $k \in \{30, 50, 100\}$ (sgnn-topk-k), and the baselines versus the size of the denser subset s, in our synthetic experiment of dynamic networks with $n = 400$ nodes and multiple change-points of types "Birth" and "Death". Our results are averaged over 10 repetitions

spread of the change in the graph, we analyse the sensitivity of our method to the hyperparameter k, tuning the size of the Sort-k pooling layer in the s-GNN and thus the size of the displacement that can be detected by our method (see Sect. 3.2). In particular, we test variants of our method where this hyperparameter is fixed to a value, i.e., 30, 50 or 100 (denoted **sgnn-topk-30**, **sgnn-topk-50**, **sgnn-topk-100**). In each setting, we average our results over 10 repetitions.

In Fig. 7, we report the test performance, in terms of the Adjusted F1-score with tolerance level $t = 3$, of our method and the baselines, for different cluster sizes $s \in [20, 80]$. We note that our general method (**sgnn**) outperforms all non-trained baselines, across all difficulty levels, and is only matched by the CUSUM-2 baseline in the easiest settings, i.e., $s \geq 60$. Besides, the variant with $k = 30$ (**sgnn-topk-30**) appears to perform slightly better on average than the other two, namely **sgnn-topk-50** and **sgnn-topk-100**, in particular, in the settings with small subset size s. These results suggest that our method may perform better if k is chosen close to the size of the displacement; nonetheless, it is in general better to validate it.

Finally, we note that we also tested the ARI metric for this experiment, and obtained similar performance as with the Adjusted F1-score. Our numerical results can be found in Appendix A.4. We also note that we excluded from these results the Procrustes distance, the WL kernel, DeltaCon similarity, and the LAD algorithm which were performing poorly.

Remark 5 In comparison to the non-trained baseline methods, our method requires to train the s-GNN model before using it in an online setting. Therefore, it has in general a bigger computational cost. In fact, the complexity of each training epoch for the s-GNN is of order $O(NJ(Md + nd^2) + Nn \log n)$, where J and d are respectively the number of layers and hidden units of the GCN encoder, M and n are the average number of edges and the number of nodes in each snapshot, and N is the number of training samples (i.e., pairs of graphs sampled from the training sequence). Note that the $n \log n$ term comes from the Sort-k pooling layer. Once the s-GNN model is trained, the complexity of our change-point detection algorithm with a window size L on a test sequence with T snapshots is of order $O(TLJ(Md + nd^2) + TLn \log n)$, where in general $TL << N$. We also note that in our experiments, we construct our s-GNN model using the DGL Python library, which leverages the sparse graph and batch representations to accelerate matrix products and gradient

Table 1 Average computing times in seconds of our method (s-GNN) (training + testing times) and the baselines for one instance of the synthetic scenarios with multiple change-points ($T = 2400$ and $n = 400$)

Method	Average computing time (s)
s-GNN	5080.1 + 15.46
Frobenius distance	21.7
Procrustes distance	118.4
Weisfeiler–Lehman kernel	2372.3
LAD	119.6
SC-NCPD	496.7
CUSUM	127.3
CUSUM-2	843.4

computations. In comparison, the Cumulative Sums statistics and the Frobenius distance method only compute sums and norms of adjacency matrices, operations that are of order $O(TLM)$. Moreover, for sparse graphs, the baseline methods that solve an eigenvalue problem (LAD, SC-NCPD, and Procrustes distance) are of complexity $O(Tn^2)$. Finally, the WL kernel method is generally of order $O(TLnn_{iter})$, where n_{iter} is the number of iterations of the algorithm. In summary, our algorithm requires a bigger training cost that the baseline methods; nonetheless, at test time, it has a comparable complexity to the other methods. We note that the GCN encoders in our GNN architecture are quite shallow, since GNN models typically do not benefit from stacking many layers (Oono & Suzuki, 2019), due to oversmoothing phenomena. To illustrate this complexity analysis, we report in Table 1 the average runtime of our method and the baselines on one instance of the multiple change-point synthetic setting.

4.4 Real-world network data

4.4.1 Dynamic correlation network of stock returns

In this experiment, we analyse a dynamic correlation network, constructed from the time series of log-returns of stocks from the S &P 500 index, during a period of about 20 years (February 2000–December 2020). More precisely, each timestamp of this dynamic network is a month (between 02/2000 and 12/2020) and each snapshot is an unweighted graph obtained from the transformation of a correlation matrix between $n = 685$ stocks.

For a correlation matrix C, each entry C_{ij}, with $i, j = 1, \dots, n$, corresponds to the correlation coefficient between the time series of open-to-close (intra-day) and close-to-open (overnight) log returns of stocks i and j, over a sample of one month. Typically, there are 21 trading days in a calendar month, hence each stock has associated a time series of length 42, since each day of the month contributes with two returns. The correlation matrices define a fully-connected, weighted graph, with entries in $[-1, 1]$ and we first transform them into sparse matrices with entries in $\{0, 1\}$ using the following truncation rule. For each correlation matrix $C \in [-1, 1]^{n \times n}$, we define a binary adjacency matrix $A \in \{0, 1\}^{n \times n}$ as

$$A_{ij} = \mathbb{1}_{C_{ij} > q_{0.9}} + \mathbb{1}_{C_{ij} < q_{0.1}}, \quad \forall i, j = 1, \dots, n,$$

where $q_{0.1}, q_{0.9}$ are the 10% and 90% quantiles of all entries C_{ij} of all snapshots.

Table 2 Mean, median and standard deviation of network statistics, for the snapshots of the dynamic correlation network of S &P index stock log-returns

Financial network ($T = 244$, $n = 685$)			
	Mean	Median	Standard deviation
Number of edges per graph	46.3×10^3	40.3×10^3	23.6×10^3
Edge density	0.20	0.17	0.10
Average degree	135	96	111
Average shortest path length	1.8	1.8	0.1
Diameter	2.8	3.0	0.5

After this pre-processing step of the time series of log-returns, each snapshot of the dynamic network is a connected graph containing self-loops. We note that a similar procedure has been applied in Yu et al. (2021), while other previous work transforms the correlation matrices into complete weighted graphs, e.g., by squaring the correlation coefficients (Chakraborti et al., 2020) or computing the inverse of the ultra-metric distance (Samal et al., 2021). Here we adopt the sparsifying approach to avoid dealing with a large complete graph.

Moreover, in this experiment, we include the following additional information of the stocks' economic activity during each month as node attributes.

- volatilities (two features): the standard deviations of the above 42 open-to-close and close-to-open returns, based on which the correlation network was built;
- average daily volume, in shares, over the 21 days of the month;
- average shares outstanding, over the 21 days of the month.

We standardize the node attributes matrices using the mean and standard deviation of each attribute, across all nodes and snapshots of the dynamic network.

In summary, this dynamic network contains $T = 244$ unweighted, attributed snapshots $G_t = (\boldsymbol{A}_t, \boldsymbol{X}_t)$, $\boldsymbol{X}_t \in \mathbb{R}^{685\times 4}$, $t = 1, \ldots, T$, with an average edge density of 0.20 and average degree of 135 (see Table 2 for additional network properties). However, there is no ground-truth knowledge of change-points for this dynamic network. In fact, data sets of stock returns have previously been analysed in the NCPD context by Yu et al. (2021), Barnett and Onnela (2016), and Dubey et al. (2021). Closely related to our problem, Chakraborti et al. (2020) and Samal et al. (2021) cluster market behaviours in the USA S &P 500 and Japan Nikkei 225 stock networks and interpret changes in the behaviour following different economic or global events. These previous work therefore provides strong evidence that some major events, such as the ones listed in Table 3, have impacted the dynamics of stock returns and their correlation structure (Barnett & Onnela, 2016).

Nonetheless, without ground-truth change-points, our s-GNN cannot be trained in a supervised setting. Consequently, we design a self-supervised training procedure for this problem which consists in pre-estimating a set of change-points in a training and validation sub-sequences. We first divide the dynamic network into consecutive windows of 50%, 20% and 30% graph snapshots for respectively training, validation, and testing, then partition the timestamps in the training and validation sub-sequences using the following methodology. Since stocks are often clustered into market sectors (Chakraborti et al., 2020), we

Table 3 Dates of major financial crashes and bubbles in the USA market

Major crashes	Period date
9/11 Financial Crisis	11/09/2001
Stock Market Downturn Of 2002	09/10/2002
US Housing Bubble	2005–2007
Lehman Brothers Crash	16/09/2008
DJ Flash Crash	06/05/2010
Tsunami/Fukushima	11/03/2011
Black Monday/Stock Markets Fall	08/08/2011
Chinese Black Monday	24/08/2015
Dow Jones plunge	02/2018–03/2018
WHO public emergency state (COVID-19)	30/01/2020

conjecture that this cluster structure is reflected in the snapshots of our correlation network, and can be used as a proxy for the underlying state of the financial market at a given time. Therefore, we consider the following three-step procedure:

1. In each graph snapshot, we cluster the stocks into 13 groups, using a spectral clustering algorithm based on the normalised symmetric Laplacian (Gallier, 2016);
2. For each pair of graph snapshots, we measure the similarity between the partitions of stocks using the Adjusted Rand Index (ARI), and we use the latter as a graph similarity function. Then, we apply the spectral clustering algorithm but this time on the graph snapshots, using the symmetric normalised Laplacian of the similarity matrix obtained from the ARI scores (see the heatmaps of scores in Fig. 13 in Appendix A), and dividing the snapshots into 9 groups. We therefore obtain snapshot labels, interpreted as the state or behaviour of the stock market at each timestamp.
3. We pre-estimate change-points by "smoothing" the snapshots' labels: for each cluster of snapshots, we compute the "centroid" timestamp, and re-label each snapshot with the label of the closest centroid. These new labels now define a partition of the temporal window $[1, T]$ into consecutive intervals, and therefore, pre-estimated change-points in the training and validation sub-sequences.

