Detecting Anomalous Events in Object-centric Business Processes via Graph Neural Networks

Alessandro Niro and Michael Werner

University of Amsterdam, Amsterdam, The Netherlands
{a.niro, m.werner}@uva.nl

Abstract. Detecting anomalies is important for identifying inefficiencies, errors, or fraud in business processes. Traditional process mining approaches focus on analyzing 'flattened', sequential, event logs based on a single case notion. However, many real-world process executions exhibit a graph-like structure, where events can be associated with multiple cases. Flattening event logs requires selecting a single case identifier which creates a gap with the real event data and artificially introduces anomalies in the event logs. Object-centric process mining avoids these limitations by allowing events to be related to different cases. This study proposes a novel framework for anomaly detection in business processes that exploits graph neural networks and the enhanced information offered by object-centric process mining. We first reconstruct and represent the process dependencies of the object-centric event logs as attributed graphs and then employ a graph convolutional autoencoder architecture to detect anomalous events. Our results show that our approach provides promising performance in detecting anomalies at the activity type and attributes level, although it struggles to detect temporal anomalies.

Keywords: Object-centric Process Mining · Graph Neural Networks · Anomaly Detection

1 Introduction

Process mining aims to discover, monitor, and enhance existing business processes by leveraging data traces generated by their execution and stored into event logs [2]. Business processes can be defined as a set of activities that enable the organization to achieve a specified goal. They are seldom friction-less. Errors, inefficiencies, and fraud during process executions can lead to significant losses for the organizations. The ability to detect and mitigate harmful anomalies is crucial for maintaining the effectiveness and efficiency of business operations.

Traditional approaches to process mining rely on 'flattened' event logs. Events are characterized by a single case identifier [1] and process instances are assumed to be strictly ordered sequences of events. Anomaly detection techniques have mostly focused on approaches applied to flattened event logs. Conformance checking approaches [10,9,12] focus on the detection of deviations of process instances, as captured by the event logs, from an *a priori* process model, which

is a necessary input for this type of techniques. Machine learning approaches, based on distance measures [21,30] or reconstruction errors [27,28,26,20], have been focused on detecting anomalies directly from event logs, adopting methods suited for strictly ordered sequences of events.

However, in real-world processes there usually exist multiple potential identifiers. A single case notion leads to a loss of information (i.e. *deficiency*, *convergence* and *divergence* issues [2]). Object-centric process mining [18] is an emerging paradigm that drops the single case assumption and instead assumes events can be associated to any number of objects (cases) of different types, with the aim to overcome the limitations of traditional approaches and provide a more accurate depiction of the actual process. Compared to the strictly ordered linear structure of process instances resulting from the single case notion, object-centric process instances can be naturally represented as directed graphs [5].

This study introduces an approach for anomaly detection in business process that is natively designed for object-centric event logs. We propose an unsupervised machine learning approach based on GNNs, and specifically on a graph convolutional autoencoder (GCNAE) [25,33] architecture.

The approach is illustrated in Figure 1. It first reconstructs the dependencies between events within an object-centric event log as a set of attributed graphs, then it uses the GCNAE to compute the events' anomaly scores. We employ a simple heuristic based on the inter-quartile range (IQR) to automatically set the threshold and label the anomalies without the need of prior knowledge of the contamination rate of the data.

The main contribution of this study is the introduction of a novel unsupervised anomaly detection approach for business processes, leveraging GNNs and the enriched event data structure offered by object-centric process mining. We are not aware of any other studies that employ GNN on object-centric events logs for anomaly detection. Our approach does not rely on prior information about the process model, contamination rate, or a clean training set, making it suitable for real-world applications.

We evaluated the performance of our approach on two different, publicly available[1] object-centric event logs: one a synthetic dataset and the other a real-life dataset. We measured the performance of the approach across different metrics by injecting various types of anomalies in the event logs. The evaluation results demonstrate that our approach performs well in regard to events activity type and attribute anomalies. They also showed limitations regarding the detection of structural and temporal process anomalies which requires further exploration of GNN architectures that can better detect such anomalies.

