Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2023) 31: 876 – 891
© TÜBİTAK
doi:10.55730/1300-0632.4023

Transforming temporal-dynamic graphs into time-series data for solving event detection problems

Kutay TASCI[1*], **Fuat AKAL**[2]
[1]Computer Engineering Department, Bilkent University, Ankara, Turkiye
[2]Computer Engineering Department, Hacettepe University, Ankara, Turkiye

Received: 13.01.2023 • **Accepted/Published Online:** 11.08.2023 • **Final Version:** 29.09.2023

Abstract: Event detection on temporal-dynamic graphs aims at detecting significant events based on deviations from the normal behavior of the graphs. With the widespread use of social media, many real-world events manifest as social media interactions, making them suitable for modeling as temporal-dynamic graphs. This paper presents a workflow for event detection on temporal-dynamic graphs using graph representation learning. Our workflow leverages generated embeddings of a temporal-dynamic graph to reframe the problem as an unsupervised time-series anomaly detection task. We evaluated our workflow on four distinct real-world social media datasets and compared our results with the related work. The results show that the performance depends on how anomalies deviate from normal. These include changes in both size and topology. Our results are similar to the related work for the graphs where the deviation from a normal state of the temporal-dynamic graph is apparent, e.g., Reddit. On the other hand, we achieved a 3-fold improvement in precision for the graphs where deviations exist on size and topology, e.g., Twitter. Also, our results are 20% to 5-fold better even if we introduced some delay factor. That is, we beat our competition while detecting events that occurred some time ago. As a result, our study proves that graph embeddings as time-series data can be used for event detection tasks.

Key words: Event detection, graph representation learning, anomaly detection, Temporal-dynamic graphs

1. Introduction

Temporal-dynamic graphs are continuous sequences of time-stamped snapshots of a given network. With the increased use of social media, they have become more popular in recent years. There are different domains like social, communication, citation, and terrorist networks that can be modeled as temporal dynamic graphs [1, 2]. Detecting abnormal time points, such as change points in such graphs, can provide valuable insights into the underlying data. These change points often correspond to significant events such as critical state changes, anomalies, faults, intrusion, etc., depending on the application domain [3]. Event detection in Temporal-dynamic graphs is a research area that aims to find these change points.

Another example of temporally ordered stream-type data is the time series data. In the time-series anomaly detection problem, the aim is similar to event detection, which is to detect abnormal time points in given time series data. Despite their similarity, the time-series anomaly detection problem has a strong research background [4, 5], when compared to temporal dynamic networks. In this work, we propose a workflow to transform temporal-dynamic graph data into time series data by extracting graph embeddings of the network

*Correspondence: kutay.tasci@bilkent.edu.tr by using Graph Representation Learning. This transformation allows us to leverage a variety of algorithms from time-series anomaly detection to event detection problems. By incorporating time-series anomaly detection algorithms, we aim to expand the event detection literature and enhance the capabilities of event detection on temporal-dynamic graphs.

Graph representation learning shows the effectiveness of embeddings on various tasks [6], such as graph similarity, link prediction, node classification, etc. There are different types of embeddings available for graphs. These are node-level, edge-level, subgraph, and graph embeddings. Although these embeddings can be used successfully in static graphs, many algorithms can not capture the temporal dynamics in temporal-dynamic graphs. A graph representation algorithm, however, can successfully embed temporal-dynamic graphs if it follows the given stability and growing graph conditions proposed in DynGEM [7]. In our study we have used tdGraphEmbed [8] as our graph representation learning algorithm, which can ensure the stability and growing graph conditions.

In our work, we propose a workflow to transform the temporal-dynamic graph data to multi-variate time-series data and solve the event detection problem using unsupervised anomaly detection methods. The existing literature's main shortcoming is the lack of research and implementations for event detection algorithms. Unsupervised time-series anomaly detection has more extensive literature compared to event detection in temporal-dynamic networks. By transforming, we can access more extensive literature and available implementations. Upcoming studies on graph representation learning and time-series anomaly detection can also improve our proposed workflow. Thus, it provides a flexible method compared to previous studies.

The novelty of our proposed workflow is how it addresses the dynamic graph problem of anomaly detection in the time series domain using time graph representation learning and then applying time series anomaly detections. We can summarize our contribution as follows:

- We proposed a workflow that uses graph representation learning to transform temporal-dynamic graphs to time series data and unsupervised time series anomaly detection methods to solve event detection problems.

- Our proposed method is flexible and general. It can be used with different graph representation learning algorithms that ensure the given conditions and different time series anomaly detection algorithms can be used.

- With this work, we show that using temporally ordered n-dimensional graph embeddings as time series data is feasible for anomaly detection problems.

The rest of the paper is organized as follows. Section 2 surveys the related work. Section 3 presents the proposed workflow architecture. Section 4 gives evaluations and discusses the results. Section 5 concludes the paper.

2. Related work

2.1. Event detection

The event detection problem aims to find abnormal changes in a temporally ordered data stream. In our work, we specifically focus on event detection on temporal-dynamic networks. Event detection is a widely researched area that has been active for many years. Several studies in the literature on detecting important events in social media networks [9], most popularly on Twitter networks [10, 11].

Many domains, such as social networks, communication networks, citation networks, and terrorist analysis, can be modeled as temporal dynamic networks [1, 2]. These networks can be represented as nodes and edges of a graph, where new nodes and edges can be added or removed over time. Event detection aims to detect important temporal changes in the graph structure compared to past states. These important points may correspond to critical events and can be informative. Depending on the domain, such events can vary from intrusions to anomalies, or unusual activities in social networks. This problem can be formulated as; **Given** a set of Graphs= $\{ G_1, G_2, ..., G_T \}$, **find** time points t' which $G_{t'}$ differs significantly from $G_{t'-1}$.

There exist important studies such as machine learning-based ensemble models [3] and recently deep learning-based approaches using Transformers [12] and Graph Attention Networks[13]. Also, important work for our study is the tdGraphEmbed [8] study, which uses graph embeddings and graph similarity for event detection. On the other hand, their formulation is similar in terms of expected input and output. Both expect temporally ordered data and produce novel points significantly different from the rest of the data. This study aims to transform this problem from the temporal-dynamic graph domain into the time series domain.

2.2. Graph representation learning

Graph representation learning is the process of generating node/graph-level vector embeddings. The goal is to generate vector embeddings that can be used to analyze graph-structured data. Node-level embeddings generate embeddings for each node in the graph, and similar nodes are closer to each other in vector space. Graph-level embeddings generate an embedding for the whole graph, and similar graphs are closer to one another in vector space. This measure of similarity can vary, depending on the problem. Recent works on graph representation learning proved that they are very effective in graph mining and embedding graphs [6].

