Semi-supervised Detection of Contaminated Business Process Instances using Graph Autoencoders and Dynamic Edge Convolutions for BPM Anomaly Detection

Semi-supervised Detection of Contaminated Business Process Instances using Graph Autoencoders and Dynamic Edge Convolutions for BPM Anomaly Detection

Teoman Berkay Ayaz*, Ege Gülce*, Alper Özcan†, Akhan Akbulut*‡
R&D Center of Next4biz Sahrayicedit Mah, Pakpen Plaza,
No:40/4, 34734 Istanbul, Turkey
teoman.ayaz@next4biz.com, ege.gulce@next4biz.com, akhan.akbulut@next4biz.com
†*Department of Computer Engineering Akdeniz University* Antalya, Turkey
alperozcan@akdeniz.edu.tr
‡*Department of Computer Engineering Istanbul Kültür University* 34536 Istanbul, Turkey
a.akbulut@iku.edu.tr

***Abstract*—Anomalies in business processes pose a significant threat to the operational performance of modern enterprises. Consequently, detecting these anomalies in business process management (BPM) systems is a critical task for organizations. However, the inherent complexity of business processes and the limited availability of labeled data compound this challenge. In this study, we introduce innovative models that leverage graph representations of business process data, incorporating an unsupervised neural network architecture known as Graph Autoencoders combined with Dynamic Edge Conditioned Convolutions to enhance learning on edge attributes. Additionally, we propose a novel metric for evaluating autoencoder performance. We developed two models: a fully unsupervised model trained on the entire dataset and a semi-supervised model that refines the initial model's outputs. Our empirical results demonstrate that the unsupervised model achieves a maximum F1-score of 0.82 and a success rate of 74.11% at the 80th percentile threshold for detecting anomalous sub-sequences in business processes. The semi-supervised model, which builds on the unsupervised model's findings, achieves a maximum F1-score of 0.89 and a success rate of 80.50%, indicating a substantial performance improvement. Both models offer promising capabilities for automating anomaly detection in BPM systems.**

***Index Terms*—business process, graph neural networks, edge attribute, anomaly detection, unsupervised, semi-supervised**

I. Introduction

Business Process Management (BPM) plays an important role in the modern world as it involves the structured design, execution, monitoring, and optimization of business processes. Successful BPM empowers organizations to improve operational productivity and achieve strategic goals. By evaluation of process data, BPM systems provide insights into workflow performance and pinpoint areas for enhancement. Business process anomalies refer to deviations from the expected process behavior, signaling potential inefficiencies, errors, or fraudulent behaviors. These anomalies can disrupt regular operations, resulting in higher costs, lower productivity, and decreased efficiency. Therefore, identifying and addressing anomalies is crucial for maintaining the integrity and efficiency of business processes. Anomaly detection in this context can assist in the correction of said deviations, increasing the efficiency and preventing potential fraudulent activity in turn increasing the profitability of the BPM tenant.

Anomalies can serve as indicators of challenges in a business process, but they can also facilitate the identification of fraudulent activities [1]. Detecting anomalies in business processes [2] is challenging due to several reasons. The scarcity of labeled data is the first obstacle to tackle. Obtaining annotated datasets to train anomaly detection models is often a difficult and costly task. Additionally, business workflows are inherently complex, involving various flows and multiple interdependent activities. Moreover, the fact that each company's business processes can vary significantly, with unique workflows and operational practices. This variability makes it difficult to accurately identify anomalous behavior, as anomalies can appear in diverse forms and contexts. Traditional approaches to anomaly detection, including statistical techniques and rule-based systems, frequently struggle to handle the complexities of modern business processes. These methods typically need extensive domain expertise and may not adapt well to changing process dynamics. Therefore, there is a growing demand for advanced techniques that can utilize machine learning and deep learning to enhance detection accuracy and efficiency.