We note that in the first and second steps, the number of cluster is chosen by evaluating the Silhouette index of the result clustering for different number of clusters $k \in \{2, \dots, 20\}$. The pre-estimated change-points that are obtained with the previous procedure are represented in Fig. 14a in Appendix B. We then sample $N = 3000$ pairs of graphs in the training sub-sequence using the Random scheme (see Sect. 3.4), and construct the validation set using the Windowed scheme with a window of size 12 on the validation sub-sequence. The set of hyperparameters of the s-GNN is chosen after a grid search on the validation set, using the Adjusted F1-score

Next, we detect change-points in the network sequence using our NCPD statistic $Z_t(s, L)$ (3) with a window size $L = 6$ (months) and our detection rule (4). In Fig. 8, we plot $1 - Z_t(s, L)$, over the whole sequence of the dynamic network, i.e., training, validation and test sub-sequences, and mark the detected change-points with red stars. In the top row, we represent a timeline of major market events that occurred during this period, i.e., major financial crashes and global events, also listed in Table 3. We note that these events could only consist of a subset of the ground-truth change-points for

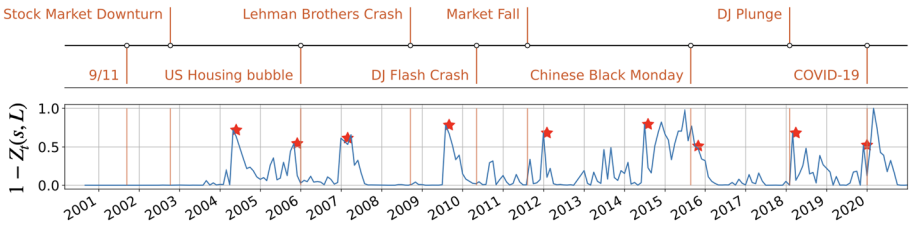


Fig. 8 Change-point statistic $1 - Z_t(s, L)$ and detected change-points (marked with red stars) obtained with our method on the dynamic correlation network of S &P 500 stock log-returns, with $n = 685$ nodes, observed from February 2000 to December 2020. This period covers a training period from February 2000 to August 2010, a validation period from September 2010 to October 2014 and a test period from November 2014 to December 2020. We note that the main financial events that occurred during this period, indicated with vertical red bars, are not ground-truth change-points for the network per-se. Nonetheless, the peaks of our change-point statistic in the test period (from November 2014 to December 2020) coincide with three market events, namely the Chinese Back Monday, the Dow Jones Plunge, and the COVID-19 pandemics

the correlation network of stocks. In particular, they potentially do not include changes in the relative importance of stocks, and other factors unreported in this analysis may also influence the structure of this dynamic network. Nonetheless, we can qualitatively interpret the detected change-point by our method and the peaks of our change-point statistics in light of this list of known events.

We note that some of the detected change-points coincide or happen soon after market events, in particular, in the test sub-sequence, from November 2014 to December 2020. During this period, a group of peaks with two change-points are observed around the Chinese Black Monday in August 2015, and two other change-points are detected in March 2018 and December 2019, which could be attributed respectively to the Dow Jones index's plunge in February–March 2018 and the emergence of COVID-19 in late 2019. Moreover, in the training and validation period (from February 2000 to October 2014), the change-point in November 2005 could be related to the US Housing Bubble, which spans a period between 2005 and 2007, while the change-point in February 2007 could also be linked to the latter event or to the premises of the financial crisis of 2007–2008. However, the change-points in August 2009 and January 2012 are difficult to relate to one of the listed events (possibly the Stock Market Fall for the latter one) and could potentially be false positives of our method. We note that these change-points are very close to some pre-estimated change-points (see Fig. 14 in Appendix B), therefore, could be due to false positives in our self-supervised learning procedure.

We then qualitatively compare our findings with the change-point detection statistics of the baseline methods applied to this data set, and the time series of VIX volatility indices during the period 2000–2020, in Fig. 9. We note that almost all statistics of the baselines display high values in the time period between 2010 and 2012, a period when several financial crashes happened, including the 2010 Dow Jone flash crash, the Fukushima nuclear incident in March 2011, and the Stock Markets Fall of August 2011. In contrast to our method, most baselines fail to indicate any potential change-point outside of this two-year period. One exception holds for the **CUSUM-2** method, which displays high values at roughly six periods. However, this method provides evidence of periods of disruptions for the network rather than clear change-points.

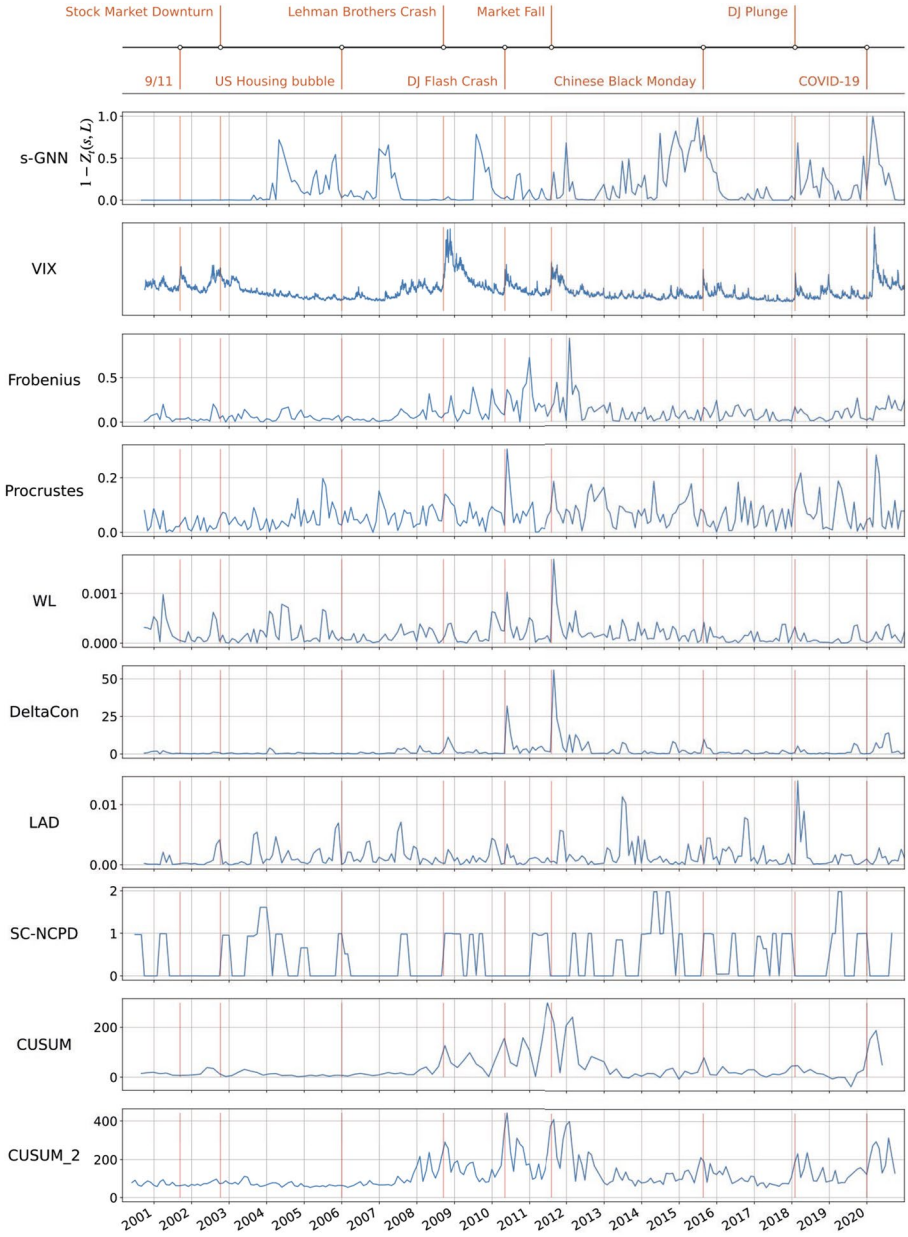


Fig. 9 VIX index and change-point detection statistics obtained with our method (s-GNN) and the baselines, on the dynamic correlation network of S &P 500 stock returns with $n = 685$ nodes, observed from February 2000 to December 2020. The top row is a timeline of main financial events that occurred during this period

Therefore, in comparison to the baseline NCPD methods, our algorithm seems to be able to detect more market events. One explanation could be that our method benefits from incorporating the stock attributes in the analysis, which are generally not taken into account

Table 4 Properties of the eight dynamic networks obtained from the physical activity monitoring data set

Subject	Activity performed						Number of change-points	Number of timestamps
	1–4	5	6	7	12, 13, 16, 17	24		
1	✓	✓	✓	✓	✓	✓	13	2490
2	✓	✓	✓	✓	✓	✓	13	2618
3	✓	–	–	–	✓	–	10	1732
4	✓	–	✓	✓	✓	–	11	2302
5	✓	✓	✓	✓	✓	✓	13	2709
6	✓	✓	✓	✓	✓	✓	13	2487
7	✓	✓	✓	✓	✓	–	12	2314
8	✓	✓	✓	✓	✓	✓	13	2606

We note that activities that are not listed have not been performed by any of the subjects in this experiment

by standard methods. To test our hypothesis, we also applied our method on this data set, replacing the node attributes by synthetic node encodings (see our results in Fig. 14c, d in Appendix B). We observed that without the node information, our method detects more change-points and is harder to interpret, thus providing evidence that the stock attributes are beneficial for our NCPD method.

4.4.2 Dynamic correlation network of physical activity monitoring sensors

In this experiment, we test our method on a dynamic correlation network constructed from a multiple sensors data set with changes of activity.