These algorithms can be used in static or temporal-dynamic graph contexts. Static node/graph embeddings generate a vector embedding for a given graph snapshot. Temporal-dynamic node/graph embeddings analyze multiple sets of graphs, which are snapshots taken from different time steps. Each graph represents a different time step, and embeddings should be able to capture temporal-dynamic changes over time. For this work, we will focus on graph-level embeddings in the temporal dynamic context.

Two of the fundamental algorithms proposed for static node embeddings are Node2Vec [14] and DeepWalk [15]. These two algorithms are random walk-based methods. They use random walks in depth-first and breadth-first for exploring graph properties. Random walks are also used in the embedding algorithm preferred in this work. Other than this approach, there also exist methods such as matrix factorization[16, 17] and deep learning-based methods [2, 18, 19] for generating static embeddings.

In literature, there are different graph representation learning models which capture temporal changes in dynamic graphs like Dyngraph2vec [20], and DynGEM [7]. However, these methods are designed to use a fixed number of nodes in each time step. In many real-world cases, the number of nodes changes in temporal dynamic graphs over time. This problem is addressed in DyRep [21] and DynamicTriad [22] with better scalability. It is important to note that all of these studies we mentioned above are on node embeddings.

As previously stated, graph embedding is the primary topic of this work. There are available studies for graph embeddings, which mainly focus on static graphs and subgraphs. Studies about unsupervised static graph embeddings can be found in Graph2vec [23], Sub2vec [24], and UGraphEmbed [25]. Finally, for temporal-dynamic graph embedding, there is tdGraphEmbed [8], which embeds each time step of a given temporal dynamic graph into vector embeddings. The model generates embeddings for each static snapshot of the corresponding time step. This embedding process is fully unsupervised. We concentrate on temporal dynamic graph embeddings to progressively embed each graph state throughout a specified period.

2.3. Unsupervised anomaly detection on time series data

The event detection problem aims to find outlier points in temporally ordered data. Event detection on time series and temporal dynamic graph data has similar formulations as anomaly detection. The problem extends to different anomaly types, data types, and applications. In this study, we will focus on unsupervised time series anomaly detection. A wide variety of anomaly detection algorithms exists for this problem [4, 5, 26, 27]. The challenge in this problem is the nature of time series data. Unlike regular data, time-series data has its implicit characteristics. Data in a time series is periodic, seasonal, and erratic. Aside from those, it also has a trend characteristic, which makes it difficult for standard anomaly detection models on time-series data to function and generalize because they would fail to recognize the trend over time and take appropriate action.

Unsupervised methods utilize some fundamental requirements for this problem to be effective. In the TadGAN study, these properties are explained in detail [28]. First, in unsupervised anomalies, we do not have prior knowledge of anomalies. Such models are used to detect anomalies with no prior knowledge. Second, there are no normal baselines for this problem. Models should be trained with data that contains anomalies and normal data. In other words, models should not rely on normal data. Third, the problem comes from the no prior knowledge approach. Because of this, we do not know anything about detected anomalies. Detected points are not always problematic or do not have to indicate problems. Lastly, evaluation is a big problem in this task. Usually, we rely on datasets labeled manually by human experts. However, unlabeled anomalies can always be present in the given dataset.

Unsupervised approaches to the anomaly detection problem can be divided into three basic categories: (i) proximity-based, (ii) prediction-based, and (iii) reconstruction-based methods. Proximity-based methods use the distance between single data points or fixed-length sequences as a similarity measure. Data points that are not similar to others are considered anomalies. Examples of these kinds of studies are; Isolation Forest [29], K-Nearest Neighbor (KNN) [30], Local Outlier Factor [31], and Clustering-Based Local Outlier Factor [32]. These methods use distance as a measure to identify anomalies. Prediction-based methods utilize predictive models for time series data. These models try to predict future data points and use prediction errors to detect anomalies. If the error between the predicted and true value is above the given threshold, it can be classified as an anomaly. Statistical models like ARIMA [33] and FDA [34] can achieve this. On the other hand, machine learning-based methods like Hierarchical Temporal Memory (HTM) [35] and Long Short Term Recurrent Neural Networks (LSTM RNNs) [36] proved to be effective in this problem. Reconstruction-based methods are based on latent structures of generative models. The intuition behind this approach is abnormal points lose information when mapped to lower dimensional spaces. Therefore they cannot be reconstructed successfully. These methods compare original data with reconstructed signal and detect anomalies based on reconstruction error. Auto-encoder-based methods such as Principal Component Analysis (PCA) [37], Auto-Encoder (AE) [38], Variational Auto-Encoder (VAE) [38] and LSTM Encoder-Decoder [39] exist for this problem. Also, recent studies show that generative adversarial networks like MAD-GAN [40] and TadGAN [28] are also effective.

Our work aims to transform an unsupervised temporal dynamic graph event detection problem into a time series domain. Since we are using an unsupervised time series anomaly detection method, understanding this problem's various aspects is important. Some of these studies are only conducted on uni-variate data. Since we will embed our graphs into multi-dimensional vectors, we will focus on studies available for multivariate data.

3. Proposed workflow architecture

3.1. Problem definition

In the event detection task, input is a sequence of T static graph snapshots that form a temporal dynamic graph G = $\{G_1, G_2, ..., G_T\}$. Each static snapshot has edges and vertices defined as $G_t = (V_t, E_t)$. Vertices (nodes) and edges of G_t are represented as V$_t$ and E$_t$, respectively. Each static snapshot models the graph's state in the time interval of [t - Δ_t, t]. Δ_t is a configurable parameter defined as time granularity. Since we will use graph representation learning to embed graphs, static snapshots of the graph can also be represented as n-dimensional embeddings $G_t \rightarrow E_t$. In this context, E_t is the n-dimensional vector embedding of the static graph G_t. That is, we can represent the embedding of a temporal dynamic graph as E=$\{E_1, E_2, ..., E_T\}$.

The problem aims to detect time points t' where G_t' differs significantly from the graph's previous states [0 to $G_{t'-1}$]. These events may correspond to important events like anomalies and important events depending on the domain. Novelty, in our work, we aim to solve this problem in the time series domain using graph representation learning. We can also define our problem as multivariate time series anomaly detection problem, which can be formulated after obtaining embeddings as time points t' which $E_{t'}$ differs significantly from the previous graph embeddings [0 to $E_{t'-1}$]. In short, we aim to detect time step t', different from the previous data points.