The contributions of this study are to accomplish a way to leverage existing techniques to develop an approach for BPM anomaly detection that is scaleable across workflow sequences of different lengths, applicable across workflows of different companies and most importantly fully automatizable so no human intervention is needed, eliminating the issues arising due to the scarcity of labeled data and the expense of the

979-8-3503-7943-3/24/$31.00 ©2024 IEEE

needed human supervision. The primary focus of this study is on the application of graph neural networks (GNNs) and auto-encoders to effectively capture the sequential dependencies and complex relationships present in business process event logs. By leveraging these advanced models, we aim to enhance the accuracy and real-time applicability of business process anomaly detection systems. The main contributions made within this study are as outlined below;

- We leveraged an unsupervised graph neural network, which can be leveraged and generalized across different business process workflows without needing human intervention.
- We introduced a generalizable and automatable approach to train semi-supervised autoencoders for further performance improvements.
- We introduced a new method to evaluate autoencoder performance above a given threshold making for better evaluation.

The remainder of the paper is as follows; The related works section offers a brief overview of the previous work done in the area of BPM anomaly detection. In section 3, the methodology leveraged in this study including the graph creation and model construction are explained in detail. In section 4 experimental results, we present the results of our experiments. Lastly, in section 5 conclusion and future work, we give a brief conclusion of our work and discuss potential directions for future work.

II. Related Works

Anomaly detection in business processes is essential for identifying deviations from standard process behavior. Recent advancements in machine learning have made significant improvements in detecting these anomalies. This section will review the key contributions in this area. As a predecessor to this study, Sezer et al. [3] used an approach where anomaly detection on business process sub-sequences took place in an unsupervised manner using isolation forest with differing contamination levels. The labels acquired were verified by domain experts and business intelligence specialists to be anomalous instances. Auto-encoders have been widely used for anomaly detection in various fields, including business processes. The work by Nolle et al. [4] introduced a method using auto-encoders for detecting anomalies in business process logs. It works well with noisy datasets, representing real business logs, and does not require prior knowledge of the process. They demonstrated the effectiveness of their approach by testing it on multiple datasets, achieving substantial performance improvements compared to state-of-the-art techniques.

The GRASPED model by Guan et al. [5], is a further contribution in this area. The model utilizes a GRU-AE network for multi-perspective business process anomaly detection. It uses auto-encoders with gated recurrent units (GRU) and introduces the teacher forcing method along with an attention mechanism to enhance detection performance. One of the significant advantages of GRASPED is that it can train on logs containing anomalies without requiring a clean dataset, which is common in real business process data. By adopting GRU within their auto-encoder framework, the researchers captured forward and backward dependencies in sequential data. This enhanced their model's ability to handle noisy datasets, significantly enhancing the effectiveness and reliability of their proposed solution. Another work by Krajsic et al. [6], proposed a semi-supervised classification model that combines deep generative models, time series analysis, and sequence processing. Their model filters activity-related and time-related anomalies from event data by leveraging deep learning techniques such as long short-term memory (LSTM) networks and attention mechanisms. Graph neural networks are also used for anomaly detection in business processes. Huo et al. [7] developed a graph auto-encoder for this purpose, leveraging the structural properties of process graphs to detect anomalies. This method showed promising results in identifying both control flow and data anomalies in business processes.

The researchers who previously worked in this domain made considerable advancements, proven by the empirical evidence. Nolle et al. [4] utilized Denoising Auto Encoders (DAE) for unsupervised anomaly detection in noisy business process event logs. They achieved an F1 score of 0.87 on average testing on 700 various datasets. Guan et al. [5] employed graph neural networks for link restoration between split process events, achieving an F1-score of 0.86 on synthetic logs, and 0.55 on real-life logs. Lastly, Krajsiv et al. [6] developed a semi-supervised classification model based on deep learning, specifically using bidirectional LSTM recurrent neural networks as the encoder and decoder, reaching an F1 score up to 93% on multidimensional benchmark datasets. In summary, while there have been significant improvements in the field of business process anomaly detection, each approach has its strengths and limitations. Auto-encoder-based methods are good at handling noisy datasets and providing high accuracy, but they struggle with long-term dependencies. Graph-based approaches effectively utilize the structural information of processes but can be computationally expensive. Despite these advancements, there are still gaps in addressing the scalability and real-time application of these methods. This paper proposes a novel approach that combines the strengths of existing methods. By leveraging an approach based on the graph representation of business process data combined with auto-encoders, we aim to provide a solution for real-time detection of anomalous business process instances.