This public data set[3] was built for benchmarking time series classifiers on physical activity monitoring (Reiss & Stricker, 2012a, b), and contains multivariate time series recorded from eight subjects wearing 3D inertial measurement units (IMUs). These subjects have performed a protocol of 12 different physical activities such as sitting, walking, descending and ascending stairs, vacuum cleaning, etc. The eight time series, associated to each subject, have 27 dimensions, corresponding to measurements from 3 IMUs with 3-axis MEMS sensors and a sampling period of 0.01s, on 3 body parts. Moreover, each timestamp of this time series has a label corresponding to the performed activity. We note that, to our knowledge, this data has been analysed in the change-point detection task for time series, for instance by Zhang et al. (2020), but not in the context of NCPD yet. However, previous work notes that the correlations between pairs of axis are particularly useful for differentiating activities (Reiss & Stricker, 2012a).

Therefore, similarly to Sect. 4.4.1, for each subject, we build a dynamic correlation network using the 27-dimensional time series. In this network, a node is a dimension of the time series, and a timestamp corresponds to a window of 50 observations (i.e., a window of length 0.5 s). For each window, we compute the correlation matrix between the dimensions of the time series and transform it into a binary adjacency matrix by thresholding the absolute values of the correlation coefficients at 0.2. We then obtain eight unweighted, unattributed, dynamic networks with 27 nodes and around 2500 timestamps (see Table 4). Finally, each graph snapshot is labelled with the activity label of the time series window.

[3] http://www.pamap.org/demo.html.

We naturally consider that a change of activity between two consecutive snapshots corresponds to a change-point for the dynamic network.

We then design our NCPD evaluation set-up by defining the two following tasks.

- **Individual-level NCPD.** In this case, each of the eight dynamic networks (corresponding to one subject) is considered separately, and segmented into training, validation, and testing sub-sequences. Our method is then trained and tested on each network independently; in particular, the learnt graph similarity functions are subject-dependent. We note that, in this task, since activities are only performed once, the testing sub-sequence of the dynamic network contains snapshots with unseen activity labels and therefore, unseen types of change-points.
- **Cross-individual NCPD.** In this setting, the eight dynamic networks are mixed in the training, validation, and testing sub-sequences, we train and test a single s-GNN model for all subjects.

In the **Individual-level NCPD** task, we randomly split each dynamic network sequence into 70% training and validation, and 30% test, by isolating a test and a validation intervals. For the latter, we uniformly sample one change-point and select a window of size 60 s centered at this change-point. For the test interval, we sample uniformly the lower end, in the whole sequence. For this data, we use the **Random scheme** for both the training and validation sets of the s-GNN, and sample respectively 4000 and 1000 pairs. Furthermore, we sub-divide the **Cross-individual NCPD** task into two settings:

1. Random split: in this setting, we aggregate all the training, validation and test sub-sequences and sets obtained in the **Individual-level NCPD** task. Our method and the baselines are then tested on sub-sequences from every dynamic network.
2. Leave-one-subject-out (LOSO): in this setting, we keep the whole sequence of one subject for testing, and train and validate on the seven remaining ones. The training and validation sets are then obtained by aggregating the sets obtained in the **Individual-level NCPD** task.

In Appendix C.1, we report a preliminary analysis that we conducted to get an insight on the feasibility of the previously described NCPD tasks. In particular, we analyse (a) the similarity of adjacency matrices within each dynamic network, grouped by activity labels (Figs. 17 and 18); (b) the similarity of adjacency matrices within the same activity, grouped by network, i.e., subject (Figs. 19 and 20). We note that some activities have similar correlation matrices, such as activities $\{1, 2, 3\}$, that correspond to three static activities, i.e., sitting, lying, and standing. Moreover, for a subset of activities considered separately, the Frobenius distance between the correlation matrices of different subjects are bigger than the ones from the same subject. However, this difference is not always significant. Moreover, the average Frobenius distance between the graphs with the same label and from different subjects is smaller than the average distance between graphs with different labels, indicating that the dissimilarity between subjects is smaller than the dissimilarity between tasks.

Since in this data set, the networks are unattributed and the number of nodes is small, we use a s-GNN model with **Identity** node encoding in our method. Moreover, we use a window length of $L = 20$ timestamps in our change-point statistic (3). In Table 5, we report the performances of our method and baselines, measured in terms of the adjusted F1-score

Table 5 Adjusted F1-score of our method and baselines in the **Individual-level** and **Cross-individual** NCPD tasks on the physical activity monitoring data

Subject	s-GNN-I	Frobenius	SC-NCPD	CUSUM	CUSUM 2
Individual-level NCPD					
1	0.76 (0.20)	0.62 (0.31)	**0.82 (0.14)**	0.54 (0.30)	*0.81 (0.20)*
2	**0.91 (0.11)**	0.45 (0.22)	0.61 (0.08)	0.45 (0.12)	*0.84 (0.11)*
3	*0.60 (0.18)*	0.37 (0.14)	**0.67 (0.15)**	0.21 (0.27)	0.34 (0.26)
4	**0.73 (0.18)**	0.58 (0.26)	*0.70 (0.08)*	0.60 (0.07)	0.59 (0.22)
5	**0.85 (0.19)**	0.61 (0.22)	0.72 (0.16)	0.36 (0.24)	*0.72 (0.13)*
6	*0.74 (0.19)*	0.73 (0.22)	**0.75 (0.17)**	0.56 (0.30)	0.58 (0.16)
7	**0.90 (0.13)**	*0.79 (0.19)*	0.67 (0.35)	0.57 (0.23)	0.72 (0.37)
8	0.72 (0.24)	**0.88 (0.13)**	0.65 (0.14)	0.57 (0.28)	*0.82 (0.13)*
Cross-individual NCPD					
Random split	**0.81 (0.07)**	0.75 (0.03)	0.75 (0.03)	0.59 (0.12)	*0.80 (0.04)*
LOSO	**0.89 (0.02)**	0.70 (0.20)	*0.77 (0.06)*	0.62 (0.11)	0.75 (0.12)

The bold, respectively italics, values in each row denote the top best, respectively second best, performing methods. The values in the parentheses denote the standard deviation over 10 repetitions of the random splits train/validation/set, except for the Leave-one-subject-out (LOSO) setting for which mean and standard deviation are computed over the 8 folds

with a tolerance level of 5 timestamps. We note that our method has the best performance in most evaluation settings, in particular, it largely outperforms the baselines in the LOSO task. These results thus indicate that our s-GNN model is able to learn a graph similarity function that is generalisable to unseen subjects and activities.

5 Discussion and concluding remarks

In this work, we proposed a novel method for detecting change-points in dynamic networks using a data-driven graph similarity function, trained and validated on pairs of snapshots from the network sequence. We demonstrated on synthetic and real-world network data that our trained model is more accurate at distinguishing graphs with different generative distributions, and therefore, detecting change-points using a short history of data, compared to existing baselines.

Our method assumes that change-points can be detected by measuring the similarity between the current graph and its past snapshots, generated from a stationary reference distribution. However, this type of methodology is likely to under-perform when the stationarity assumption does not hold, or when the change of distribution is gradual and spans multiple timestamps. In particular, when there exists a transition regime between two consecutive stationary distributions, it is likely that the similarity scores $s(G_t, G_{t-1}), s(G_t, G_{t-2}), \ldots$ would stay quite high in the transition period and the change would not be detected by our algorithm, in particular when using a small window size L.

Moreover, our change-point statistic (3) does not explicitly take into account correlations between consecutive snapshots, although we aim at mitigating these correlations in our training set using the Random scheme (see Sect. 3.4). Even though the conditional independence assumption between snapshots is common in the dynamic network literature (see for instance Yu et al. (2021) which derive optimality results under this condition), this

may be a restrictive assumption in practice. In particular, the performance of our method may drop when the snapshots in a window of size L are highly correlated; nonetheless, one could potentially increase the robustness of our algorithm by sub-sampling the snapshots in $[t-L, t]$ in (3) (if the window size L can be chosen to a large value), or by using a forward window in combination with a two-sample test statistic (see Remark 2).

Besides, as previously noted, one main challenge posed by using a deep-learning based model for NCPD is the training and validation procedures, which necessitate either a dynamic network data set with change-point labels, or an adequate unsupervised or self-supervised learning procedure. Since the former is quite rare, a future direction for this work could be to develop the latter approach, for instance, using data augmentation strategies (Carmona et al., 2021) for introducing artificial change-points in the training set. Another possible extension would be to adapt our framework to more general types of dynamic networks, e.g., snapshots with varying node sets or with missing edges. In certain application domains, it may well be the case that change-points phenomena are localized only in certain parts of the network (as considered in some of our synthetic experiments), and are not affecting the global structure. To this end, yet another interesting addition to the current framework is to be able to pinpoint specifically which part of the network is mainly driving the change-point to enhance explainability, for instance using the nodes selected in the Sort-k pooling layer.

Finally, testing the methodology on different types of networks, such as directed networks, is an interesting direction to explore, especially in the context of recent work in the literature that encodes various measures of causality or *lead-lag* associations in multivariate time series as directed graphs (Bennett et al., 2022; Run, 2019). The structure of such weighted directed graphs may evolve over time, which motivates the need for change-point detection techniques, and in such setting, adapting traditional spectral methods for change-point detection would be challenging, due to the asymmetry of the adjacency matrix.

Appendix A: Additional material on the synthetic experiments

In this section, we provide additional details and results from our synthetic experiments in Sect. 4.3, in particular, the hyperparameter selection procedure, two sensitivity studies, on the window size parameter L and the choice of pooling layer of our s-GNN model, and the numerical performance in the multiple change-point settings.

A.1 Hyperparameter selection

In each scenario and difficulty level, we train our s-GNN over maximum 100 epochs using early stopping and the F1-score as our validation metric. Our hyperparameters are the number of hidden units, the size of the Sort-k layer, the dropout rate, the weight decay, and the learning rate of the ADAM optimiser. We select one set of hyperparameters per scenario by searching over a grid of values in one difficulty level, i.e., $p = 0.03$ in Scenario 1, $s = 60$ in Scenario 2, $p = 0.06$ in Scenario 3, and $h = 0.1$ in Scenario 4. The tested values are $\{0.001, 0.01\}$ for the learning rate, $\{0.01, 0.05, 0.1\}$ for the dropout rate, $\{20, 40, 100\}$ for the size of the Sort-k layer, and $\{16, 32, 64\}$ for the number of hidden units.