3.2. Proposed workflow

In this work, we propose a workflow for the event detection problem. We propose a workflow consisting of a graph embedding algorithm to transform given temporal dynamic graph data to multi-variate time series data. Then, the model feeds this multi-variate time series data to an unsupervised time series anomaly detector. Novelty, in this study, we transform the event detection problem from a temporal dynamic graph to time-series data using graph representation learning. To achieve this, we proposed the workflow given in Figure 1. Our workflow takes a temporal graph as input and outputs anomaly scores for each time step. The workflow has two essential steps. First, the graph representation learning algorithm embeds the temporal dynamic graph into n-dimensional vector embeddings. Second, the model uses these embeddings as time series data and passes it to an unsupervised anomaly detector, which gives the anomaly scores. By using anomaly scores, we can find time step t', which is different from the previous data points.

As mentioned in the previous section, each static snapshot models the graph's state in the time interval of [t - Δ_t, t]. Δ_t is a configurable parameter defined as time granularity. The granularity defines the interval between time steps. The interval between time steps can be monthly, weekly, daily, or hourly. Input consists of temporally ordered static graphs that form a temporal-dynamic graph.

We defined the structure of temporal dynamic graphs in the previous section. The first step in our workflow is to embed temporal graphs into n-dimensional vector embeddings. For this task, we are going to use graph representation learning. But given graph representation learning algorithm must yield two properties for using these embeddings as time-series data. These properties are stability and growth.

Stability: Embeddings generated by static methods are not stable. Embeddings of given subsequent time steps can differ considerably from each other in static methods. Even a static embedding algorithm can give different results for the same graph. A stable graph embedding algorithm should reflect temporal changes stably.

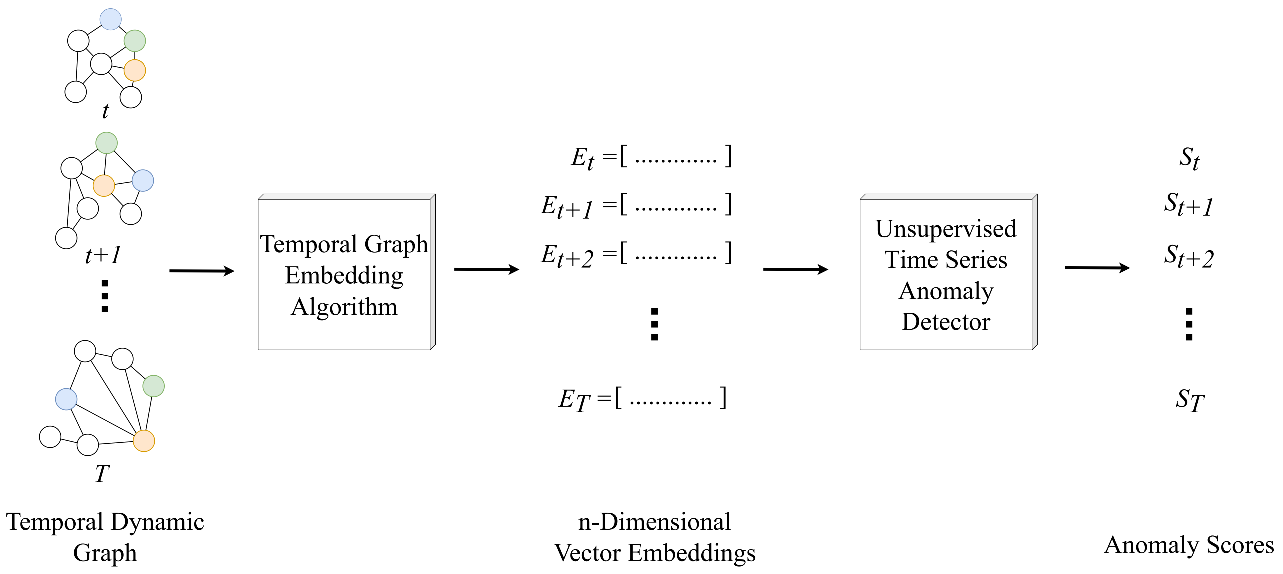

t
t+1
T
Temporal Dynamic Graph
Temporal Graph Embedding Algorithm
E_t =[............]
E_{t+1} =[............]
E_{t+2} =[............]
E_T =[............]
n-Dimensional Vector Embeddings
Unsupervised Time Series Anomaly Detector
S_t
S_{t+1}
S_{t+2}
S_T
Anomaly Scores

Figure 1. In this figure, you can see our proposed model workflow. Input is a temporal-dynamic graph G consisting of static snapshots taken in different time steps. In the first step, the model generates n-dimensional vector embeddings from a given input graph using graph representation learning. After this step, the model passes these embeddings to an unsupervised anomaly detector. The output of the proposed workflow is the anomaly scores corresponding to each time step.

Growth: In a temporal dynamic graph, new nodes and edges can appear and disappear through the graph's lifetime. Most of the previous studies assume that a given graph has a fixed number of nodes. Because of this, they can not handle the growth of the graph.

In our study, we chose the tdGraphEmbed [8] algorithm because, among other graph representation algorithms, tdGraphEmbed holds the properties mentioned above for our task. This algorithm is a random walk-based algorithm. It combines random walks generated with the node2vec [14] algorithm with natural language processing (NLP) algorithms to generate graph embeddings. With these random walks, the algorithm generates a document with each line corresponding to a walk. Then generates embeddings using NLP algorithms designed for document embedding. Thus, documents containing similar walks become closer to each other in vector space, which can be used as a graph similarity metric. As a result, the algorithm provides stable embeddings that allow us to measure the similarity of a given static graph in each timestamp. In addition, document embedding algorithms can process an arbiter amount of inputs, which handles the growth property. In the tdGraphEmbed [8] study, further information about the algorithm, evaluations on graph similarity, and distance-based anomaly detection algorithms are provided.

In the first step of the proposed workflow, the model transforms the temporal dynamic graph data into n-dimensional vector embeddings using the Temporal Graph Embedding Algorithm. These embeddings can be used as multivariate time-series data since the tdGraphEmbed [8] algorithm yields the stability and growth properties we defined.

In the second step of the workflow, unsupervised time-series anomaly detection is performed. Here, we use embeddings generated in the previous step as time-series data. We used the Merlion[41] library in Python for implementation. This library is an open-source machine-learning library for time series. It supports many unsupervised time-series anomaly detection algorithms for univariate and multivariate data. In this study, we have tried different models as anomaly detectors and compared their performance.

Two important assumptions exist for unsupervised anomaly detection problems. The first assumption is that the dataset contains few anomalistic points compared to normal data points. This imbalance is necessary for many unsupervised anomaly detection models. By definition of anomaly, if similar data points are common in a given dataset, models classify that point as normal. The Second assumption is that anomaly points differ greatly from normal data points. Which means their attribute values are considerably different from normal data. The models we used in this study are Isolation Forest [29], Variational Auto-Encoder [38], and LSTM Encoder-Decoder [39].