III. Methodology

In this study, we trained two Graph Neural Network (GNN) based Autoencoder (AE) models, one fully unsupervised and the other semi-supervised, to detect business process instances with anomalous subsequences using the Edge Conditioned Convolutional (ECC) layers (a.k.a. Dynamic Edge Conditioned Convolutions) proposed by Simonovsky et al. [8], to serve as a component of AnomalyNet, a multi-perspective anomaly detection framework for business processes as a part of the Next4biz BPM platform.

A. Dataset

The dataset used within the study consists of 44,413 rows of business process logs of 6178 distinct business process instances, belonging to an anonymous tenant of the Next4biz BPM platform. Each log (business process event instance) contains a total of 8 fields: the issue ID of the respective business process record, the name of the transition/action that the log represents, the state of the process instance which the activity took the process to, starting time and ending time of the action represented, duration of the aforementioned action, the ID of the user who commenced the action alongside a description of the mentioned user.

Throughout the dataset, a total of 995 of the process records are contaminated with anomalous business process sub-sequences. The detection of these sub-sequences was done in a study by Sezer et al. [3], related to this dataset. Within the aforementioned study, the definition of an anomalous sub-sequence is; that even if a previous-current-next sequence of activities complies with the rules of the workflow, it is anomalous if it deviates from the usual flow of business process records making the entire record contaminated. This definition is meant to cover instances where the anomaly can co-exist with the workflow rules defined by the tenant, such as fraudulent activity in line with the business process rules, detectable by the defined approach.

B. Advantages of Graph Data Structure

The Business Process Workflows within the Next4biz BPM platform are represented by two main components as shown in Figure 1; states which represent the current status of a business process record and a transition which is the connection and the flow between different states within the process workflow. This makes it so that the business processes found within the platform are naturally fit to be represented as graphs. Hence, we can represent different states as nodes and the transitions between these states as edges. The graph representation allows us to not only train models based on the features of singular logs but also the entire process of learning from both nodes and their connections to different nodes in addition to the attributes that can be contained on said connections.

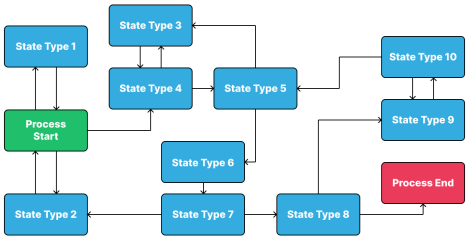

Fig. 1. Example of a Business Process Diagram.

C. Graph Creation Scheme

The graph creation scheme used within the study is an adaptation of the graph construction algorithm proposed by Huo et al. [7] in 2021, done in a more suitable way for the dataset this study is done on. The graph creation is done per business process record instance (case) as each case will be represented by a separate graph. The process from the tabular dataset can be summarized as follows:

1) Initialization of Edge Embedding Space: To initialize the edge embedding space, the maximum trace length is calculated. The maximum trace length T_{max} is defined as the maximum number of events in a single case. Given that:

- D is the dataset containing n rows.
- C is the set of unique cases in D.
- $trace(c)$ is the sequence of activities for case c.

The maximum trace length calculation can be formulated as:

$$T_{max} = \max_{c \in C}(|trace(c)|)$$

where $|trace(c)|$ represents the number of states the business process goes through with respect to case c.

2) State-State Pair (Edge) Formation: State-state pairs are formed to create the edges of the graphs. Each state at time t is paired with the state at time $t+1$ to pair consecutive states of the business process.

$$trace(c) = \{a_0, a_1, a_2, \ldots, a_n\}$$

is the sequence of states for case c, ordered on the basis of time. The set of edges E_c with respect to case c can be defined as:

$$E_c = \{(a_i, a_{i+1}) | a_i, a_{i+1} \in trace(c), i = 1, 2, \ldots, T_{max} - 1\}$$

3) Creation of Edge Positional Embeddings (Edge Attributes): To create the edge embeddings, a $T_{max} - 1$ dimensional feature space is first initialized. The edge embeddings are created by the insertion of the respective features of the transition represented by each edge, into their indexes in the feature space. Given that:

- T_{max} is the maximum trace length.
- E_c is the set of edges for case c.
- $d(a_i, a_{i+1})$ is the duration of the edge (a_i, a_{i+1}).