2 Preliminaries and Background

Object-centric Process Mining. Traditional approaches to process mining involve 'flattened' event logs that are based on the assumptions of a single case

[1] Event logs and source code are available at: github.com/niro-a/DAEiOcBPvGNN

notion and of each event to be associated to exactly one case [1]. This leads to a gap between the real event data and the event log, and specifically to the issues of *deficiency* (deletion of events), *convergence* (duplication of events) and *divergence* (ordering unrelated events) [1]. An object-centric event log [18] is a collection of events where each event is associated with one or more objects, which may be of different types. Similarly to traditional event logs, each event is also associated to an activity, timestamp, and additional attributes.

Definition 1 (Object-centric Event Log). *Let T be the universe of timestamps. An object-centric event log $L = (E, O, OT, A, AV, \pi_{type}, \pi_{time}, \pi_{trace}, \pi_{act}, \pi_{attr})$ is a tuple where:*

- *E is a set of events, O is a set of objects, OT is a set of object types, A is a set of activities and AV is a set of attribute values,*
- *π_{type}: $O \to OT$ maps each object to an object type,*
- *π_{time}: $E \to T$ maps each event to a timestamp,*
- *π_{trace}: $O \to E^*$ maps each object to a temporally ordered sequence of events,*
- *$\pi_{act} : E \to A$ maps each event to its activity,*
- *$\pi_{attr} : E \nrightarrow AV$ maps each event onto attributes values.*

By modeling the relationship between events and multiple objects of different types, object-centric event logs exhibit a graph structure [8] and also the traditional concepts of cases (i.e. process instances) and variants have been extended, in the object-centric setting, from sequences to graphs [5].

Definition 2 (Object-centric Process Instance). *Let L be an object-centric event log. Given all the temporally ordered traces of events $\pi_{trace}(o_i) = \langle e_1, \ldots, e_n \rangle$ associated to each object $o_i \in L$, an object-centric process instance $P = (E', D)$ is a directed graph with nodes E' (representing events), edges D (representing the events temporal dependencies) for a set of traces joined directly or transitively by one or more common events.*

Given this definition, an object-centric event log can be reconstructed as a set of one or more process instances, where process instances are made by set of traces that are connected by common bridge events. This representation is free of convergence, deficiency, or divergence issues [3] and is equivalent to the *connected component process execution* found in [5]. It represents a generalization of the traditional case concept to object-centric event logs.

Problem Statement. While most of the traditional process mining literature has focused on detecting anomalous process instances (i.e. cases), we move our focus to detecting anomalous events since, in the context of object-centric process mining, process instances can be overly complex and large, to the extreme of comprising one single instance for the whole event log [5].

Unlike some of the previous machine learning approaches that characterized anomaly detection as a semi-supervised task [21,24,26] which assumes the availability of a suitable labeled dataset of normal behavior, we characterize anomaly detection as an unsupervised task. We further impose the requirement for the algorithms to explicitly discriminate the anomalous events from the normal ones.

Definition 3 (Event Anomaly Detection). *Event anomaly detection is the task of identifying events that deviate significantly from normal behavior in a given object-centric event log. Formally, given an object-centric event log L, let $E_n \subseteq E$ be the set of normal events and $E_a \subset E$ be the set of anomalous events. The goal of event anomaly detection is to learn a function $f : E \to \{0, 1\}$ that assigns a binary label to each event $e_i \in E$ indicating whether it is anomalous or not, based only on the knowledge of L.*

3 Related Work

Anomaly Detection in Business Processes. The task of anomaly detection involves identifying observations that do not follow a pattern of normal behavior [14], or more specifically in the context of business processes and of process mining, of normal process behavior [23]. The exact notion of normal behavior and conversely of anomalous behavior are heavily dependent on the application domain [14] and on the level of analysis of the specific detection approach [23]. A possible categorization of process anomalies is between event-level and process instance-level anomalies [23], where the first refers to anomalies in one or more attributes of a specific event, while the latter to anomalies in the order and dependencies between events belonging to the same process instance. Following the more general taxonomy found in [14], [13] classifies process anomalies into three categories based on their nature: *point anomalies*, *contextual anomalies*, and *collective anomalies*. Point (or global) anomalies refer to individual observations that are anomalous compared to the rest of the data, while contextual (or local) anomalies are observations that are only anomalous in specific contexts. Collective anomalies are sets of related observations that are anomalous compared to the entire dataset, even if the individual observations may not be anomalies on their own.