Isolation Forest is a distance-based anomaly detection algorithm. It detects anomalies by isolation, which measures the distance of a data point to the rest of the data. Isolation forest consists of isolation trees, which build an ensemble model. The model isolates the points that are different from the overall data.

The Variational Auto-Encoder Model is a reconstruction-based anomaly detector. It aims to reconstruct a given time-series signal. Since anomalistic points are rare as compared to normal data. The model overfits to reconstruct a normal signal. Consequently, it fails to reconstruct the given signal. Variational auto-encoders utilize latent space much better than standard auto-encoders, thus providing better performance.

LSTM Encoder-Decoder model is also a reconstruction-based method like auto-encoders. But LSTM Recurrent Neural Networks (RNN) models are built to use time series information of the data. LSTM models have additional learning on utilizing time series information, which provides better use of time series data.

The output of the anomaly detector is the anomaly score at each time step. The time steps of the temporal dynamic graph G and n-dimensional vector embeddings E match, and our graph representation algorithm yield given stability and growth properties. Because of this, we can say that if the anomaly score for time step t is above the anomaly threshold, E_t is considered an anomaly. Consequently, G_t in a temporal dynamic graph G is also an anomaly. This means an anomalistic event in t' differs significantly from the rest of the data.

4. Evaluation

We evaluated our work on four real-world datasets previously used in similar works. Datasets are collected from social media platforms, the ground truths of these datasets correspond to important events. We used these provided ground truths to evaluate our workflow's performance. Also, we compared our quantitative results with previous similar works.

There are two studies we have used as a baseline for our work. The first study proposes an ensemble approach to the event detection problem. Four different approaches (SelectV, SelectH, Full, DivE [42]) are used in the study [3]. These are ensemble models that contain different models. The second study provides the tdGraphEmbed[8] model, the graph embedding algorithm we used in our workflow. In this study, anomaly detection is done by comparing the similarity of the given time steps.

Our evaluations aim to show that the proposed workflow can successfully transform temporal-dynamic graph data to time series data and solve the event detection problem using unsupervised time series anomaly detection methods.

4.1. Datasets

In this study, we used four different real-world datasets to evaluate our proposed workflow. They are called the TwitterSecurity[3], TwitterWorldCup[3], Reddit ”Game Of Thrones”[8], and Reddit ”Formula 1”[8] datasets.

Datasets consist of edges with time stamps, which will be used to generate temporal dynamic graphs. The datasets used were taken from papers [3, 8]. These datasets originate from Twitter and Reddit social media platforms.

The TwitterSecurity[3] dataset is collected from Twitter and contains interactions between tweets, hashtags, mentions, and users as edges. The dataset contains information gathered for four months (May 12–Aug 1, 2014). Tweets are filtered by words in Department of Homeland Security keywords related to terrorism and domestic security. This is an entity-entity co-mention temporal graph. The dataset has a total of 80 data points; 21 of these data points correspond to important events. Ground truths contain major world news of 2014, like the Türkiye mine accident, the Boko Haram kidnapping of school girls, and killings during Yemen raids, etc.

The TwitterWorldCup[3] dataset is also gathered from Twitter API, and data is collected according to the World Cup 2014 season (June 12–July 13) and filtered by popular and official hashtags like #worldcup, #fifa, and #brazil. This is also an entity-entity co-mention temporal graph. Ground truths are based on important events like injuries, goals, and penalties. The dataset has a total of 1791 data points; 119 of these data points correspond to important events. In this dataset, some events with uncertain time stamps in ground truths exist. We removed them in our work since the lack of information prevents us from evaluating the results. Also, granularity is hourly in this dataset, so overlapping events are counted as one ground truth.

The Reddit Game-Of-Thrones[8] dataset is derived from the 'Game of Thrones' subreddit. They used interactions between users who replied to other users' posts as links. The dataset spans 62 days, from July 1, 2017, through August 31, 2017. Ground truths are the release dates of new episodes. Thus, we have 7 anomaly points among 62 data points.

The Reddit Formula-1[8] dataset is derived from the 'Formula 1' subreddit. The dataset spans 61 days, from March 1, 2019, through April 30, 2019. Ground truths are important race dates. Thus, we have 4 anomaly points among 61 data points

4.2. Data preparation

For each dataset, we have an edge list with timestamps. With this data, we can construct static snapshots of the temporal-dynamic graph. First, we divide our data into chunks with respect to time granularity. For each static snapshot, we construct the graph by updating $G_t - 1$ with the edges in the interval [t - Δ_t, t]. Time granularity can be hourly, daily, monthly, etc. Second, we drop nodes with degrees less than a given threshold. Although this process reduces performance, it also reduces the computational cost. Then we obtain a list of temporally ordered static snapshots of a given temporal-dynamic graph $G = \{ G_1, G_2, ..., G_T \}$.

4.3. Experimental setup

In the following experiments, we are going to use our proposed workflow. The workflow's time complexity depends on the preferred algorithms in each step. After pre-processing our data, the first step is to generate graph embeddings. We used the tdGraphEmbed [8] model for this task. The model generates 40 random walks for each node in the graph, and the length of each walk is 16 in our experiments. The model is trained in 50 iterations, with generated a random-walk document. In the second step, we use the time-series anomaly detectors mentioned before. For these algorithms, we used the Merlion[41] machine learning library for time series data. The time complexity of this step is dependent on the preferred algorithm. We implemented our workflow and algorithms with Python. Our source codes are available on GitHub[1]. Experiments were conducted on a computer with an NVIDIA RTX3060Ti GPU with 8GB of GDDR6 memory and an AMD Ryzen 5 5600X CPU with 16GB of DDR4 memory.

4.4. Evaluation metrics

We quantified our workflow performance with the metrics of Precision and Recall. In many real-world applications, abnormal points are much rarer than normal points. That is, precision and recall help to identify the performance of our anomaly detection.

In practice, ground truth and the event's effect on data can have delays. That is, some amount of delay should be tolerable. In some datasets, detecting an event that occurred at point t in the interval of $[t - Delay, t + Delay]$ is counted as accurate. Especially on social media networks, the event's impact is usually reflected in social media after some time. Also, the event's impact can continue for some time after the event, and more importantly, this impact can peak anytime after the event. The delay factor helps us to evaluate our performance by considering these possible effects.

In the evaluation, we counted the Top K data points with the highest anomaly scores as an anomaly. In each experiment, top K data points are counted as an anomaly, and precision-recall metrics are calculated concerning this assumption. We have experimented with different values of K for a better comparison. Because of this, we provide our metrics in the form of $P@K$ and $R@K$, which mean the precision and recall values for the selected K value, respectively.