For each baseline based on a graph distance, kernel, or similarity function (i.e., the Frobenius and Procrustes distances, DeltaCon similarity, and the WL kernel), we use

the training and validation sets to tune the classification threshold θ, also using the best F1-score.

A.2 Sensitivity analysis with respect to the window size

In this experiment, we test the sensitivity of our NCPD method and the baselines to the window size parameter L, that sets the history of data used in (3), or in the baselines' change-point statistics, described in Sect. 4.2. We recall that this hyperparameter also sets the minimal distance between change-points that a method can detect.

For this analysis, we consider the **"Merge"** scenario from Sect. 4.3 and three difficulty levels $p = 0.3, 0.4, 0.5$, and evaluate performances using different window sizes $L \in \{6, 12, 24\}$. Our numerical results, reported in Fig. 10, show that our method is not very sensitive to the window size, in particular our best variant, **s-GNN-RW**, which uses the Random-Walk node encodings, outperforms the baselines for all values of L. We also note that the performance of methods based on CUSUM statistics, i.e., **CUSUM** and **CUSUM-2**, significantly increase for larger L. We can therefore from this experiment that the choice of L in our NCPD statistic (3) does not have a big impact on the performance, and therefore does not require to be finely tuned.

A.3 Sensitivity analysis with respect to the pooling layer

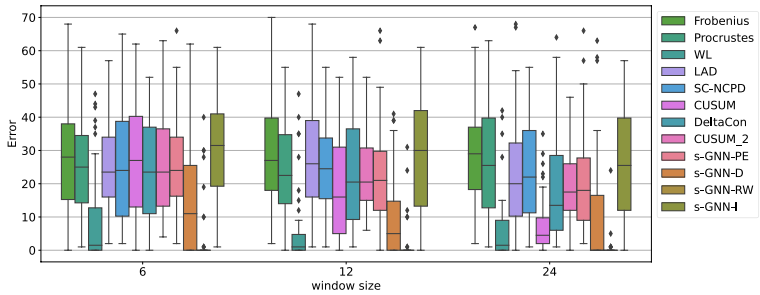
In this section, we conduct an ablation experiment to test the importance of the Sort-k pooling layer in our similarity module (see Sect. 3.2. For this analysis, we consider the Scenario 2 from Sect. 4.3 with different difficulty levels $s \in [40, 100]$ and evaluate the performance of our method when using an s-GNN with a Sort-k, Max, or Average pooling layer. Our findings are reported in Fig. 11.

We observe that in this experiment, using Max pooling significantly increases the localisation error of our method, while Average pooling has a higher variance and mean error than Sort-k pooling in most settings. We conjecture that Max pooling is poorly performing in this context because it is less robust to the sparsity of the network than Sort-k and Average pooling. Moreover, since Average pooling sums the displacements, here measured in terms of the Euclidean between the node embeddings in the two graphs, over the whole set of nodes, it may not be able to detect local changes. Therefore, we can conclude that using a Sort-k pooling layer in our s-GNN model is beneficial for our NCPD method, and is probably more adapted to detect small distribution changes, while being robust to the sparsity of edges.

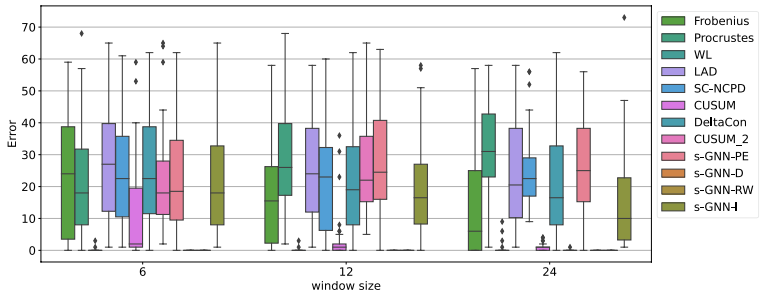
A.4 Adjusted Rand Index performance

In this section, we report the numerical results of our synthetic experiments with multiple change-points from Sect. 4.3.2, evaluated in terms of the Adjusted Rand Index (ARI) and compare to the performance as measured by the Adjusted F1-score.

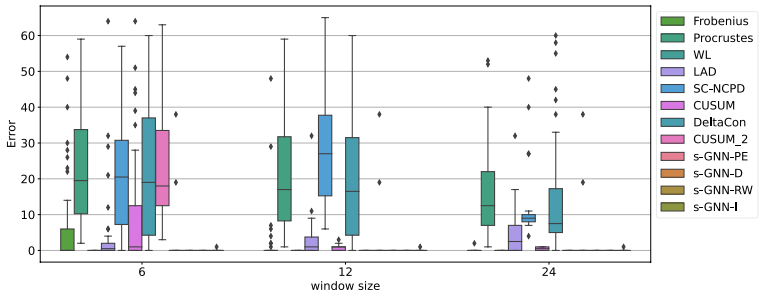
From Fig. 12, which is the analog of Fig. 7 with the ARI as the performance metric, we note that we obtain similar performance as with the Adjusted F1-score, an intuition that is confirmed by the numerical results in Table 6.



(a) Difficult level



(b) Moderate level



(c) Easy level

Fig. 10 Localisation error for a single change-point of our method and the baselines, when using different window sizes $L \in \{6, 12, 24\}$, in the **"Merge"** scenario from Sect. 4.3.1. The three panels correspond to three difficulty levels of the detection task: difficult ($p = 0.3$) (**a**), moderate ($p = 0.4$) (**b**), and easy ($p = 0.5$) (**c**)

Appendix B: Additional material on the application to S &P 500 stock returns

In this section, we first provide additional illustrations related to our self-supervised learning procedure, then, report a supplementary analysis of this financial network data set using the methodology of Chakraborti et al. (2020), that we relate to our findings from Sect. 4.4.1.

We recall that, for this data set, the lack of ground-truth change-points leads us to design a self-supervised method which consists in pre-estimating change-points in the training and

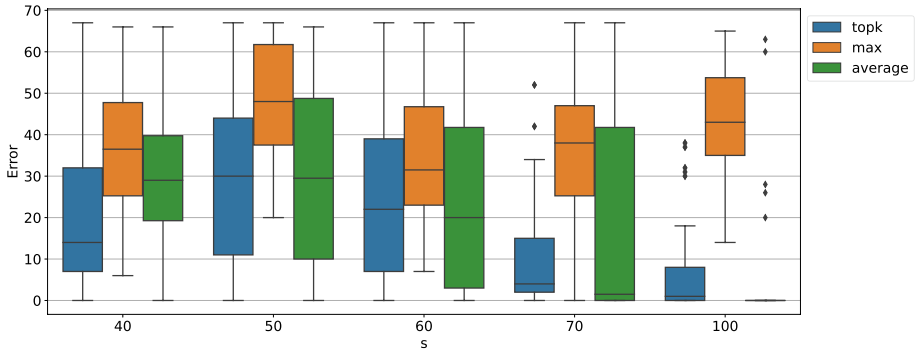


Fig. 11 Localisation error of our method when using a s-GNN model with Max, Average, or Sort-k pooling layer, in the Scenario 2 from Sect. 4.3.1 with different difficulty levels $s \in \{40, 50, 60, 70, 100\}$. We note that the Sort-$k$ pooling method has the best performance in this experiment

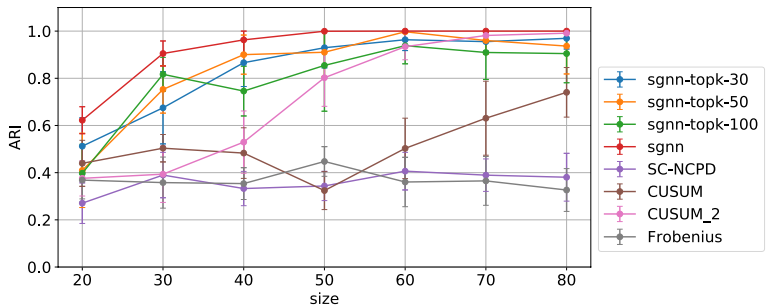


Fig. 12 Adjusted Rand Idex of our method (sgnn), its variants with fixed hyperparameter $k \in \{30, 50, 100\}$ (sgnn-topk-k), and the baselines versus the size of the denser subset, in our synthetic experiment with multiple change-points of types "Birth" and "Death", described in Sect. 4.3.2. The results are averaged over 10 repetitions. We note that these results are similar to the ones in Fig. 7, where the performances are measured with the Adjusted F1-score

validation sub-sequences. In Fig. 13, we plot the heatmap of the similarity matrix between the graph snapshots, as measured by the Adjusted Rand Index, between the stocks partitions obtained by the spectral clustering algorithm applied to each snapshot. We note that this similarity matrix seems to have a cluster structure; in particular, high similarity scores can be found during the period of the financial crisis from 2007 to 2011 and in 2001–2002.

The pre-estimated change-points after clustering the previous similarity matrix and smoothing the labels are represented in Fig. 14a. We recall that these change-points allow to sample labelled training and validation sets, using the procedure described in Sect. 3.4, and to train our s-GNN model. Then, we can compare the change-points estimated by our method applied to the whole network sequence, i.e., including training, validation, and test sub-sequences, represented in Fig. 14b, to the pre-estimated ones. We note that our method detects fewer change points than our pre-estimation procedure. However, most detected change-points are less than four months away from a pre-estimated one.