In regard to existing approaches, [6] have proposed a generalization of the traditional conformance checking concepts of precision and fitness to object-centric process mining, but otherwise approaches to anomaly detection in object-centric event logs are, to the best of our knowledge, currently unexplored. More research has been done in the context of traditional process mining, which has mostly taken a case-level (i.e. process instance level) perspective. [23].

Conformance checking based approaches revolve around detecting anomalous behavior in an event log compared to a reference process model or a discovered process model. These approaches can either focus on a control-flow perspective [10,9] or consider also additional event attributes [12,7].

Distance-based approaches group traces based on distance measures, either via clustering [21] or classification [30] algorithms, while reconstruction-based approaches employ autoencoder neural networks to compute the reconstruction errors for the case encodings, which are used as anomaly scores. Approaches applying standard autoencoders [27,26] involve representing each case as an ordered vector of event activity types and events attributes, eventually padding shorter cases to the maximum length found in the event log. Similarly, approaches based

on recurrent neural networks leverage the sequential nature of the traditional case concept to train autoencoders based on gated-recurrent units (GRUs) [28] or long-short term memory (LSTM) neural networks [26,24]. [20] proposed an approach based on graph autoencoders that represented cases as loops and self-loops between activity types.

A common characteristic of the aforementioned approaches is that they require the definition of a threshold to discriminate between anomalous and normal cases [23] derived from domain knowledge or some heuristic.

Graph Neural Networks. GNNs are a class of neural networks designed to operate on graphs. They have been applied successfully to various tasks, including anomaly detection [25]. In general terms, GNNs are based on learning representations of the nodes of graphs and of their neighborhood (a k-hop of connected nodes) via a local function that is invariant to permutation of the neighboring nodes ordering [31]. [11] categorizes most GNNs into three classes based on their local function: convolutional, attentional, and generic message-passing. We are not aware of GNNs applications to object-centric process mining, but, in the context of traditional process mining, they have been employed in approaches to process discovery [29], predictive process mining [15,19,32] and anomaly detection [20]. These approaches have relied on encodings of process instances as ordered sequences of connected nodes, with the exception of [15], who employed a technique proposed in [16] to embed some temporal loops and events parallelism in the process instances, and of [20] who, as mentioned, encoded process instances as loops and self-loops between activity types (instead of events).

4 Method

Approach Overview. Our approach, as illustrated in Figure 1, relies on a GCNAE [33] trained on object-centric event logs containing both normal and anomalous events. We first reconstruct the object-centric process instances from the event logs as a single (disconnected) graph that serves as input for the GCNAE. The GCNAE is trained to reconstruct the nodes attributes of the input graph. The nodes' reconstruction errors serve as the anomaly scores. We finally apply a simple heuristic based on the IQR to automatically assign a binary label

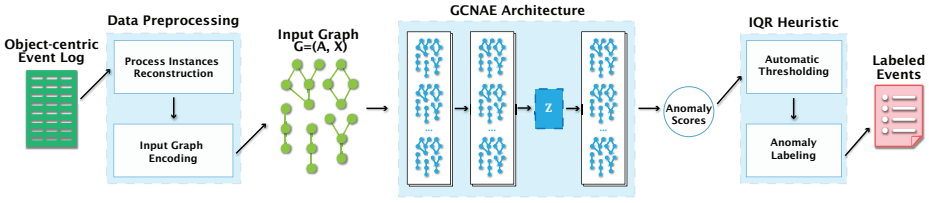

Fig. 1. An overview of our proposed approach.

to each event indicating whether it is anomalous or not, without the need to manually set an anomaly score threshold. The following sub-sections explain the different steps in detail.

4.1 Data Preprocessing

Process Instances Reconstruction. We reconstruct the object-centric process instances via the *ocpa* Python library [4] to represent the dependencies between events in the object-centric event log.

As an example, given the simple object-centric event log L in Table 1, the resulting process instances P_1 and P_2 can be visualized in Figure 2, where P_1 is composed by the set of traces $\pi_{\text{trace}}(a_1) \cup \pi_{\text{trace}}(a_3) \cup \pi_{\text{trace}}(b_3) = \langle e_1, e_4, e_7 \rangle \cup \langle e_4, e_7 \rangle \cup \langle e_4, e_8 \rangle$ which are bridged by events e_4 and e_7 and where P_2 is composed by the set of traces $\pi_{\text{trace}}(a_2) \cup \pi_{\text{trace}}(b_1) \cup \pi_{\text{trace}}(b_2) = \langle e_2, e_5, e_6 \rangle \cup \langle e_2, e_3 \rangle \cup \langle e_2, e_3, e_6 \rangle$ which are bridged by events e_2, e_3 and e_6.