4.5. Baselines

As mentioned above, we use previous works on given datasets as a baseline. For Twitter datasets, there are three ensemble models [3] as a baseline to these datasets. On the other hand, for Reddit datasets, we are going to use the tdGraphEmbed [8] model to compare our quantitative results.

The models used for Twitter datasets are ensemble models that consist of different algorithms and models. Two of our baseline models, SelectV and SelectH, are ensemble models proposed in [3]. SelectV models use the proposed SELECT algorithm with vertical selection, and the SelectH algorithm uses horizontal selection. Full and DivE [42] algorithms are, on the other hand, given as a baseline in the study.

TdGraphEmbed[8] is proposed as a graph embedding algorithm in the study, but it is also evaluated on event detection problems. In that study, the algorithm is compared with other embedding algorithms using the distance between time steps as an anomaly measure. The TdGraphEmbed algorithm claims to perform best among these algorithms. Because of this, we only used TdGrahphEmbed for comparison.

4.6. Results

In this section, we quantified our workflow's performance by comparing our results with the results provided in the previous studies that used similar methods for comparison.

Table 1 provides the results for Reddit datasets. Precision and recall metrics are provided for given top K anomalies, and the model with the superior performance is highlighted for each metric. For instance, p@5

[1]https://github.com/KutayTasci/TDG2TS-Event-Detection-Workflow

in the table shows the percentage of true positives among the top 5 detections. These datasets are relatively simple, and the previous study already provides the maximum available performance for given datasets.

Our proposed workflow yields the same performance metrics in the Game of Thrones dataset as the previous study. These metrics are also the highest possible performance on this dataset. LSTMED and VAE time series anomaly detectors can detect all seven ground truths. Our proposed workflow can achieve state-of-the-art performance for this particular dataset.

Table 1. Results for the event detection task for Reddit datasets. Top score for each model is highlighted.

	Reddit - Game of Thrones						Reddit - Formula1					
	p@5	r@5	f1@5	p@10	r@10	f1@10	p@5	r@5	f1@5	p@10	r@10	f1@10
tdGraphEmbed[8]	**1**	**0.71**	**0.83**	**0.7**	**1**	**0.82**	**0.8**	**1**	**0.88**	**0.4**	**1**	**0.83**
Our Workflow	**1**	**0.71**	**0.83**	**0.7**	**1**	**0.82**	0.6	0.75	0.66	0.3	0.75	0.42

In the Formula1 dataset, our proposed workflow yields lower performance than the previous study. Both LSTMED and VAE models can achieve results on the metrics. The 25% performance deterioration is because our workflow missed one of the ground truths. This result is interesting because the graph embedding algorithm for our workflow is the tdGraphEmbed algorithm which yields higher performance by just using point-to-point distance-based similarity. The previous study also achieved the maximum possible performance for this dataset.

In table 2, you can see our workflows and the previous studies' performance metrics for Twitter datasets. Twitter datasets are much closer to real-world use cases. Recall metrics were unavailable for the previous study, so we compared our workflow with different setups. In the previous study, four different ensemble models were used, i.e. SELECT-H, SELECT-V, FULL, and DivE. These ensemble models use the same learning algorithms with different strategies. Our workflow's precision is consistent with different parameters of $@K$.

Table 2. Results for the event detection task for Twitter datasets. The top score for each model is highlighted. F1 Scores and Recall values for baseline models were not available.

	Twitter - Security			**Twitter - WorldCup**		
	Precision	Recall	F1 Score	Precision	Recall	F1 Score
SELECT-H [3]	0.58	-	-	0.16	-	-
SELECT-V [3]	0.55	-	-	0.22	-	-
FULL [3]	0.52	-	-	0.18	-	-
DivE [3]	0.48	-	-	0.18	-	-
Our Workflow@5	**0.6**	0.14	0.22	0.6	0.025	0.048
Our Workflow@10	0.5	0.23	0.31	0.6	0.04	0.075
Our Workflow@20	0.5	**0.47**	**0.48**	0.65	0.1	0.17
Our Workflow@30	-	-	-	**0.66**	**0.16**	**0.25**

Before going into the analysis, we will remind some of the problems of event detection on these datasets. The first problem is data imbalance; most of the data for real-world event detection problems are imbalanced. This imbalance is a good thing for unsupervised anomaly detection methods. In our datasets, the Twitter-WorldCup dataset reflects this property better than the Twitter-Security dataset as data imbalance goes up, our workflow performance increases. Because we rely on unsupervised time-series anomaly detection algorithms. The second problem is our ground truths are not perfect. Our ground truths yield information about the exact time an event happened. Conversely, we do not know the precise time and duration of the event's effect on the network. Because of this property, we also analyzed the dataset with a delay factor in Figure 2.

For the Twitter-Security as you can see in Table 2, our proposed workflow has similar precision metrics compared to previous studies. Since we did not have recall metrics for previous studies, we tried different setups for our tests. As you can see, our precision is stable when we increase the number of predictions. On the other hand, our recall values increased as expected. In initial metrics with $Delay = 0$, only @5 has increased by 3.5% compared to the previous best model SELECT-H. But, in this dataset delay factor is important, as you can see in Figure 2. Our precision performance increases with the delay factor. Our proposed model immediately starts to perform better than previous models. For instance, according to the top left chart in Figure 2, our model yields 1.0 as precision for @5 and @10 when $Delay = 1$, while the closest related work, i.e. SELECT-H yield 0.9. Precision values of P@5 and P@10 have a 10% improvement compared to the previous study. When compared to the previous best model SELECT-H. Also, the precision value P@20 yields the same performance compared to SELECT-H, with a recall value of 80%.

Our preferred unsupervised time-series anomaly detection algorithm for the Twitter-Security dataset is the Isolation Forest algorithm. This algorithm yields better performance compared to other algorithms. The reason behind this is the ratio between normal and abnormal data points. Our experiment shows the importance of data imbalance in our proposed workflow.

