Graph Based Anomaly Detection in Chimbuko: Feasible or Fallible?

Chase Phelps[1], Ankur Lahiry[1] Tanzima Z. Islam[1], Christopher Kelly[2]
[1] *Texas State University,* [2] *Brookhaven National Laboratory*

Abstract

Detection of performance anomalies can aid in discovering algorithmic inefficiencies or potential hardware issues in an application's environment. The CHIMBUKO framework monitors large-scale workflow applications in real-time and identifies function executions which deviate from accumulated statistics (performance anomalies). Performance anomalies across runs correlate with variation in execution times of an application; quicker resolution of performance anomalies caused by hardware issues improves cluster performance. Anomalous and normal executions are stored as events in CHIMBUKO. In this study, we investigate the applicability of graph-based deep learning methods for anomaly classification. We hypothesize that transforming data into a graph will allow correlations to be modeled, thus allowing graph-based methods to learn embeddings that can improve the effectiveness of downstream anomaly classification tasks. We propose to map events to nodes and calculate edge weights using the distance between features. Our evaluations demonstrate that the graph-based methods yield up to 95% accuracy and outperform a state-of-the-art gradient-based method. Moreover, we provide an explanation of the classification model's decision-making process through explainable AI techniques.

CCS Concepts

• **Computing methodologies → Machine learning approaches**.

Keywords

Real-time anomaly detection, Performance anomaly prediction.

ACM Reference Format:
Chase Phelps[1], Ankur Lahiry[1] Tanzima Z. Islam[1], Christopher Kelly[2], [1] *Texas State University,* [2]*Brookhaven National Laboratory*, . 2023. Graph Based Anomaly Detection in Chimbuko: Feasible or Fallible?. In *Proceedings of SC (SC '23).* ACM, New York, NY, USA, 3 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

CHIMBUKO [7] is a real-time anomaly detection framework that aims to detect anomalies in the performance of large-scale applications. In this context, a performance anomaly refers to a deviation from accumulated statistics of the execution time of a function to a new event's execution time. The CHIMBUKO framework uses the Histogram-based Outlier Score (HBOS) algorithm [4] by default to detect performance anomalies; anomalous events and other events within a configurable execution window are stored in a persistent database called the Provenance Database (ProvDB). Each event includes features such as the rank id, process id, entry timestamp, hardware performance counters collected during execution, and available information provided by the operating system, i.e., counter information from the /proc filesystem. The combined corpora of anomalous and normal executions in the ProvDB is a rich labeled dataset that can advance the state-of-the-art in deep learning-based anomaly classification algorithm development.

In this work, we propose a novel idea of representi ng performance samples as a graph to leverage correlations between samples and features when classifying a new performance sample. Unlike other domains, e.g., social networks, such a graph does not exist. Hence, we propose a method to construct a performance graph by mapping performance samples as nodes and the distance between their features as their edge weights. Following graph construction, we propose to leverage Graph Neural Network (GNN) based approaches, specifically GraphSAGE (GraphSAGE), for learning node embeddings to classify the label of a node.

We evaluate the effectiveness of our proposed approach using CHIMBUKO's labels as the ground truth. Our experiments demonstrate that our proposed graph-based anomaly classification technique outperforms XGBoost [3], a state-of-the-art method used in the literature for tabular data.

2 Related Work and Background

A related work, TabGNN [5], creates a heterogeneous graph from tabular data for classification, but incurs a significant overhead that is not scalable and achieves a much lower relative performance increase compared to Graph Attention Network (GAT) and Graph Convolution Network (GCN) than we do with GraphSAGE.

ProvDB: Of the information provided by CHIMBUKO, e.g., the call stack and elapsed time, we only consider numerical features that have variance and are not nominal or ordinal. In the future, we will also leverage call stack distance in our graph-building method.

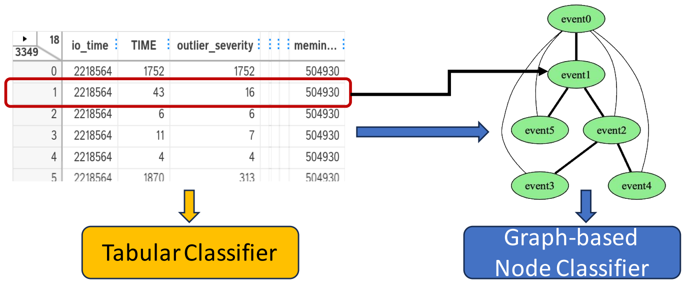
Embeddings: Since the time to build a model depends on the size of this input vector, deep-learning pipelines project the original features into a low-dimensional space called embeddings. A descriptive embedding makes the downstream analytics task effective.