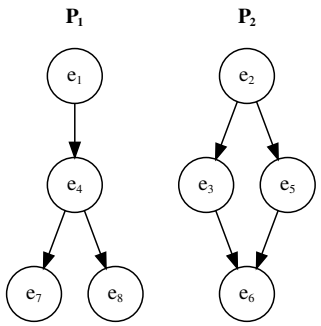

Fig. 2. The reconstructed process instances for the object-centric event log in Table 1.

Input Graph Encoding. We combine the process instances into a single graph $\mathcal{G}$, composed of a set of disconnected subgraphs (i.e., the process instances). This way, we do not need to apply padding or other transformations to the instance graphs. For the example in Figure 2, therefore the input graph would correspond to the set of the two subgraphs P_1 and P_2, $\mathcal{G}_L = \{P_1, P_2\}$.

Definition 4 (Adjacency Matrix, A). *Given the input graph $\mathcal{G}$ for the event log L, where $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a directed graph with nodes $\mathcal{V} = E_L$ and edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$, we represent it with an adjacency matrix $\mathbf{A} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$ where, for an ordered pair of nodes (u, v), $a_{u,v}$ is equal to 1 if $(u, v) \in \mathcal{E}$ and 0 otherwise.*

Table 1. An object-centric event log with two object types (A, B).

Event ID	Timestamp	Obj. Type A	Obj. Type B	Activity	Attr_1	Attr_2	...
e_1	01/04/23 09:01	a_1		act_1	0.12	0.75	...
e_2	01/04/23 09:07	a_2	b_1, b_2	act_1	0.33	0.98	...
e_3	01/04/23 09:14		b_1, b_2	act_2	0.24	0.39	...
e_4	01/04/23 09:22	a_1, a_3	b_3	act_3	0.15	0.67	...
e_5	01/04/23 09:37	a_2		act_3	0.89	0.21	...
e_6	01/04/23 09:44	a_2	b_2	act_4	0.58	0.46	...
e_7	01/04/23 10:02	a_1, a_3		act_5	0.73	0.81	...
e_8	01/04/23 10:09		b_3	act_4	0.42	0.34	...

Definition 5 (Feature Matrix, X). *Given an object-centric event log L and its input graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, we map to each node $u \in \mathcal{V}$ a feature vector, $\mathbf{x}_u \in \mathbb{R}^k$ corresponding to the event activity type $\pi_{act}(u)$ and original attributes $\pi_{attr}(u)$, where categorical features are one-hot encoded, and thus k is equal to the sum of unique activity types in A_L, unique categorical values in AV_L and numerical attributes in AV_L. Then, we stack all the feature vectors into a feature matrix $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}| \times k}$, where $\mathbf{X} = \left[\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{|\mathcal{V}|}\right]^\top$.*

Accordingly, the input graph $\mathcal{G}$ for the GCNAE is encoded as $\mathcal{G} = (\mathbf{A}, \mathbf{X})$. For the running example of the object-centric event log L in Table 1, the adjacency matrix $\mathbf{A}_L$ and the feature matrix $\mathbf{X}_L$ would look as:

$$\mathbf{A}_L = \begin{bmatrix} 0&0&0&1&0&0&0&0 \\ 0&0&1&0&1&0&0&0 \\ 0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&1&1 \\ 0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0 \end{bmatrix} \quad \mathbf{X}_L = \begin{bmatrix} 1&0&0&0&0&0.12&0.75&\cdots \\ 1&0&0&0&0&0.33&0.98&\cdots \\ 0&1&0&0&0&0.24&0.39&\cdots \\ 0&0&1&0&0&0.15&0.67&\cdots \\ 0&0&1&0&0&0.89&0.21&\cdots \\ 0&0&0&1&0&0.58&0.46&\cdots \\ 0&0&0&0&1&0.73&0.81&\cdots \\ 0&0&0&1&0&0.42&0.34&\cdots \end{bmatrix}$$

4.2 GCNAE Architecture

The GCNAE uses a graph convolutional network (GCN) [22] component both for the the encoder and for the decoder. Graph convolution aggregates the information from neighboring nodes and updates node representations based on their local graph structure.