We also report in Fig. 14c, d our results when we ignore the node attributes of the network. In this ablation study, we use our s-GNN model with Identity node encodings,

Table 6 Adjusted F1-score and Adjusted Rand Index in the multiple change-point detection settings of Sect. 4.3.2, with size $s \in [20, 80]$

Size	s-GNN	CUSUM 2	CUSUM	SC-NCPD	Frobenius
Adjusted F1-score					
20	**0.69**	0.50	0.53	0.32	0.53
30	**0.96**	0.44	0.58	0.34	0.46
40	**0.97**	0.62	0.57	0.39	0.64
50	**1.00**	0.89	0.42	0.35	0.57
60	**1.00**	0.98	0.60	0.36	0.48
70	**1.00**	**1.00**	0.69	0.41	0.51
80	**1.00**	**1.00**	0.84	0.32	0.36
Adjusted Rand Index					
20	**0.62**	0.38	0.44	0.27	0.37
30	**0.91**	0.39	0.50	0.39	0.36
40	**0.96**	0.53	0.48	0.33	0.35
50	**1.00**	0.80	0.32	0.34	0.45
60	**1.00**	0.93	0.50	0.41	0.36
70	**1.00**	0.98	0.63	0.39	0.37
80	**1.00**	0.99	0.74	0.38	0.33

These performances are averaged over 10 repetitions and the best ones are highlighted in bold

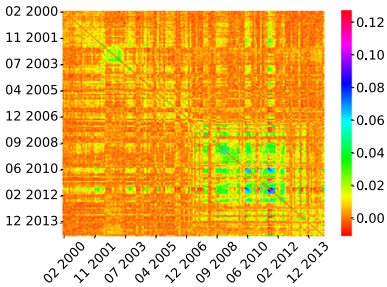
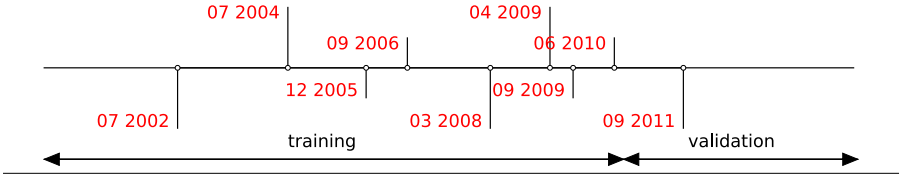


Fig. 13 Heatmap of the pairwise similarity matrix between the graph snapshots of the training and validation sub-sequences from financial correlation network. We recall that the similarity is measured in terms of the Adjusted Rand Index values between the partitions obtained for each pair of graph snapshots. The first two digits denote the month, followed by the year, of the timestamp of each snapshot

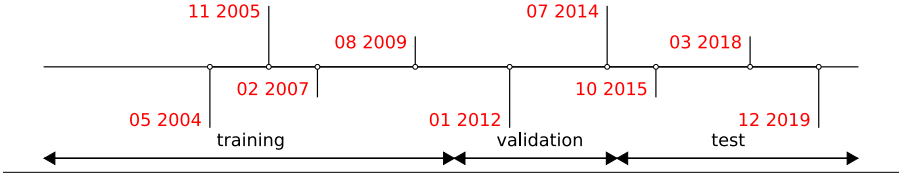
and observe that our method then detects additional change-points in the sequence. Although we cannot draw any definite conclusion from this analysis, we conjecture that some of the latter change-points are False Positives, and that using the node attributes data decreases the False Discovery Rate.

Moreover, we compare our findings with existing methodology applied to similar data and we analyse this dynamic correlation networks using the eigen-entropy, employed by Chakraborti et al. (2020) to discover market behaviours—an analysis distinct though related to change-point detection.

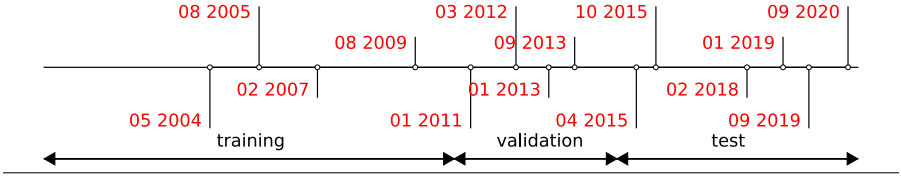
We recall from their method that the eigen-entropy of a graph is the entropy of the eigen-centrality vector, defined as a L_1-normalised version of the principal eigenvector of the graph adjacency matrix. Note that the principal eigenvector is related to the



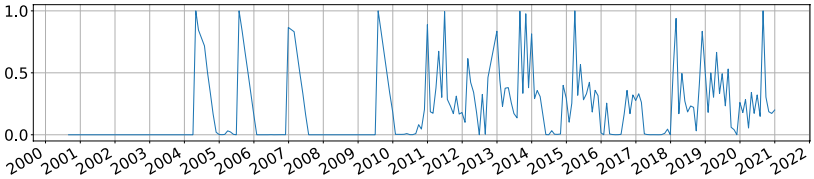
(a) Pre-estimated change-points on the training and validation sequences.



(b) Change-points estimated by our method on the training, validation and test sequences.



(c) Change-points estimated by our method with the **Identity encoding**



(d) Network change-point statistic obtained with our method with the **Identity encoding**.

Fig. 14 Change-points estimated on the S &P 500 stock returns correlation network by the pre-estimation procedure described in Sect. 4.4.1 (**a**) and by our method on the attributed data (**b**) and on the non-attributed data (**c**). In the latter case, we have used the **Identity node encodings** as synthetic attributes and the obtained network change-point statistic is reported in the last panel (**d**)

relative ranks of the different stocks in the market, and its entropy can be interpreted as a measure of the market "disorder". For each timestamp t, Chakraborti et al. (2020) compute the eigen-entropy of the correlation matrix, denoted $H(t)$, and of its two subparts, the market mode, given by the principal eigen matrix, and the composite group plus random mode, respectively $H_M(t)$ and $H_{GR}(t)$. In Fig. 15, we represent H, H_M, H_{GR} over time, however, this time series is not directly interpretable. As in Chakraborti et al. (2020), we represent these eigen-entropy values in 2D-phase spaces, together with the snapshot labelling obtained with our self-supervised procedure, in Fig. 16a.

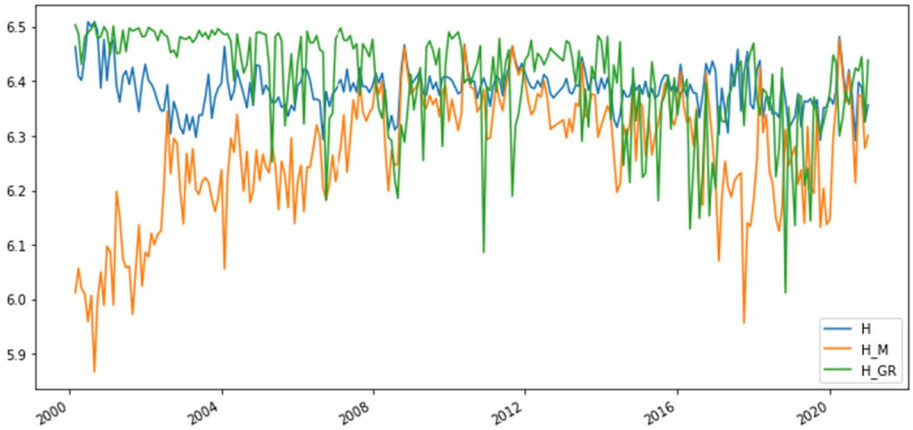


Fig. 15 Eigen-entropy of the correlation matrices $(H(t))_t$, the market mode $(H_M(t))_t$ and the group plus random mode $(H_{GR}(t))_t$ over time, in our financial data set of S &P stock returns

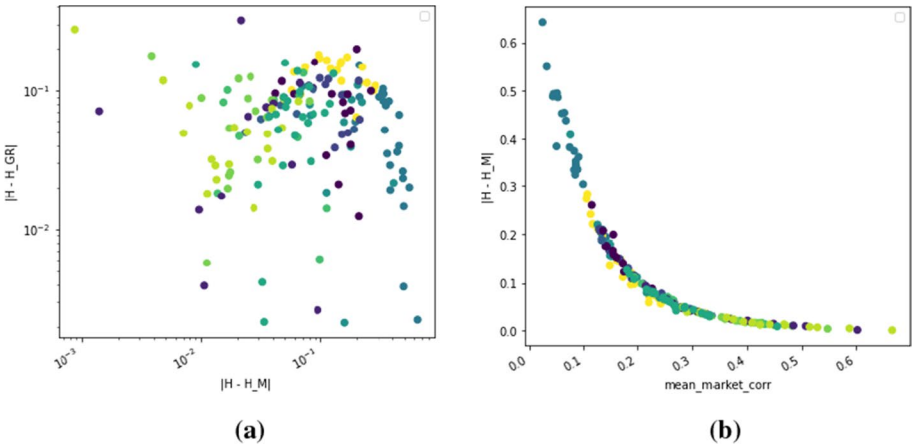
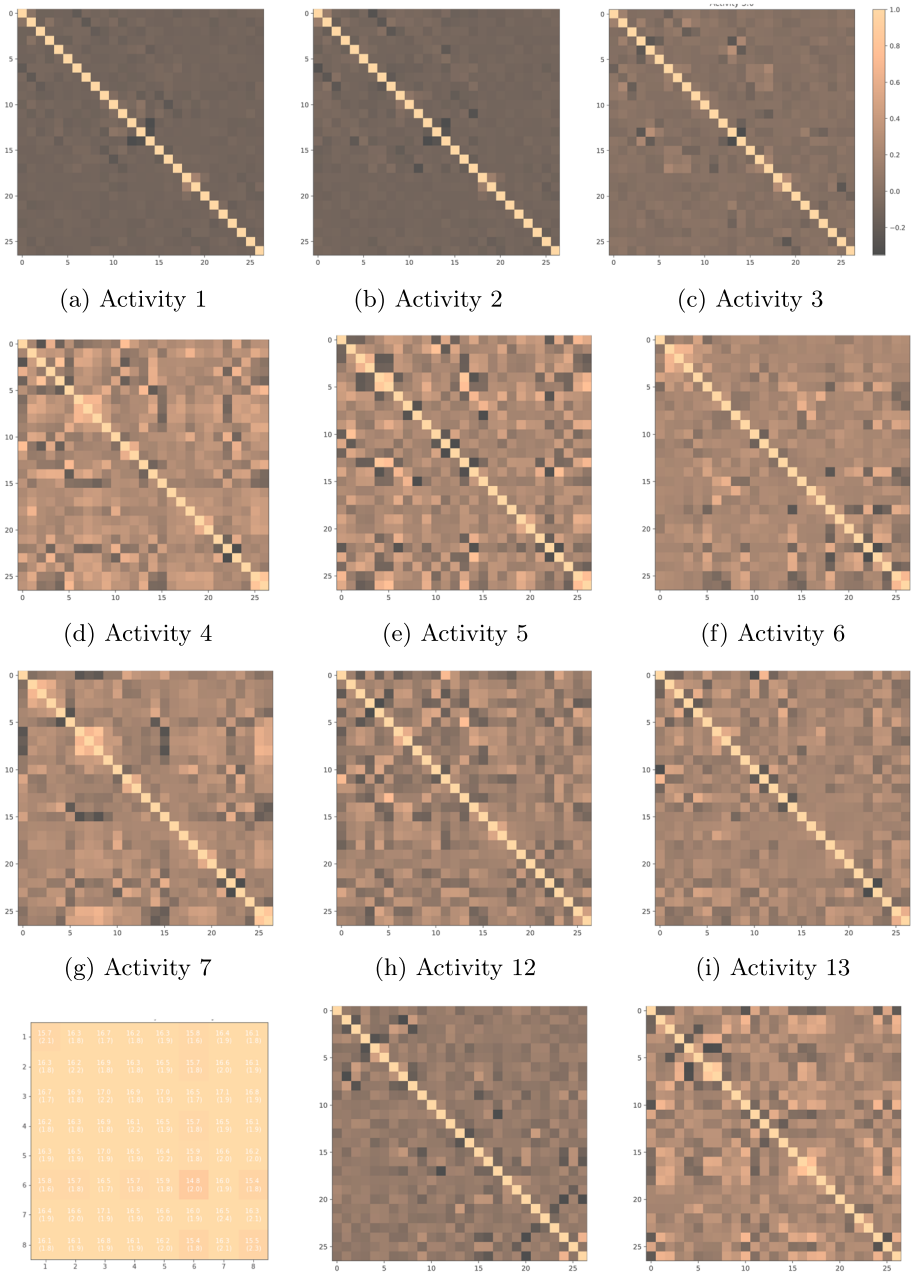