3 Design and Implementation

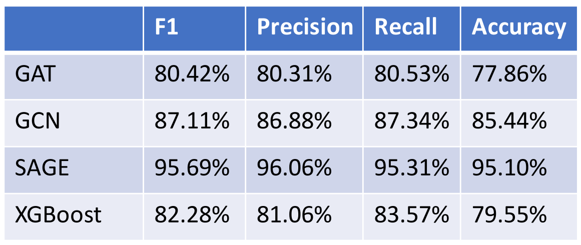
We formulate the problem of anomaly classification as that of node classification, where a graph-based deep learning model can predict the label of a node based on its embeddings.

Graph Construction: As Figure 1a shows, we map each event in the tabular dataset to a node and columns as a feature vector. We add edges between nodes and neighboring nodes with the n nearest distances, as determined by the normalized Euclidean distance between all features. Here, n is a configurable parameter; we use $n = 3$ for this work. The constructed graph induces relationships between execution events labeled as anomalous or normal. Next, we propose to use this graph as input to a graph-based representation learning technique such as GNN to learn an embedding for each node of the graph.

SC '23, November 14–16, 2023, Denver, CO

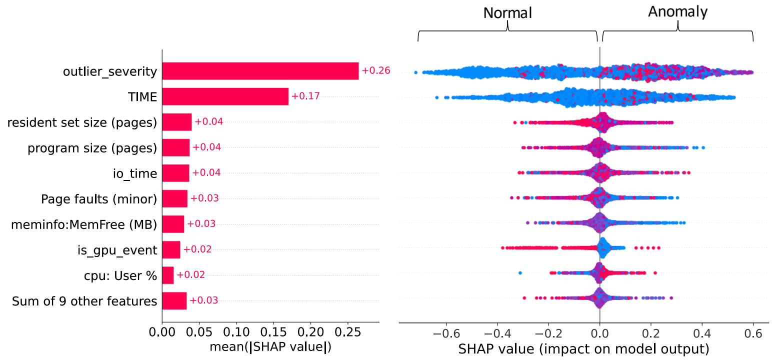


(a) Use of tabular data

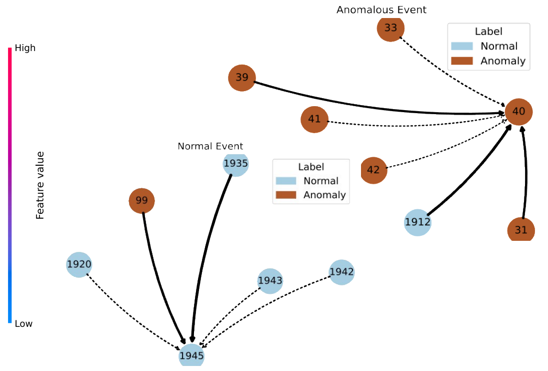


	F1	Precision	Recall	Accuracy
GAT	80.42%	80.31%	80.53%	77.86%
GCN	87.11%	86.88%	87.34%	85.44%
SAGE	95.69%	96.06%	95.31%	95.10%
XGBoost	82.28%	81.06%	83.57%	79.55%

(b) Classification accuracy



(c) Feature importance using SHAP value



(d) Samples that influence GraphSAGE's decision in labeling nodes 1945 as normal and 40 as anomalous

Figure 1: Overall view of our research.

Training: During model training, we split the indices of anomalous and normal events into 5-random train-test-validation sets, evaluate the models on each disjoint test set, and report the average accuracy along with F1 score, precision, and recall. While splitting the dataset, we balance the number of samples per class to avoid creating any class bias. This strategy is important for the anomaly datasets as CHIMBUKO collects many more anomalous samples than normal execution samples. Training is performed on the training data, the validation scores are used to monitor progress, and the model with the highest validation score used during testing.