The encoder component of the GCNAE learns a latent representation of the input graph while the decoder learns to reconstruct the nodes features from this latent representation. The GCNAE is trained by minimizing the reconstruction error between the original nodes features and the reconstructed ones.

Adapted from [33], our encoder consists of a two-layer GCN:

$$\mathbf{Z} = GCN(\mathbf{X}, \mathbf{A}) = \text{ReLU}(\tilde{\mathbf{A}}\text{ReLU}(\tilde{\mathbf{A}}\mathbf{X}\mathbf{W}_{(\mathbf{0})})\mathbf{W}_{(\mathbf{1})}) \tag{1}$$

where the output $\mathbf{Z}$ is a matrix of node embeddings, $\mathbf{X}$ is the feature matrix, $\mathbf{A}$ is the adjacency matrix of the graph. ReLU is the rectified linear unit activation function with ReLU(x) = max(0,x). $\tilde{\mathbf{A}}$ is the symmetrically normalized adjacency matrix with $\tilde{\mathbf{A}} = \mathbf{D}^{-\frac{1}{2}}(\mathbf{A} + \mathbf{I})\mathbf{D}^{-\frac{1}{2}}$ (where $\mathbf{I}$ is the identity matrix and $\mathbf{D}$ is the diagonal degree matrix), used to add self-loops and normalize nodes features aggregation. $\mathbf{W}_{(\mathbf{l})}$ denotes a learnable weight matrix for each l-th layer.

The decoder's reconstructed node attributes $\hat{\mathbf{X}}$ are computed with a third GCN layer that maps the node embeddings back to the original input space:

$$\hat{\mathbf{X}} = \text{ReLU}(\tilde{\mathbf{A}}\mathbf{Z}\mathbf{W}_{(\mathbf{2})}) \tag{2}$$

As the loss function to optimize the GCNAE we take the average row-wise mean squared error (MSE) between $\mathbf{X}$ and $\hat{\mathbf{X}}$, while, to account for different shapes in the features encoding, we compute the events anomaly scores by taking the average MSE for each event original feature which are then averaged again.

4.3 IQR Heuristic: Automatic Thresholding and Anomaly Labeling

The IQR is is often used to detect outliers and is defined as the difference between the first quartile (Q_1) and the third quartile (Q_3) of a dataset, $IQR = Q_3 - Q_1$.

To determine a threshold to label anomalies, we compute the Q_3 and the IQR for anomaly scores generated by the GCNAE and set the threshold τ as:

$$\tau = Q_3 + k \cdot \text{IQR} \tag{3}$$

Once the threshold is set, we can automatically label the events in the event log as normal or anomalous based on whether their anomaly score is respectively below or above the threshold. In this study, we set $k = 1.5$ based on convention, but it could be adjusted to tailor the sensitivity of the threshold to specific uses.

5 Experiments

We evaluate our proposed approach on both synthetic and real object-centric event logs, since real event logs can help prove the feasibility of the approach but at the same can also contain unlabeled real-life anomalies which can impact the reliability of the results [26,28]. Like in previous work [26,28,27,24], we introduce artificial anomalies into the event logs to simulate various types of irregularities that can occur in real-world processes. Summary statistics for the datasets and the reconstructed process instances can be found in Table 2.

5.1 Datasets

We use the *BPIC 2017* dataset [17] as a representative of a real event log. This event log is commonly employed in research on process discovery and on predictive process mining. It contains over 500.000 events for the loan application process of a Dutch financial institution between 2016 and 2017. The event log has two types of objects (the application and the offer) and thirteen attributes.

The second event log is the synthetic *DS2* dataset found in [5]. It simulates an order management process with an especially high amount of connected objects and variability. The event log has three types of objects (items, orders, and packages) and four attributes.

Table 2. Description of the datasets used in the experiments.

Dataset	Original Events	Final Events	Injected Anomalies	Process Instances
BPIC 2017	393.931	407.499	40.704	31.509
DS2	22.367	23.137	2.310	83

5.2 Anomalies Injection

Before reconstructing the process instances, we directly manipulate the object-centric event logs by introducing three types of anomalies in equal parts, amounting in total to circa 10% of the final number of events in each event log.