Fig. 16 Entropy differences $|H(t) - H_{GR}(t)|$ versus $|H(t) - H_M(t)|$ (in log scale) (**a**) and $|H(t) - H_M(t)|$ versus mean market correlation (**b**) of the graph snapshots in the financial correlation dynamic network. Each dot in the phase spaces corresponds to a snapshot, and its colors indicate its label, as pre-estimated in our self-supervised procedure

We note that, although we do not observe a clear clustering of the snapshots in this 2D representation, graphs with the same labels seems to lie closer to each other, which supports our intuition that the cluster structure of the snaphots reflects the state of the market at a certain time stamp.



(a) Activity 1 (b) Activity 2 (c) Activity 3

(d) Activity 4 (e) Activity 5 (f) Activity 6

(g) Activity 7 (h) Activity 12 (i) Activity 13

(j) Activity 16 (k) Activity 17 (l) Activity 24

Fig. 17 Heatmaps of the average adjacency matrices of the snapshots from the first dynamic correlation network (Subject 1) of the physical sensor data, grouped by their activity label

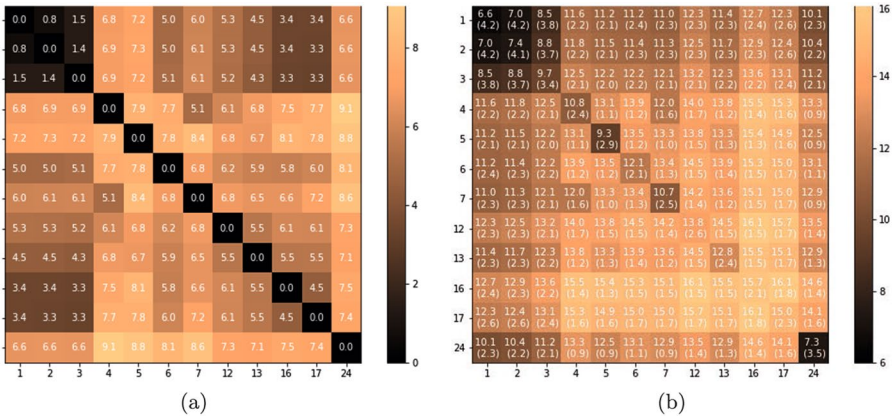


Fig. 18 Heatmaps of Frobenius distances between the snapshots' adjacency matrices from the first dynamic network (Subject 1): between the average adjacency matrices per activity (**a**), and average (and standard deviation) distances between adjacency matrices grouped by activities (**b**)

Appendix C: Additional material on the physical activity monitoring data set

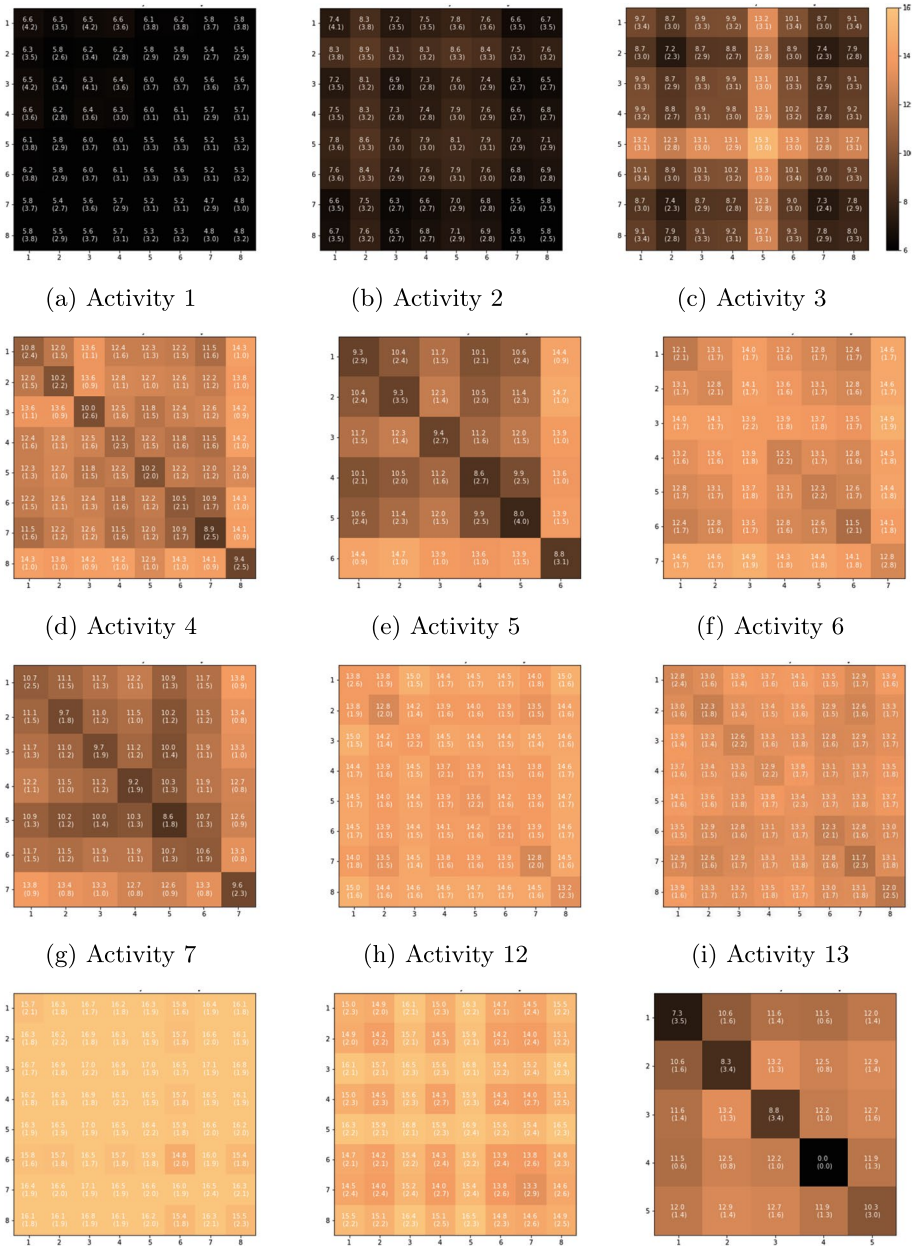
In this section, we report two additional analysis of the real-world data set of physical sensors measurements.

C.1 Preliminary analysis of the dynamic networks

In this section, we evaluate the feasibility of the NCPD tasks that we designed for the dynamic correlation networks of physical sensors (see Sect. 4.4.2). In particular, we are interested in the dissimilarity between the snapshots' adjacency matrices corresponding to each activity and subject.

In Fig. 17, we plot the average adjacency matrices, for each activity, from the first subject, and in Fig. 18, the pairwise Frobenius distance between these matrices. We note that the smallest distances are mostly diagonal entries, indicating that the matrices with the same activity label are more similar to each other than ones with different labels.

In Fig. 19, we plot the Frobenius distances between the graphs of different subjects, with the same activity label, for four activities. We observe that for activities 5 and 7, the distances between matrices of different subjects are bigger than the ones from the same subject, however this difference is not always significant and does not seem to appear for activities 1 and 17. Figure 20 confirms this observation: we note that the average Frobenius distance between the graphs with the same label and from different subjects is smaller than the average distance between graphs with different labels.