The positional embedding P_c for case c is an $(T_{\max} - 1)$-dimensional vector, where the duration of each edge (a_i, a_{i+1}) is inserted at the i-th index:

$$P_c[i] = d(a_i, a_{i+1}) \quad \text{for } i = 1, 2, \ldots, k-1$$

4) Aggregation of Edge Embeddings: Edges consisting of the same state-state pair are aggregated via the summation of each respective edges positional embeddings. Given:

- A is the set of unique state-state pairs in the dataset.
- $P_{a_i, a_{i+1}}$ is the positional embedding vector for the edge (a_i, a_{i+1}).

The aggregated positional embedding P_{agg} for each unique state-state pair (a_i, a_{i+1}) is defined as:

$$P_{agg}(a_i, a_{i+1}) = \sum_{(a_i, a_{i+1}) \in E} P_{a_i, a_{i+1}}$$

D. Graph Autoencoder with Edge Conditioned Convolutions

The main task of the study is unsupervised anomaly detection due to the fact that business process data, more often than not, does not contain any labels that could be used to train a supervised model. An often leveraged approach for this particular type of task is to use an Autoencoder. An autoencoder is a breed of neural networks designed to first compress a given input into a meaningful low-dimensional representation followed by the reconstruction of said input in a way that it is as close to the original as possible [9]. In typical fashion, the autoencoder model deployed within this study as shown in Figure 2 consists of three main components; an encoder which reduces the input into a meaningful latent feature representation, the latent feature representation which contains the information the input will be reconstructed through, and the decoder which will reconstruct the input features from the latent feature representation as similar to the original features as possible [10]. As shown both the encoder and the decoder consists of ECC layers, where feature $X_l(1)$ on vertex 1 in the l-th network layer is calculated as a sum of features ($X_{l-1}(n)$) with weights on the vertices it follows. The weights leveraged are generated dynamically by filter generating network [8].

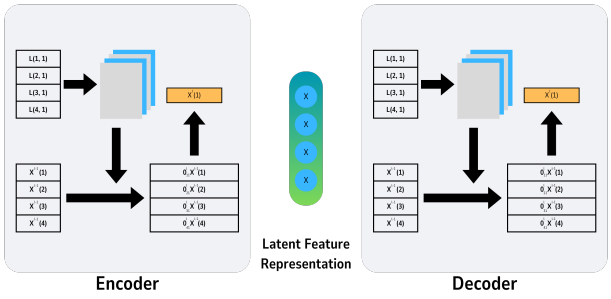

Fig. 2. ECC-GAE Architecture

The training process of an autoencoder is typically done using the reconstruction error which is used to inquire about how well the model can compress the features into a meaningful representation to reconstruct the original from the latent feature representations [10]. In this study, the method leveraged to calculate the deviation of the reconstructed input was Mean Squared Error (MSE) due to it being the most common choice to evaluate how much the reconstructed input deviates from the original.

In addition to the model training and evaluation, the reconstruction error can also be used to tell us which instances in our dataset are anomalous. Anomalous instances by definition are instances that are out of the ordinary. When training, the autoencoder will minimize its reconstruction error. With the assumption that an autoencoder is trained on a dataset with none to negligible amounts of outliers, the autoencoder should be able to reconstruct normal inputs with minimal error. With regards to this case, any input giving an above-normal reconstruction error when reconstructed via the autoencoder can be regarded as an anomaly [10]. However with regards to the scalability and applicability of autoencoders across different datasets, not every dataset is going to contain negligible or no anomalies. Due to this reason, in an application where human intervention is simply not possible, an alternative approach is needed to overcome the potential issue of an autoencoder learning over anomalies resulting in lower anomaly detection performance.

With the consideration of business process data naturally aligning itself with graph represented data, we can leverage GNNs for better learning on the aforementioned business process graphs. Specifically, we leverage an autoencoder breed based on Graph Convolutional Networks (GCN) called Graph Autoencoders (GAE). GCNs in comparison to more conventional types of neural networks which work on Euclidian data with regular structures, are a type of neural network designed to capture and learn from the structure of graph data. GCNs leverage the information present on graphs by aggregating and transforming feature information from nodes and node neighbourhoods [11]. As our data can naturally be represented as graph data, leveraging the use of GCNs can greatly improve the quality of anomaly detection.