Attributes Swap: these anomalies are created by altering the attributes of an event, introducing inconsistencies when compared to events in the same process instance. Given the candidate event i, we select the event j whose attributes deviate the most from the attributes of event i by maximizing the euclidean distance $||x_i - x_j||$ and replace the original attributes of i with those of j.

Timestamp Shift: these anomalies are introduced by sampling an existing event in the event log and shifting its timestamp within the time-frame ($\pm 5\%$) of all other events that share a common object with the candidate event. This leads to discrepancies in the temporal order of the events in the process instance.

Random Activities: similarly to [28], we inject events into the process instances with an activity type that does not come from the original process, while the attributes are sampled from the target process instance.

5.3 Baselines

We compare the performance of our graph-based approach to existing approaches for traditional event logs. To be able to do so, we 'flatten' the object-centric process instances to temporally ordered sequences composed by the set of events belonging to each process instance. While this flattening strategy still leads to divergence issues, it does not lead to the deletion or duplication of events.

As baselines, we implement a standard autoencoder (AE) similar to the one in [27,26] and a LSTM autoencoder (LSTMAE) similar to the recurrent neural network approaches found in [28,26,24]. For both, we used the hyperparameters found in previous implementations [28,26] in the literature.

5.4 Results

Table 3 shows the experiments results. We ran our experiments five times with different random seeds. To compute the F1 Scores, we applied the IQR heuristic to all models.

Table 3. Performance comparison of the different models. Results reported as mean ± standard deviation. Best model in bold.

Dataset	Model	F1 Score	AUC ROC	AUC PR	Recall @ 10
BPIC 2017	AE	52.5 ± 0.1	**85.6 ± 0.1**	43.5 ± 0.1	52.1 ± 0.1
	LSTMAE	44.7 ± 0.2	82.9 ± 0.1	34.4 ± 0.2	43.9 ± 3.3
	GCNAE	**61.0 ± 0.2**	82.4 ± 0.0	**60.3 ± 0.2**	**64.7 ± 0.1**
DS2	AE	OOM	OOM	OOM	OOM
	LSTMAE	31.2 ± 0.6	68.4 ± 0.4	23.5 ± 0.9	30.6 ± 0.7
	GCNAE	**59.2 ± 0.4**	**82.5 ± 0.4**	**65.0 ± 0.4**	**66.6 ± 0.1**

The GCNAE generally outperforms the baselines, which could be attributed to the inherent ability of GNNs to reason over graphs and account for dependencies between events. We also note how the AE goes out-of-memory on the *DS2* dataset since the high number of events in the process instances of this dataset result in long vector encodings.

Furthermore, in Table 4 we present the hit rate (i.e., proportion of events assigned to the correct class out of the class totals) for each class and model. Each anomaly type has a frequency of circa 3.33% and normal events of circa 90%, which can be seen as the hit rates for a random classifier.

Table 4. Hit Rate for the different classes, where classes are assigned via the IQR heuristic for all models. Results reported as mean ± standard deviation. Best model in bold.

Dataset	Model	Normal	Attr. Swap	Timestamp	Random Act.
BPIC 2017	AE	93.41 ± 0.07	93.73 ± 0.35	**28.22 ± 0.45**	48.18 ± 0.53
	LSTMAE	92.56 ± 0.02	87.37 ± 0.39	21.35 ± 0.35	35.50 ± 0.70
	GCNAE	**97.47 ± 0.06**	**95.75 ± 0.40**	2.57 ± 0.05	**63.13 ± 1.27**
DS2	AE	OOM	OOM	OOM	OOM
	LSTMAE	90.24 ± 0.16	83.79 ± 2.81	**9.82 ± 0.47**	10.47 ± 1.40
	GCNAE	**93.22 ± 0.13**	**96.73 ± 0.21**	6.55 ± 0.63	**100.00 ± 0.00**

The GCNAE struggles detecting the *Timestamp Shift* anomalies. A possible explanation is that GCNs learn nodes representations by aggregating local neighborhood information. Shifting an event within a process instance time-frame might create very subtle changes in the event neighborhood, which therefore would limit the performance of the GCNAE for this specific anomaly.