(a) Activity 1 (b) Activity 2 (c) Activity 3

(d) Activity 4 (e) Activity 5 (f) Activity 6

(g) Activity 7 (h) Activity 12 (i) Activity 13

(j) Activity 16 (k) Activity 17 (l) Activity 24

Fig. 19 Heatmaps of average (and standard deviation) Frobenius distances between the snapshots' adjacency matrices, grouped by subjects, for each activity label

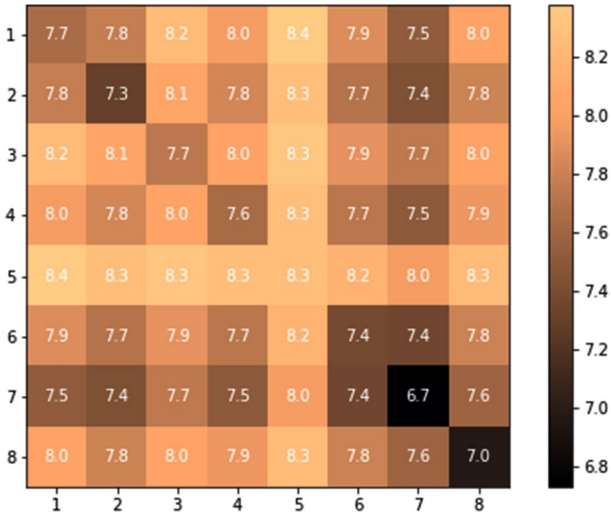


Fig. 20 Heatmaps of the average Frobenius distances between the snapshots' adjacency matrices with the same activity label, grouped by subjects

Table 7 Performance of our method (s-GNN-I) and baselines, as measured by the adjusted F1-score with different tolerance levels, in the **Cross-individual NCPD** task and random split setting of the physical activity monitoring data

Cross-individual NCPD					
Tolerance level	s-GNN-I	Frobenius	SC-NCPD	CUSUM	CUSUM 2
1	**0.60 (0.30)**	0.53 (0.22)	0.43 (0.32)	0.33 (0.24)	0.42 (0.30)
3	**0.87 (0.25)**	0.68 (0.20)	0.70 (0.31)	0.53 (0.24)	0.76 (0.29)
5	**0.61 (0.27)**	0.53 (0.20)	0.41 (0.32)	0.27 (0.22)	0.44 (0.31)
7	**0.85 (0.28)**	0.71 (0.21)	0.71 (0.30)	0.56 (0.25)	0.75 (0.29)

The bold values in each row denote the top performing method, and values in parentheses correspond to the standard deviations over 10 repetitions of the sampling scheme for the training/validation/testing sets (see Sect. 4.4.2). We note for each tolerance level, our method attains superior performance when compared to other baselines. We remark that choosing different tolerance levels essentially amounts to defining different sets of ground truth change-points, hence, we should not necessarily expect a monotonic relationship between tolerance versus the recovery accuracy of all methods

C.2 Sensitivity analysis to the tolerance level of the adjusted F1-score

In this section, we investigate the sensitivity of our results reported in Table 5 to the tolerance level chosen to compute the adjusted F1-score. For this analysis, we consider the **Random split** setting of the **Cross-Individual** task (see Sect. 4.4.2), and reproduce this experiment for different tolerance level $t \in \{1, 3, 5, 7\}$. From our numerical results in Table 7, we note that our method has the best performance for all considered levels.

Author contributions All authors contributed to the writing of the paper. DS performed numerical experiments.

Funding MC acknowledges support from the EPSRC Grant EP/N510129/1 at The Alan Turing Institute. XD acknowledges support from the Oxford-Man Institute of Quantitative Finance and the EPSRC (EP/T023333/1). DS is supported by the EPSRC and MRC Centre for Doctoral Training in Statistical Science, University of Oxford (Grant EP/L016710/1). HK is supported by the EPSRC Centre for Doctoral Training in Autonomous Intelligent Machines and Systems, University of Oxford (EP/L015897/1).

Availability of data, material and code The code of the NCPD method is available on https://github.com/dsulem/DyNNet.

Declarations

Conflict of interest The authors have no conflicts of interest to declare.

Ethics approval The authors did not need ethics approval for this work.

Consent to participate Not applicable.

Consent for publication Not applicable.

References

Runge, K., et al. (2019). Detecting causal associations in large nonlinear time series datasets. *Science Advances*. https://doi.org/10.1126/sciadv.aau4996

Bao, D., You, K., & Lin, L. (2018). Network distance based on Laplacian flows on graphs. arXiv:1810.02906.

Barnett, I., & Onnela, J. P. (2016). Change point detection in correlation networks. *Scientific Reports, 6*(1), 18893. https://doi.org/10.1038/srep18893

Bennett, S., Cucuringu, M., & Reinert, G. (2022). Lead-lag detection and network clustering for multivariate time series with an application to the us equity market. *KDD 2021 MileTS (preliminary workshop version)*. https://doi.org/10.48550/ARXIV.2201.08283.

Bhamidi, S., Jin, J., & Nobel, A. (2018). Change point detection in network models: Preferential attachment and long range dependence. *The Annals of Applied Probability, 28*(1), 35–78.

Bhattacharjee, M., Banerjee, M., & Michailidis, G. (2020). Change point estimation in a dynamic stochastic block model. *Journal of Machine Learning Research, 21*, 1–59.

Bhattacharjee, M., Banerjee, M., & Michailidis, G. (2020b). Change point estimation in a dynamic stochastic block model. 1812.03090.

Bourqui, R., Gilbert, F., & Simonetto, P., et al. (2009). Detecting structural changes and command hierarchies in dynamic social networks. In *2009 International conference on advances in social network analysis and mining, IEEE* (pp. 83–88).

Bruna, J., & Li, X. (2017). Community detection with graph neural networks. *stat, 1050*, 27.

Cai, L., Chen, Z., & Luo, C., et al. (2021). *Structural temporal graph neural networks for anomaly detection in dynamic graphs*. Association for Computing Machinery, New York, NY, USA (pp. 3747–3756). https://doi.org/10.1145/3459637.3481955.

Carmona, C.U., Aubet, F.X., & Flunkert, V., et al. (2021). *Neural contextual anomaly detection for time series*. arXiv:2107.07702.

Chakraborti, A., Sharma, K., Pharasi, H. K., et al. (2020). Phase separation and scaling in correlation structures of financial markets. *Journal of Physics: Complexity, 2*(1), 015,002.

Chu, L., & Chen, H. (2018). Sequential change-point detection for high-dimensional and non-euclidean data. 1810.05973.

Corneli, M., Latouche, P., & Rossi, F. (2018). Multiple change points detection and clustering in dynamic networks. *Statistics and Computing, 28*, 989–1007.

Cribben, I., & Yu, Y. (2017). Estimating whole-brain dynamics by using spectral clustering. *Journal of the Royal Statistical Society Series C: Applied Statistics, 66*(3), 607–627. https://doi.org/10.1111/rssc.12169. arXiv:1509.03730.

De Ridder, S., Vandermarliere, B., & Ryckebusch, J. (2016). Detection and localization of change points in temporal networks with the aid of stochastic block models. *Journal of Statistical Mechanics: Theory and Experiment, 2016*(11), 113,302. https://doi.org/10.1088/1742-5468/2016/11/113302

Delvenne, J. C., Yaliraki, S. N., & Barahona, M. (2010). Stability of graph communities across time scales. *Proceedings of the National Academy of Sciences, 107*(29), 12,755-12,760. https://doi.org/10.1073/pnas.0903215107

Desobry, F., Davy, M., & Doncarli, C. (2005). An online kernel change detection algorithm. *IEEE Transactions on Signal Processing, 53*(8), 2961–2974.

Donnat, C., & Holmes, S. (2018). Tracking network dynamics: A survey of distances and similarity metrics. 1801.07351.

Dubey, P., Xu, H., & Yu, Y. (2021). Online network change point detection with missing values. 2110.06450.

Dwivedi, V.P., & Bresson, X. (2021). A generalization of transformer networks to graphs. 2012.09699.

Dwivedi, V.P., Luu, A.T., & Laurent, T., et al. (2022). Graph neural networks with learnable structural and positional representations. 2110.07875.

Enikeeva, F., & Klopp, O. (2021). Change-point detection in dynamic networks with missing links. 2106.14470.

Gallier, J. (2016). Spectral theory of unsigned and signed graphs. Applications to graph clustering: A survey. 1601.04692.

Ghoshdastidar, D., Gutzeit, M., Carpentier, A., et al. (2020). Two-sample hypothesis testing for inhomogeneous random graphs. *The Annals of Statistics*. https://doi.org/10.1214/19-aos1884

Gretton, A., Borgwardt, K., & Rasch, M. J., et al. (2008). A kernel method for the two-sample problem. 0805.2368.

Hamilton, W., Ying, Z., & Leskovec, J. (2017). Inductive representation learning on large graphs. *Advances in neural information processing systems 30*.

Han, Y., Huang, G., Song, S., et al. (2021). Dynamic neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 44*(11), 7436–7456.

Harchaoui, Z., Moulines, E., & Bach, F. R. (2009). Kernel change-point analysis. In *Advances in neural information processing systems* (pp. 609–616).

Hewapathirana, I. U., Lee, D., Moltchanova, E., et al. (2020). Change detection in noisy dynamic networks: A spectral embedding approach. *Social Network Analysis and Mining, 10*(1), 1–22.

Horváth, A. (2020). Sorted pooling in convolutional networks for one-shot learning. arXiv preprint arXiv:2007.10495.

Huang, S., Hitti, Y., & Rabusseau, G., et al. (2020). Laplacian change point detection for dynamic graphs https://doi.org/10.1145/3394486.3403077. arXiv:2007.01229.

Kipf, T. N., Welling, M. (2016). Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907.

Koutra, D., Shah, N., Vogelstein, J. T., et al. (2016). Deltacon: Principled massive-graph similarity function with attribution. *ACM Trans Knowl Discov Data, 10*, 28:1-28:43.

Kriege, N. M., Johansson, F. D., & Morris, C. (2020). A survey on graph kernels. *Applied Network Science*. https://doi.org/10.1007/s41109-019-0195-3

Ktena, S. I., Parisot, S., & Ferrante, E., et al. (2017). Distance metric learning using graph convolutional networks: Application to functional brain networks. In *International conference on medical image computing and computer-assisted intervention* (pp. 469–477). Springer.

Kumar, S., Zhang, X., & Leskovec, J. (2019). Predicting dynamic embedding trajectory in temporal interaction networks. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. https://doi.org/10.1145/3292500.3330895.