4 Preliminary Results

We use a unique tabular anomaly dataset collected by running the Grid [2, 11] application, a highly optimized parallel code for lattice QCD simulations, with CHIMBUKO. The dataset contains a total of 3, 349 events, with 1, 905 anomalous and 1, 444 normal executions. We experiment with 3 graph-based classification models, GAT [10], GCN [8], and GraphSAGE [6], and compare our results to the baseline—XGBoost [3], known to work well with tabular data.

Accuracy: Figure 1b shows that both GCN (85%) and GraphSAGE (95%) perform better than XGBoost (80%). *This observation supports our hypothesis; modeling relationships explicitly using a graph improves the accuracy of the downstream classification task.*

Feature Importance: Figure 1c shows the feature importance of the XGBoost classifier as determined through the SHAP [1, 9] library. The left figure shows that the classifiers identify the `outlier_severity` and `TIME` variables, the inclusive and exclusive runtime of a function, respectively, as important. The right-hand figure in Figure 1c shows that anomalous events tend to occur with less free memory and virtual memory size.

Model Explanation: In this work, we leverage GNNExplainer [12] to explain the GraphSAGE model's decision to classify a node as normal or anomalous. Figure 1d depicts two randomly selected nodes of the graph, their incoming edges, and their relative weights. We visualize the edge weights with thicker lines indicating a greater weight and dotted lines indicating less weight. Figure 1d shows that our graph construction process based on feature distance groups nodes with similar labels, which XGBoost does not. In addition, GraphSAGE outperforms both GAT and GCN because GraphSAGE learns a function to generate embeddings for local neighborhoods k-hops away compared to GAT and GCN that create node embeddings through neighborhood feature aggregation 1-hop away. In the future, we will compare the graph-based method with an encoder-based approach such as Variational Autoencoders.

5 Conclusion

In this research, we propose a novel approach of transforming performance data into a graph for improving anomaly classification accuracy. We design and develop a graph construction technique; learn embeddings using GraphSAGE; compare the performance of graph-based models with that of a state-of-the-art tabular model. Our graph-based learning approach outperforms XGBoost by more than 15%. In the future, we will evaluate our approach on more datasets and compare our results with additional models, including TabGNN.

References

[1] 2020. From local explanations to global understanding with explainable AI for trees. *Nature Machine Intelligence* 2, 1 (2020), 2522–5839.

[2] Peter Boyle, Azusa Yamaguchi, Guido Cossu, and Antonin Portelli. 2015. Grid: A next generation data parallel C++ QCD library. arXiv:1512.03487 [hep-lat]

[3] Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. 785–794.

[4] Markus Goldstein and Andreas Dengel. 2012. Histogram-based outlier score (hbos): A fast unsupervised anomaly detection algorithm. *KI-2012: poster and demo track* 9 (2012).

[5] Xiawei Guo, Yuhan Quan, Huan Zhao, Quanming Yao, Yong Li, and Weiwei Tu. 2021. TabGNN: Multiplex graph neural network for tabular data prediction. *arXiv preprint arXiv:2108.09127* (2021).

[6] William L Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*. 1025–1035.

[7] Christopher Kelly, Sungsoo Ha, Kevin Huck, Hubertus Van Dam, Line Pouchard, Gyorgy Matyasfalvi, Li Tang, Nicholas D'Imperio, Wei Xu, Shinjae Yoo, et al. 2020. Chimbuko: A workflow-level scalable performance trace analysis tool. In *ISAV'20 In Situ Infrastructures for Enabling Extreme-Scale Analysis and Visualization*. 15–19.

[8] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[9] Scott M Lundberg and Su-In Lee. 2017. A Unified Approach to Interpreting Model Predictions. In *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.). Curran Associates, Inc., 4765–4774. http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf

[10] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).

[11] Azusa Yamaguchi, Peter Boyle, Guido Cossu, Gianluca Filaci, Christoph Lehner, and Antonin Portelli. 2022. Grid: OneCode and FourAPIs. *PoS* LATTICE2021 (2022), 035. https://doi.org/10.22323/1.396.0035 arXiv:2203.06777 [hep-lat]

[12] Rex Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. 2019. Gnnexplainer: Generating explanations for graph neural networks. *Advances in neural information processing systems* 32 (2019), 9240.

Received 08 August 2023