6 Conclusion

Detecting anomalies is important for identifying inefficiencies, errors, or fraud in business processes. Object-centric event logs provide benefits over traditional event log representations. We have presented a novel approach for anomaly detection in business processes that can leverage the joint capabilities of GNNs and object-centric process mining. Our approach has displayed promising performance, while also not requiring a process model, manual labeling the data, or the availability of a clean training set.

Future work could explore the use of GNNs architectures that are better suited to learn the temporal and structural dependencies of business process instances and address the limitations displayed by the GCNAE. Future work could also investigate the combined performance of our approach and of object-centric conformance checking approaches since they could complement each other, the first tackling attribute anomalies and the second control-flow anomalies.

References

1. van der Aalst, W.M.P.: Object-Centric Process Mining: Dealing with Divergence and Convergence in Event Data. In: Ölveczky, P.C., Salaün, G. (eds.) Software Engineering and Formal Methods. pp. 3–25. Lecture Notes in Computer Science, Springer International Publishing, Cham (2019)
2. van der Aalst, W.M.P.: Process mining: A 360 degree overview. In: van der Aalst, W.M.P., Carmona, J. (eds.) Process mining handbook, pp. 3–34. Springer International Publishing, Cham (2022)
3. Adams, J.N.: Addressing Convergence, Divergence, and Deficiency Issues
4. Adams, J.N., Park, G., van der Aalst, W.M.: ocpa: A Python library for object-centric process analysis. Software Impacts p. 100438 (2022)
5. Adams, J.N., Schuster, D., Schmitz, S., Schuh, G., van der Aalst, W.M.: Defining Cases and Variants for Object-Centric Event Data. In: 2022 4th International Conference on Process Mining (ICPM). pp. 128–135 (2022)
6. Adams, J.N., Van Der Aalst, W.M.: Precision and Fitness in Object-Centric Process Mining. In: 2021 3rd International Conference on Process Mining (ICPM). pp. 128–135. IEEE, Eindhoven, Netherlands (2021)
7. Bergami, G., Maggi, F.M., Marrella, A., Montali, M.: Aligning Data-Aware Declarative Process Models and Event Logs. In: Business Process Management: 19th International Conference, BPM 2021, Rome, Italy, September 06–10, 2021, Proceedings. pp. 235–251. Springer-Verlag, Berlin, Heidelberg (2021)
8. Berti, A., Herforth, J., Qafari, M., Aalst, W.M.v.d.: Graph-Based Feature Extraction on Object-Centric Event Logs. preprint, In Review (2022)
9. Bezerra, F., Wainer, J.: Algorithms for anomaly detection of traces in logs of process aware information systems. Information Systems **38**(1), 33–44 (2013)
10. Bezerra, F., Wainer, J., van der Aalst, W.M.P.: Anomaly Detection Using Process Mining. In: Halpin, T., Krogstie, J., Nurcan, S., Proper, E., Schmidt, R., Soffer, P., Ukor, R. (eds.) Enterprise, Business-Process and Information Systems Modeling. pp. 149–161. Lecture Notes in Business Information Processing, Springer, Berlin, Heidelberg (2009)
11. Bronstein, M.M., Bruna, J., Cohen, T., Veličković, P.: Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges (2021), arXiv:2104.13478 [cs, stat]
12. Böhmer, K., Rinderle-Ma, S.: Multi-perspective Anomaly Detection in Business Process Execution Events pp. 80–98 (2016), mAG ID: 2538859255
13. Böhmer, K., Rinderle-Ma, S.: Anomaly Detection in Business Process Runtime Behavior – Challenges and Limitations (2017), arXiv:1705.06659 [cs]
14. Chandola, V., Banerjee, A., Kumar, V.: Anomaly detection: A survey. ACM Computing Surveys **41**(3), 1–58 (2009)
15. Chiorrini, A., Diamantini, C., Mircoli, A., Potena, D.: Exploiting Instance Graphs and Graph Neural Networks for Next Activity Prediction. In: Munoz-Gama, J., Lu, X. (eds.) Process Mining Workshops. pp. 115–126. Lecture Notes in Business Information Processing, Springer International Publishing, Cham (2022)
16. Diamantini, C., Genga, L., Potena, D., van der Aalst, W.: Building instance graphs for highly variable processes. Expert Systems with Applications **59**, 101–118 (2016)
17. van Dongen, B.F.: BPI Challenge 2017 (2017), publisher: 4TU.ResearchData
18. Ghahfarokhi, A.F., Park, G., Berti, A., van der Aalst, W.M.P.: OCEL: A Standard for Object-Centric Event Logs. In: Bellatreche, L., Dumas, M., Karras, P., Matulevičius, R., Awad, A., Weidlich, M., Ivanović, M., Hartig, O. (eds.) New Trends in Database and Information Systems. pp. 169–175. Communications in Computer and Information Science, Springer International Publishing, Cham (2021)