Lee, J., Lee, I., & Kang, J. (2019). Self-attention graph pooling. In *36th international conference on machine learning, ICML 2019. International Machine Learning Society (IMLS)*, pp 6661–6670, funding Information: This work was supported by the National Research Foundation of Korea (NRF-2017R1A2A1A17069645, NRF-2016M3A9A7916996, NRF-2017M3C4A7065887) Publisher Copyright: © 36th International Conference on Machine Learning, ICML 2019. All rights reserved.; 36th International Conference on Machine Learning, ICML 2019; Conference date: 09-06-2019 Through 15-06-2019.

Li, A., Cornelius, S. P., Liu, Y. Y., et al. (2017). The fundamental advantages of temporal networks. *Science, 358*(6366), 1042–1046. https://doi.org/10.1126/science.aai7488

Li, P., Wang, Y., & Wang, H., et al. (2020). Distance encoding: Design provably more powerful neural networks for graph representation learning. 2009.00142.

Li, Y., Gu, C., & Dullien, T., et al. (2019). Graph matching networks for learning the similarity of graph structured objects. In *ICML*.

Ling, X., Wu, L., Wang, S., et al. (2021). Multilevel graph matching networks for deep graph similarity learning. *IEEE Transactions on Neural Networks and Learning Systems*. https://doi.org/10.1109/tnnls.2021.3102234

Liu, J., Ma, G., & Jiang, F., et al. (2019). Community-preserving graph convolutions for structural and functional joint embedding of brain networks. In: *2019 IEEE international conference on Big Data (Big Data), IEEE* (pp. 1163–1168).

Liu, Y., Pan, S., & Jin, M., et al. (2021). Graph self-supervised learning: A survey. arXiv preprint arXiv:2103.00111.

Ma, G., Ahmed, N. K., & Willke, T., et al. (2019). Similarity learning with higher-order graph convolutions for brain network analysis. 1811.02662.

Ma, G., Ahmed, N. K., Willke, T. L., et al. (2021). Deep graph similarity learning: A survey. *Data Mining and Knowledge Discovery, 35*(3), 688–725.

Majhi, S., Perc, M., & Ghosh, D. (2022). Dynamics on higher-order networks: A review. *Journal of the Royal Society Interface, 19*(188), 20220,043.

Manessi, F., Rozza, A., & Manzo, M. (2020). Dynamic graph convolutional networks. *Pattern Recognition, 97*, 107,000. https://doi.org/10.1016/j.patcog.2019.107000

Miller, H., & Mokryn, O. (2020). Size agnostic change point detection framework for evolving networks. *PLoS ONE, 15*(4), e0231,035.

Nie, L., & Nicolae., D. L. (2021). Weighted-graph-based change point detection.

Ofori-Boateng, D., Gel, Y. R., & Cribben, I. (2019). Nonparametric anomaly detection on time series of graphs. bioRxiv.

Ondrus, M., Olds, E., & Cribben, I. (2021). Factorized binary search: change point detection in the network structure of multivariate high-dimensional time series. https://doi.org/10.48550/ARXIV.2103.06347, arXiv:2103.06347.

Oono, K., & Suzuki, T. (2019). Graph neural networks exponentially lose expressive power for node classification. arXiv preprint arXiv:1905.10947

Padilla, O. H. M., Yu, Y., & Priebe, C. E. (2019). Change point localization in dependent dynamic nonparametric random dot product graphs arXiv:1911.07494.

Peel, L., & Clauset, A. (2015). Detecting change points in the large-scale structure of evolving networks. In *Twenty-ninth AAAI conference on artificial intelligence*.

Ranshous, S., Shen, S., Koutra, D., et al. (2015). Anomaly detection in dynamic networks: A survey. *Wiley Interdisciplinary Reviews: Computational Statistics, 7*(3), 223–247.

Reiss, A., & Stricker, D. (2012a). Creating and benchmarking a new dataset for physical activity monitoring. In *Proceedings of the 5th international conference on pervasive technologies related to assistive environments*. Association for Computing Machinery, New York, NY, USA, PETRA '12, https://doi.org/10.1145/2413097.2413148.

Reiss, A., & Stricker, D. (2012b). Introducing a new benchmarked dataset for activity monitoring. In *2012 16th international symposium on wearable computers* (pp. 108–109). IEEE.

Rossetti, G., & Cazabet, R. (2018). Community discovery in dynamic networks. *ACM Computing Surveys, 51*(2), 1–37. https://doi.org/10.1145/3172867

Rossi, E., Chamberlain, B., & Frasca, F., et al. (2020). Temporal graph networks for deep learning on dynamic graphs. 2006.10637.

Samal, A., Pharasi, H. K., & Ramaia, S. J., et al. (2021). Network geometry and market instability. 2009.12335.

Sankar, A., Wu, Y., & Gou, L., et al. (2020). DySAT: Deep neural representation learning on dynamic graphs via self-attention networks, Association for Computing Machinery, New York, NY, USA (pp. 519–527). https://doi.org/10.1145/3336191.3371845.

Seo, Y., Defferrard, M., Vandergheynst, P., et al. (2018). Structured sequence modeling with graph convolutional recurrent networks. In L. Cheng, A. C. S. Leung, & S. Ozawa (Eds.), *Neural Information Processing* (pp. 362–373). Cham: Springer.

Shervashidze, N., Schweitzer, P., van Leeuwen, E. J., et al. (2011). Weisfeiler–Lehman graph kernels. *Journal of Machine Learning Research, 12*, 2539–2561.

Siglidis, G., Nikolentzos, G., Limnios, S., et al. (2020). Grakel: A graph kernel library in python. *Journal of Machine Learning Research, 21*(54), 1–5.

Skarding, J., Gabrys, B., & Musial, K. (2021). Foundations and modeling of dynamic networks using dynamic graph neural networks: A survey. *IEEE Access, 9*, 79143–79168. https://doi.org/10.1109/access.2021.3082932

Trivedi, R., Farajtabar, M., & Biswal, P., et al. (2019). Dyrep: Learning representations over dynamic graphs. In*International conference on learning representations*. https://openreview.net/forum?id=HyePrhR5KX

Veličković, P., Cucurull, G., & Casanova, A., et al. (2018). Graph attention networks. 1710.10903.

Wang, D., Yu, Y., & Rinaldo, A. (2021). Optimal change point detection and localization in sparse dynamic networks. *The Annals of Statistics, 49*(1), 203–232.

Wang, H., Tang, M., Park, Y., et al. (2013). Locality statistics for anomaly detection in time series of graphs. *IEEE Transactions on Signal Processing, 62*(3), 703–717.

Wang, T., & Samworth, R. J. (2018). High dimensional change point estimation via sparse projection. *Journal of the Royal Statistical Society Series B: Statistical Methodology, 80*(1), 57–83. https://doi.org/10.1111/rssb.12243. arXiv:1606.06246.

Wang, T., Chen, Y., & Samworth, R. (2022). High-dimensional, multiscale online changepoint detection. *Journal of the Royal Statistical Society Series B: Statistical Methodology, 84*, 234–266.

Wang, Y., Chakrabarti, A., & Sivakoff, D., et al. (2017). Fast change point detection on dynamic social networks. In *Proceedings of the 26th international joint conference on artificial intelligence, IJCAI'17* (pp. 2992–2998). AAAI Press.

Wilson, J. D., Stevens, N. T., & Woodall, W. H. (2019). Modeling and detecting change in temporal networks via the degree corrected stochastic block model. *Quality and Reliability Engineering International, 35*(5), 1363–1378. https://doi.org/10.1002/qre.2520. arXiv:1605.04049.

Xu, H., Feng, Y., & Chen, J., et al. (2018). Unsupervised anomaly detection via variational auto-encoder for seasonal KPIS in web applications. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web - WWW '18*. https://doi.org/10.1145/3178876.3185996

Xu, K., Hu, W., & Leskovec, J., et al. (2019). How powerful are graph neural networks? 1810.00826.

Xue, G., Zhong, M., Li, J., et al. (2022). Dynamic network embedding survey. *Neurocomputing, 472*, 212–223. https://doi.org/10.1016/j.neucom.2021.03.138

Yoshida, T., Takeuchi, I., & Karasuyama, M. (2021). Distance metric learning for graph structured data. *Machine Learning, 110*(7), 1765–1811.

Yu, Y., Padilla, O. H. M., Wang, D., et al. (2021). Optimal network online change point localisation. 2101.05477.

Zhang, M., Cui, Z., & Neumann, M., et al. (2018). An end-to-end deep learning architecture for graph classification. In *Proceedings of the AAAI conference on artificial intelligence*.

Zhang, R., Hao, Y., & Yu, D., et al. (2020). Correlation-aware unsupervised change-point detection via graph neural networks. 2004.11934.

Zhao, Q., & Wang, Y. (2019). Learning metrics for persistence-based summaries and applications for graph classification. *Advances in neural information processing systems* (Vol. 32).

Zhao, Z., Chen, L., & Lin, L. (2019). Change-point detection in dynamic networks via graphon estimation. 1908.01823.

Zou, C., Yin, G., Feng, L., et al. (2014). Nonparametric maximum likelihood approach to multiple change-point problems. *The Annals of Statistics*. https://doi.org/10.1214/14-aos1210

Authors and Affiliations

Déborah Sulem[1] · Henry Kenlay[2] · Mihai Cucuringu[1,3,4] · Xiaowen Dong[2]

✉ Déborah Sulem
deborah.sulem@stats.ox.ac.uk

Henry Kenlay
kenlay@robots.ox.ac.uk

Mihai Cucuringu
mihai.cucuringu@stats.ox.ac.uk

Xiaowen Dong
xdong@robots.ox.ac.uk

[1] Department of Statistics, University of Oxford, Oxford, UK

[2] Department of Engineering Science, University of Oxford, Oxford, UK

[3] Mathematical Institute, University of Oxford, Oxford, UK

[4] The Alan Turing Institute, London, UK