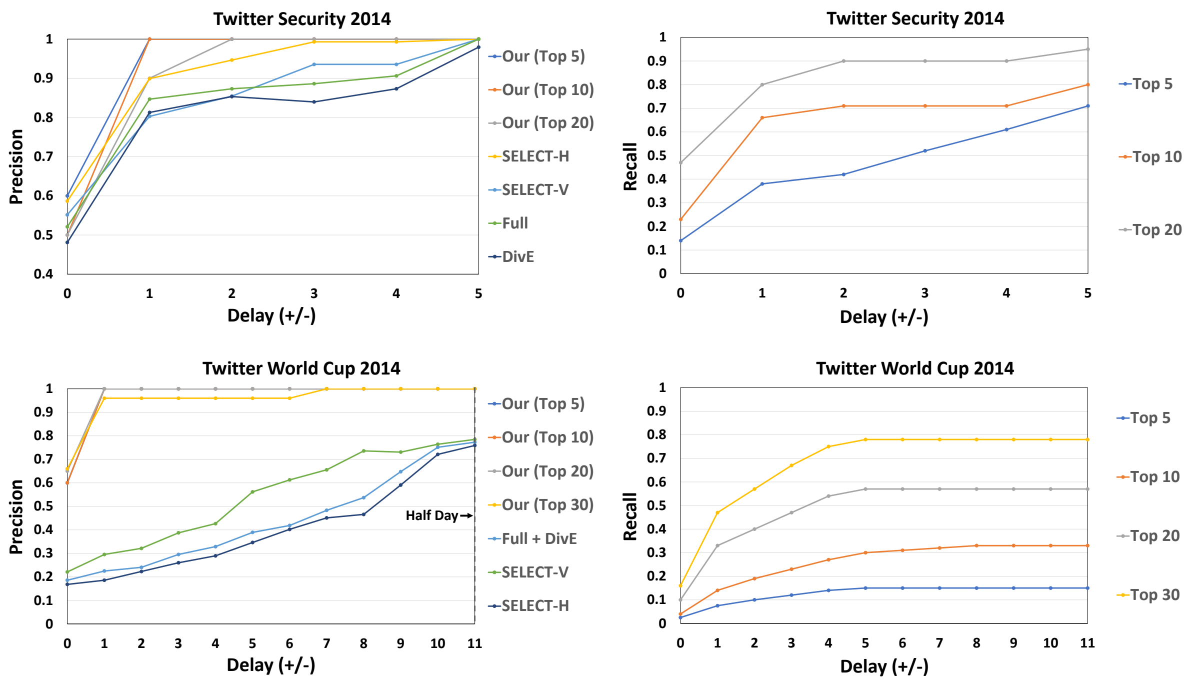
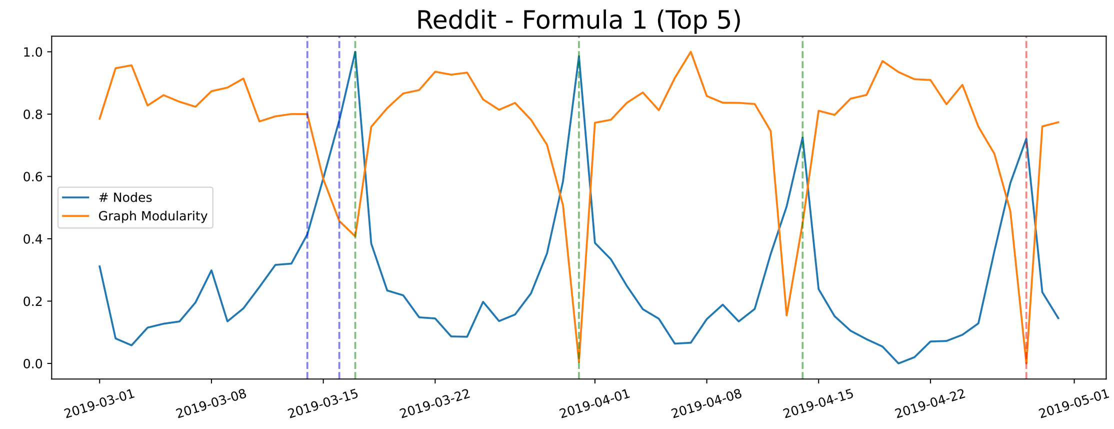


Figure 2. Results for the event detection task for Twitter datasets with delay factor. Recall values for baseline models were not available.

For the Twitter-WorldCup dataset, you can see in Table 2 that our workflow performed significantly better performance when compared to previous models. For instance, our model yields 0.66 precision while the closest related work, i.e. SELECT-V, yields only 0.22. Our model provided a threefold performance improvement compared to previous studies. This dataset is much larger than previous ones. Normal and abnormal data ratio decreases with the number of data points. Since we rely on data imbalance, this increases our performance on this dataset. Like the previous example, our precision is stable, and our recall goes higher with the number of predictions.

In this dataset, our data granularity is hourly. Because of this, the delay factor becomes more important in this case. As you can see in Figure 2, our model can make accurate detections in very small intervals compared to other models. For instance, according to the bottom left chart in Figure 2, our workflow's precision values for $p@5$, $p@10$, and $p@20$ are 100% when $Delay = 1$. This means all of our detections are true in the interval of $[t-1,\ t+1]$. Also, our preferred unsupervised time series anomaly detection model for the Twitter-WorldCup dataset is the LSTMED model. Because of data granularity, the dataset provides the data imbalance property and sparse distribution of abnormal data points, and the LSTMED model is the best fit for this dataset.



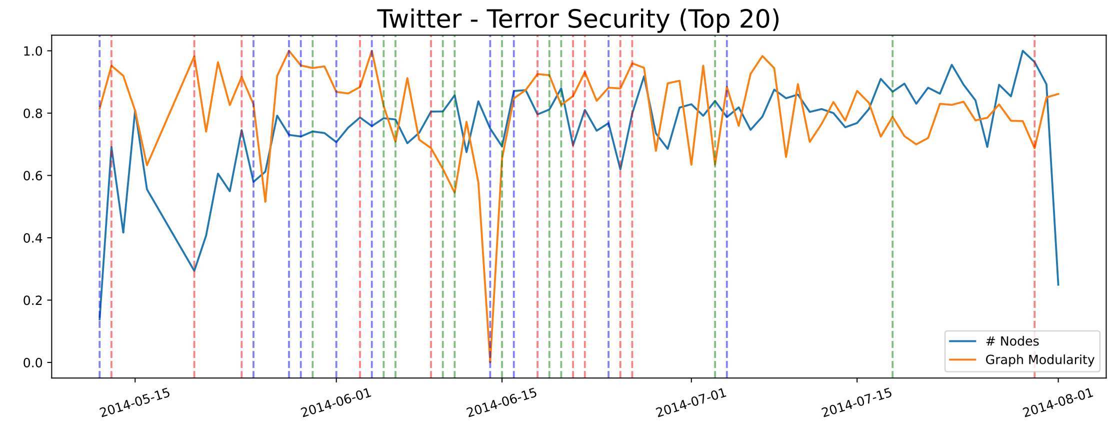


Figure 3. Graphical representations of the anomalies in datasets based on the number of nodes and modularity of the static graph in each timestamp. All values are normalized between 0 and 1. Green, blue, and red lines indicate true positives, false positives, and false negatives, respectively.