19. Harl, M., Weinzierl, S., Stierle, M., Matzner, M.: Explainable predictive business process monitoring using gated graph neural networks. Journal of Decision Systems **29**(sup1), 312–327 (2020)
20. Huo, S., Völzer, H., Reddy, P., Agarwal, P., Isahagian, V., Muthusamy, V.: Graph Autoencoders for Business Process Anomaly Detection. In: Polyvyanyy, A., Wynn, M.T., Van Looy, A., Reichert, M. (eds.) Business Process Management. pp. 417–433. Lecture Notes in Computer Science, Springer International Publishing, Cham (2021)
21. Junior, S.B., Ceravolo, P., Damiani, E., Omori, N.J., Tavares, G.M.: Anomaly Detection on Event Logs with a Scarcity of Labels. In: 2020 2nd International Conference on Process Mining (ICPM). pp. 161–168 (2020)
22. Kipf, T.N., Welling, M.: Semi-Supervised Classification with Graph Convolutional Networks (2017), arXiv:1609.02907 [cs, stat]
23. Ko, J., Comuzzi, M.: A Systematic Review of Anomaly Detection for Business Process Event Logs. Business & Information Systems Engineering (2023)
24. Lahann, J., Pfeiffer, P., Fettke, P.: LSTM-Based Anomaly Detection of Process Instances: Benchmark and Tweaks. In: Process Mining Workshops, vol. 468, pp. 229–241. Springer Nature Switzerland, Cham (2023), series Title: Lecture Notes in Business Information Processing
25. Liu, K., Dou, Y., Zhao, Y., Ding, X., Hu, X., Zhang, R., Ding, K., Chen, C., Peng, H., Shu, K., Sun, L., Li, J., Chen, G.H., Jia, Z., Yu, P.S.: BOND: Benchmarking Unsupervised Outlier Node Detection on Static Attributed Graphs (2022), arXiv:2206.10071 [cs]
26. Nguyen, H.T.C., Lee, S., Kim, J., Ko, J., Comuzzi, M.: Autoencoders for improving quality of process event logs. Expert Systems with Applications **131**, 132–147 (2019)
27. Nolle, T., Luettgen, S., Seeliger, A., Mühlhäuser, M.: Analyzing business process anomalies using autoencoders. Machine Learning **107**(11), 1875–1893 (2018)
28. Nolle, T., Luettgen, S., Seeliger, A., Mühlhäuser, M.: BINet: Multi-perspective business process anomaly classification. Information Systems **103**, 101458 (2022)
29. Sommers, D., Menkovski, V., Fahland, D.: Process Discovery Using Graph Neural Networks. In: 2021 3rd International Conference on Process Mining (ICPM). pp. 40–47 (2021)
30. Tavares, G.M., Barbon, S.: Analysis of Language Inspired Trace Representation for Anomaly Detection. In: ADBIS, TPDL and EDA 2020 Common Workshops and Doctoral Consortium. pp. 296–308. Communications in Computer and Information Science, Springer International Publishing, Cham (2020)
31. Veličković, P.: Everything is Connected: Graph Neural Networks (2023), arXiv:2301.08210 [cs, stat]
32. Weinzierl, S.: Exploring Gated Graph Sequence Neural Networks for Predicting Next Process Activities. In: Business Process Management Workshops. pp. 30–42. Lecture Notes in Business Information Processing, Springer International Publishing, Cham (2022)
33. Yuan, X., Zhou, N., Yu, S., Huang, H., Chen, Z., Xia, F.: Higher-order Structure Based Anomaly Detection on Attributed Networks. In: 2021 IEEE International Conference on Big Data (Big Data). pp. 2691–2700 (2021)