Due to the nature of our dataset where edges represent transitions, which contain important information on the process, the study requires the usage of a model which can learn via edge attributes. For this, we leveraged the Edge Conditioned Convolution (ECC) operator, proposed by Simonovsky et al. [8]. ECC operator is a generalized operator applicable to many domains where data can be represented as graph data. It works in the spatial domain by conditioning filter weights on edge attributes and dynamically generating them for each input sample. This allows the latent feature representation of our graphs to contain information from edge attributes, resulting in better representation for our cases.

By combining the aforementioned approaches, we built what will be referred to as "Edge-Conditioned Convolutional Graph Autoencoder" (ECC-GAE). An autoencoder model made to do unsupervised anomaly detection on graph-represented business process data. We've trained two different models for node feature reconstruction with the same architecture. The initial model was trained on the entire dataset followed by the acquisition of the reconstruction error of each case. The reconstruction errors were used to pick out which cases were more difficult to reconstruct for the model, indicating that they are anomalous instances. The latter model was trained in a semi-supervised manner by using the reconstruction errors from the first model.

We first calculated the z-score for each individual case by using the mean reconstruction error across each node of the given case. Due to the high variance within the reconstruction errors, the standard deviation turned out to be greater than the mean. Due to this, we left out the reconstruction errors in the top 20% (above 80th percentile) of the anomalies when calculating the mean and standard deviation for z-score calculation. Once the z-scores were calculated, we chose $Median(Z_i) \times 2$ as an upper bound for safe training samples.

"Safe" in this context means likely to be normal due to the relatively low reconstruction error, so when an autoencoder is trained using only these samples, it will yield a significantly greater reconstruction error when reconstructing anomalies in contrast to its fully unsupervised counterpart.

Lastly, we evaluated the results using a label set we acquired in the aforementioned study [3]. When evaluating we selected thresholds by choosing percentile points of our reconstruction errors to keep the knowledge of labels outside of the prediction as well. One issue we encountered was the evaluation metric. F1-scores would be lower as we picked a higher threshold as the lower bound for anomaly reconstruction error, due to the number of true positives decreasing. To evaluate the performance, alongside using the f1-score, we propose the usage of a ratio of anomalous instances to all instances. Given that:

- p is the percentile point
- P_p is all the predictions above the percentile point
- P_{true} is the entirety of true positives (anomalies) above a given threshold.

The success rate of the models above a given threshold can be defined as $SuccRate = \frac{P_{true}}{P_p} \times 100$.

IV. Experimental Results

The initial fully unsupervised model was trained using every single instance of the dataset. The training was cut at epoch 50 due to the loss plateauing after that point. The average reconstruction error was calculated to be 5.662×10^{-4} across all instances in the dataset. We then picked four thresholds based on points within the reconstruction errors of 80th percentile, 90th percentile, 95th percentile and 99th percentile as our anomaly lower bounds. The resulting thresholds as shown in Figure 3 are; 8.9×10^{-5}, 3.14×10^{-4}, 6.3×10^{-4}, 2.6×10^{-2} in the same respective order. The thresholds show us that as we go lower in our ordered set of reconstruction errors, the variance lessens and the model seems to start to reconstruct the instances with about the same accuracy.

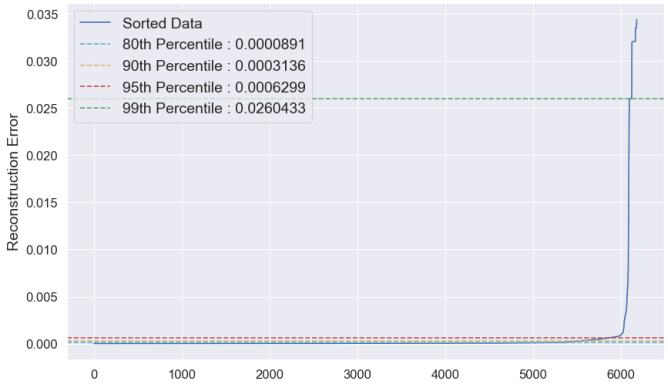

Fig. 3. Unsupervised Model Recon. Errors and Anomaly Thresholds

With our thresholds calculated, we used the aforementioned labels, with an anomaly ratio of $16 : 84$. As shown in Table I, at the top threshold the 99th percentile we get an f1-score and a success rate of 0 indicating that above this threshold, there are no true positives detected. As we progress further down amongst our thresholds, we can see that at the 95th percentile, we get an f1-score of 0.24 and a success rate of 50.81 indicating that we start to see the first of our anomalies, at the 90th percentile we get an f1-score of 0.53 and a success rate of 69.42 with most of the instances above the threshold being true positives. Lastly at the 80th percentile, as expected from the anomalous instance ratio we get the highest f1-score of 0.82 and the highest success rate of 74.11. At this threshold, a higher count of instances are labeled as anomalies than there are actual anomalies in the datasets, so after this threshold, the performance will only be lower.