So far, we have evaluated the performance of our proposed workflow without analyzing the anomalies present in the dataset and the anomalies detected. We have used two metrics for this task to analyze the temporal dynamic graph at each timestamp. We used the number of nodes in the static snapshot for graph size. Other metrics, such as the number of edges or graph density, are correlated with the number of nodes. Therefore, we stick to one metric for simplicity. For reflecting graph topology, we used the modularity metric. Modularity metric measures the level of separation of the connections between communities in the graph. For example, a graph with high modularity means communities or clusters are distinct and clear, while low modularity means communities are fuzzy and overlapping. With this metric, we can analyze the topological state of the graph. We have visualized these metrics in Figure 3 on the Reddit Formula-1 and the TwitterSecurity dataset.

In Figure 3, you can see the visualization of the anomalies detected and anomalies present in two of the datasets. In the Figure, green, blue, and red lines indicate true positives, false positives, and false negatives, respectively. Reddit Formula-1 dataset clearly shows the effect of the graph size in the event detection process. Labeled anomalies in this dataset point out the timestamps that graph size has peaked in a time frame, which is labeled after the four different race days. Our model successfully detects three of these points, but the model has two false positive detections. These two detections happened to be just before the first race on March 17, 2019. Thus, it is possible that our model detected the community's preparation as an anomaly with a higher anomaly score than the final race.

To understand the effects of topological changes, we can analyze the Twitter Security dataset in Figure 3. Events in this dataset are unpredictable, such as earthquakes, mining accidents, or terrorist attacks. It is crucial to remember that, in this dataset, the event's effect often can be shown after the actual timestamp and can have a more prolonged impact on social media. In Figure 3, you can see the anomalies depend on the size and topology. Our model in this dataset captured significant changes in size, modularity, or both. For example, the anomaly on July 3, 2014 is a good example where a change in graph modularity rather than graph size. Also, you can see the false positive on June 14, 2014, there exists an important change in graph topology without ground truth. Both show that our model is sensitive to topological changes, and abnormal points exist in the dataset with no ground truths.

5. Conclusion

In this study, we proposed a workflow for solving event detection problems on temporal-dynamic graphs. The idea behind our proposed model was to transfer this problem into the time-series domain and solve the problem with unsupervised time-series anomaly detection methods. We proposed to use graph representation learning to transform the temporal-dynamic graph data into time-series data. We also provided the necessary properties of the graph representation learning algorithm for using generated embeddings as time-series data.

Our evaluations and experiments on real-world datasets prove that our proposed workflow effectively solves event detection problems on temporal-dynamic graph data. Also, our bench-marking results show that our workflow yields competitive performance compared to previous studies. For instance, we have a threefold performance improvement on the Twitter-World-Cup dataset and produce close results on Twitter-Security and Reddit-GameOfThrones datasets. Our quantitative results provide proof of concept for the effectiveness of our proposed workflow.

In our experiments, we observed that there are two major limitations. First, as we mentioned, an event's delay factor effect can continue for several timestamps. Since our process is fully unsupervised, our workflow cannot distinguish the difference between consecutive anomalies. Thus, the model can only detect the presence of an anomaly. Secondarily, transforming the problem into an unsupervised time-series anomaly detection problem brings along the requirements of these algorithms. An example of this can be seen in the Twitter-Security dataset, where the required amount of data imbalance is not achieved, and the delay factor is highly effectual. Consequently, our workflow does not provide significant improvement on top of previous works.

Considering that our workflow is using existing models as components. Our workflow uses existing models as black-box components. That is, upcoming studies about temporal-dynamic graph embedding algorithms and unsupervised time-series anomaly detection algorithms can improve our workflow's performance. Detailed research on the existing and new models along with different datasets can provide a better understanding of detected abnormal events and our workflow's behavior on various events. In this study, we have focused on the anomaly detection task. However, applications on different tasks with transforming temporal-dynamic graphs into time-series data seem feasible. In addition, because the tdGraphEmbed model can process streaming graphs, experimenting with streaming graph data with our approach is an open research topic.

Acknowledgment

The authors have no conflicts of interest in this article. No funding was received for conducting this study. We want to thank Mert Aksu, Arcin Ulku Erguzen for their support in this study.

References

[1] Aggarwal C, Subbian K. Evolutionary network analysis: A survey. ACM Computing Surveys (CSUR) 2014; 47 (1): 1– 36. doi:10.1145/2601412

[2] Goyal P, Ferrara E. Graph embedding techniques, applications, and performance: A survey. Knowledge-Based Systems 2018; 151: 78-94. doi:10.1016/j.knosys.2018.03.022

[3] Rayana S, Akoglu L. Less is more: Building selective anomaly ensembles. ACM Transactions on Knowledge Discovery from Data (TKDD) 2016; 10 (4): 1-33. doi:10.1145/2890508

[4] Chandola V, Banerjee A, Kumar V. Anomaly detection: A survey. ACM Computing Surveys (CSUR) 2009; 41 (3): 1-58. doi:10.1145/1541880.1541882

[5] Hodge V, Austin J. A survey of outlier detection methodologies. Artificial Intelligence Review 2004; 22: 85-126. doi:10.1023/B:AIRE.0000045502.10941.a9

[6] Chen F, Wang YC, Wang B, Kuo CC. Graph representation learning: A survey. APSIPA Transactions on Signal and Information Processing 2020; 9: e15. doi:10.1017/ATSIP.2020.13

[7] Goyal P, Kamra N, He X, Liu Y. Dyngem: Deep embedding method for dynamic graphs. arXiv preprint 2018. doi:10.48550/arXiv.1805.11273

[8] Beladev M, Rokach L, Katz G, Guy I, Radinsky K. tdGraphEmbed: Temporal dynamic graph-level embedding. In: Proceedings of the ACM International Conference on Information & Knowledge Management; 2020. pp. 55-64. doi:10.1145/3340531.3411953

[9] Cordeiro M, Gama J. Online social networks event detection: A survey. Solving Large Scale Learning Tasks. Challenges and Algorithms: Essays Dedicated to Katharina Morik on the Occasion of Her 60th Birthday 2016; 1-41. doi:10.1007/978-3-319-41706-6_1

[10] Atefeh F, Khreich W. A survey of techniques for event detection in twitter. Computational Intelligence 2015; 31 (1): 132-64. doi:10.1111/coin.12017

[11] Hasan M, Orgun MA, Schwitter R. A survey on real-time event detection from the Twitter data stream. Journal of Information Science 2018; 44 (4): 443-63.

[12] Liu Y, Pan S, Wang YG, Xiong F, Wang L et al. Anomaly detection in dynamic graphs via transformer. IEEE Transactions on Knowledge and Data Engineering 2021. doi:10.1109/TKDE.2021.3124061

[13] Ding C, Sun S, Zhao J. MST-GAT: A multimodal spatial–temporal graph attention network for time series anomaly detection. Information Fusion 2023; 89: 527-36. doi:10.1016/j.inffus.2022.08.011

[14] Grover A, Leskovec J. node2vec: Scalable feature learning for networks. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining; San Francisco, California, USA; 2016, pp. 855-864. doi:10.1145/2939672.2939754

[15] Perozzi B, Al-Rfou R, Skiena S. Deepwalk: Online learning of social representations. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining; New York, USA; 2014, pp. 701-710. doi:10.1145/2623330.2623732

[16] Belkin M, Niyogi P. Laplacian eigenmaps and spectral techniques for embedding and clustering. Advances in Neural Information Processing Systems 2001; 14.

[17] Roweis ST, Saul LK. Nonlinear dimensionality reduction by locally linear embedding. Science 2000; 290 (5500): 2323-6. doi:10.1126/science.290.5500.232

[18] Cao S, Lu W, Xu Q. Deep neural networks for learning graph representations. In: Proceedings of the AAAI conference on artificial intelligence; 2016, Vol. 30: No. 1. https://doi.org/10.1609/aaai.v30i1.10179

[19] Wang D, Cui P, Zhu W. Structural deep network embedding. In: Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining; 2016, pp. 1225-1234. https://doi.org/10.1145/2939672.2939753

[20] Goyal P, Chhetri SR, Canedo A. dyngraph2vec: Capturing network dynamics using dynamic graph representation learning. Knowledge-Based Systems 2020; 187: 104816. doi:10.1016/j.knosys.2019.06.024

[21] Trivedi R, Farajtabar M, Biswal P, Zha H. Dyrep: Learning representations over dynamic graphs. In: International Conference on Learning Representations; New Orleans, USA; 2019. doi:10.48550/arXiv.1803.04051

[22] Zhou L, Yang Y, Ren X, Wu F, Zhuang Y. Dynamic network embedding by modeling triadic closure process. In: Proceedings of the AAAI Conference on Artificial Intelligence; New Orleans, USA; 2018. 32:1. doi:10.1609/aaai.v32i1.11257

[23] Narayanan A, Chandramohan M, Venkatesan R, Chen L, Liu Y et al. graph2vec: Learning distributed representations of graphs. arXiv preprint 2017. doi:10.48550/arXiv.1707.0500

[24] Adhikari, B., Zhang, Y., Ramakrishnan, N., Prakash, B.A. Sub2Vec: Feature Learning for Subgraphs. In: 22nd Pacific-Asia Conference; Melbourne, VIC, Australia; 2018, pp. 170-182. doi:10.1007/978-3-319-93037-4_14

[25] ai Y, Ding H, Qiao Y, Marinovic A, Gu K et al. Unsupervised inductive graph-level representation learning via graph-graph proximity. arXiv preprint 2019. doi:10.48550/arXiv.1904.01098

[26] Goldstein M, Uchida S. A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data. PloS one 2016; 11 (4): e0152173. doi:10.1371/journal.pone.0152173

[27] Habeeb RA, Nasaruddin F, Gani A, Hashem IA, Ahmed E et al. Real-time big data processing for anomaly detection: A survey. International Journal of Information Management 2019; 45:289-307. doi:10.1016/j.ijinfomgt.2018.08.006

[28] Geiger A, Liu D, Alnegheimish S, Cuesta-Infante A, Veeramachaneni K. Tadgan: Time series anomaly detection using generative adversarial networks. In: IEEE International Conference on Big Data; Virtual Conference; 2020, pp. 33-43. doi:10.1109/BigData50022.2020.9378139

[29] Pan R, Zhou Y, Cao B, Liu NN, Lukose R et al. One-class collaborative filtering. In: 8th IEEE International Conference on Data Mining; Pisa, Italy; 2008, pp. 502-511. doi:10.1109/ICDM.2008.16

[30] Angiulli F, Pizzuti C. Fast outlier detection in high dimensional spaces. In: 6th European Conference; Helsinki, Finland; 2002, pp. 15-27. doi:10.1007/3-540-45681-3_2

[31] Breunig MM, Kriegel HP, Ng RT, Sander J. LOF: identifying density-based local outliers. In: Proceedings of the ACM SIGMOD International Conference on Management of Data; Dallas, Texas, USA; 2000, pp. 93-104. doi:10.1145/342009.335388

[32] He Z, Xu X, Deng S. Discovering cluster-based local outliers. Pattern Recognition Letters 2003; 24 (9-10): 1641-50. doi:10.1016/S0167-8655(03)00003-5

[33] Pena EH, de Assis MV, Proença ML. Anomaly detection using forecasting methods arima and hwds. In: 32nd International Conference of the Chilean Computer Science Society; Temuco, Cautin, Chile; 2013, pp. 63-66. doi:10.1109/SCCC.2013.18

[34] Torres JM, Nieto PG, Alejano L, Reyes AN. Detection of outliers in gas emissions from urban areas using functional data analysis. Journal of Hazardous Materials 2011; 186 (1): 144-9. doi:10.1016/j.jhazmat.2010.10.091

[35] Ahmad S, Lavin A, Purdy S, Agha Z. Unsupervised real-time anomaly detection for streaming data. Neurocomputing 2017; 262: 134-47. doi:10.1016/j.neucom.2017.04.070

[36] Hundman K, Constantinou V, Laporte C, Colwell I, Soderstrom T. Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining; New York, USA; 2018, pp. 387-395. doi:10.1145/3219819.3219845

[37] Ringberg H, Soule A, Rexford J, Diot C. Sensitivity of PCA for traffic anomaly detection. In: Proceedings of the 2007 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems; San Diego, California, USA; 2007, pp. 109-120. doi:10.1145/1254882.1254895

[38] An J, Cho S. Variational autoencoder based anomaly detection using reconstruction probability. Special lecture on IE 2015; 2 (1): 1-8.

[39] Malhotra P, Ramakrishnan A, Anand G, Vig L, Agarwal P et al. LSTM-based encoder-decoder for multi-sensor anomaly detection. arXiv preprint; 2016. doi:10.48550/arXiv.1607.00148

[40] Li D, Chen D, Jin B, Shi L, Goh J et al. MAD-GAN: Multivariate anomaly detection for time series data with generative adversarial networks. In: 28th International Conference on Artificial Neural Networks; Munich, Germany; 2019, pp. 703-716. doi:10.1007/978-3-030-30490-4_56

[41] Bhatnagar A, Kassianik P, Liu C, et al. Merlion: A machine learning library for time series. arXiv preprint; 2021. doi:10.48550/arXiv.2109.09265

[42] Schubert E, Zimek A, Kriegel HP. Local outlier detection reconsidered: a generalized view on locality with applications to spatial, video, and network outlier detection. Data Mining and Knowledge Discovery 2014; 28: 190-237. doi:10.1007/s10618-012-0300-z