TABLE I
Unsupervised Model Results

Percentile	Threshold	F1-Score	Succ. Rate	Anomaly:All
99th	2.6×10^{-2}	0.00	0.00	0:62
95th	6.3×10^{-4}	0.24	50.81	157:309
90th	3.14×10^{-4}	0.53	69.42	429:618
80th	8.9×10^{-5}	0.82	74.11	916:1236

With regards to picking the instances to train the semi-supervised model, the upper bound of $Median(Z_i) \times 2$ yielded 3608 of the total 6178 cases or 58.40% of the dataset as likely normal instances, which will the training data for the semi-supervised model will compose of. The model was trained until it started plateauing around epoch 120. As an average reconstruction error, it yielded 3×10^{-8} on the data it was trained on, indicating near-perfect reconstruction. When run on the entire dataset, it yielded an average reconstruction error of 7.94×10^{-3}, which is about 14 times the fully unsupervised average. It is as expected significantly higher since it was not trained on cases most likely to be anomalous, which already had a relatively high reconstruction error, hence their reconstruction errors skyrocketed.

Continuing with the same percentiles of 80, 90, 95, and 99, the thresholds as shown in Figure 4, yielded to be 1×10^{-6}, 3.98×10^{-6}, 5.99×10^{-2} and 8.18×10^{-2}.

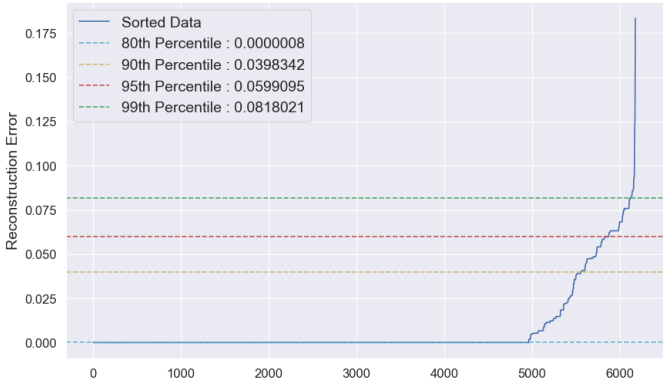

Fig. 4. Semi-supervised Model Recon. Errors and Anomaly Thresholds

With these thresholds at hand, we repeat the same evaluation for the semi-supervised model. As shown in Table II, for the 99th and 95th percentile thresholds we get a perfect success rate of 100% with all the instances labeled being true positives, indicating a significant increase in detection of anomalies. At the 90th percentile threshold, we get an f1-score of 0.70 and a success rate of 91.42%, a significant increase over the unsupervised counterpart of the model. Lastly, at the 80th percentile threshold, we get a success rate of 80.50% alongside an f1-score of 0.89 with all of the 995 anomalies detected. Across thresholds, the semi-supervised model shows an average increase of 0.14 in the f1-score and 44.39% in the success rate, displaying a considerable increase in anomaly detection performance.

TABLE II
SEMI-SUPERVISED MODEL RESULTS

Percentile	Threshold	F1-Score	Succ. Rate	Anomaly:All
99th	8.18×10^{-2}	0.12	100.00	62:62
95th	5.99×10^{-2}	0.47	100.00	309:309
90th	3.98×10^{-6}	0.70	91.42	565:618
80th	1×10^{-6}	0.89	80.50	995:1236

This is evidence of the effectiveness of the implemented approach with respect to the instance selection for autoencoder training. Not only did the semi-supervised model show higher performance in higher thresholds, but it also detected all of the 995 anomalous instances above the 80th percentile threshold.

V. CONCLUSION

The empirical results show that the graph neural network architecture used in the study indicates promising performance, yielding outstanding results for identifying cases with anomalous sub-sequences. Even when used fully unsupervised, the incorporation of ECC layers enables the model to learn from edge attributes and utilize the inferred information. This then enables us to leverage our approach across the different business process workflows throughout the BPM landscape due to; it being possible to standardize the graph creation across different workflows, making it so that no human supervision is needed when creating them and it being possible to repeat the same training process and anomaly labeling scheme to be applied without human supervision.

The semi-supervised model yielding a 44.39% average increase in success rate indicates that the supervision leveraged is appropriate for the task at hand, resulting in a considerable increase in performance. Once again, even what is referred to as supervision in this context, the idea of picking instances for model training below a threshold is achieved without human intervention hence we could argue that this approach could be automated and scaled across the BPM landscape.

In conclusion, the study makes significant contributions to the field of BPM anomaly detection, especially with regard to scalability, applicability, and automation. The study; developed an unsupervised graph neural network model leveraging it for unsupervised anomaly detection eliminating the need for labeled data and it formulated a semi-supervised and generalizable approach that functions in lack of labels for improved accuracy. Both contributions are made without human intervention making it possible to automate them hence, they can be applied to many process records across the BPM landscape. One last contribution we made was to propose a new way to evaluate autoencoder performance above thresholds enabling us to evaluate the performance better in lack of an appropriate metric.

In future work one of our main goals is to develop a more sophisticated approach for threshold calculation. In contrast to using more primitive percentile-based methods, the usage of dynamic thresholds can increase performance and give more flexibility down the lifespan of the anomaly detection framework. Leveraging statistical methods such as a formulation based on the mean and the standard deviation of the reconstruction errors seems particularly fit for such a task. Following this study, future research is expected to seek to evaluate the generalizability of the approach used to determine whether the results hold true for other business process datasets.

ACKNOWLEDGMENTS

This study is supported by the Scientific and Technological Research Council of Turkey (TUBITAK) under grant no TEYDEB-3231136. Additionally, we would like to extend our appreciation to Dr. Siyu Huo and Dr. Guan Wei for their invaluable assistance.

REFERENCES

[1] C. Iscan, O. Kumas, F. P. Akbulut, and A. Akbulut, "Wallet-based transaction fraud prevention through lightgbm with the focus on minimizing false alarms," *IEEE Access*, 2023.

[2] R. Sarno, F. Sinaga, and K. R. Sungkono, "Anomaly detection in business processes using process mining and fuzzy association rule learning," *Journal of Big Data*, vol. 7, no. 1, p. 5, 2020.

[3] E. Sezer, T. B. Ayaz, F. Ozara, and E. Celik, "İş akışı yönetiminde anormallik tespiti Üzerine bir uygulama," in *2. International Scientific Researches Congress*, Ankara, March 9-10 2024. [Online]. Available: https://biselkongreler.com/

[4] T. Nolle, S. Luettgen, A. Seeliger, and M. Mühlhäuser, "Analyzing business process anomalies using autoencoders," *Machine Learning*, vol. 107, 11 2018.

[5] W. Guan, J. Cao, Y. Gu, and S. Qian, "Grasped: A gru-ae network based multi-perspective business process anomaly detection model," *IEEE Transactions on Services Computing*, vol. PP, pp. 1–13, 09 2023.

[6] P. Krajsic and B. Franczyk, "Semi-supervised anomaly detection in business process event data using self-attention based classification," *Procedia Computer Science*, vol. 192, pp. 39–48, 09 2021.

[7] S. Huo, H. Völzer, P. Reddy, P. Agarwal, V. Isahagian, and V. Muthusamy, *Graph Autoencoders for Business Process Anomaly Detection*, 08 2021, pp. 417–433.

[8] M. Simonovsky and N. Komodakis, "Dynamic edge-conditioned filters in convolutional neural networks on graphs," 04 2017.

[9] D. Bank, N. Koenigstein, and R. Giryes, "Autoencoders," *Machine learning for data science handbook: data mining and knowledge discovery handbook*, pp. 353–374, 2023.

[10] U. Michelucci, "An introduction to autoencoders," *arXiv preprint arXiv:2201.03898*, 2022.

[11] S. Zhang, H. Tong, J. Xu, and R. Maciejewski, "Graph convolutional networks: a comprehensive review," *Computational Social Networks*, vol. 6, no. 1, pp. 1–23, 2019.