Journal of Machine Learning Research 25 (2024) 1-9 Submitted 7/23; Revised 2/24; Published 5/24

PyGOD: A Python Library for Graph Outlier Detection

Kay Liu[1*] ZLIU234@UIC.EDU
Yingtong Dou[1,2*] YIDOU@VISA.COM
Xueying Ding[3] XDING2@ANDREW.CMU.EDU
Xiyang Hu[4] XIYANGHU@ASU.EDU
Ruitong Zhang[5] ZHANGRUITONG.ZRT@ALIBABA-INC.COM
Hao Peng[6,7] PENGHCS@GMAIL.COM
Lichao Sun[8] LIS221@LEHIGH.EDU
Philip S. Yu[1] PSYU@UIC.EDU
[1] *University of Illinois Chicago,* [2] *Visa Research,* [3] *Carnegie Mellon University,* [4] *Arizona State University,* [5] *Alibaba Group,* [6] *Kunming University of Science and Technology,* [7] *Shantou University,* [8] *Lehigh University*

Editor: Sebastian Schelter

Abstract

PyGOD is an open-source Python library for detecting outliers in graph data. As the first comprehensive library of its kind, PyGOD supports a wide array of leading graph-based methods for outlier detection under an easy-to-use, well-documented API designed for use by both researchers and practitioners. PyGOD provides modularized components of the different detectors implemented so that users can easily customize each detector for their purposes. To ease the construction of detection workflows, PyGOD offers numerous commonly used utility functions. To scale computation to large graphs, PyGOD supports functionalities for deep models such as sampling and mini-batch processing. PyGOD uses best practices in fostering code reliability and maintainability, including unit testing, continuous integration, and code coverage. To facilitate accessibility, PyGOD is released under a BSD 2-Clause license at `https://pygod.org` and at the Python Package Index (PyPI).

Keywords: outlier detection, anomaly detection, graph learning, graph neural networks

1. Introduction

Outlier detection (OD), also known as anomaly detection, is a key machine learning task to identify deviant samples from the general data distribution (Aggarwal, 2017; Li et al., 2022). With the increasing importance of graph data in both research and real-world applications (Ding et al., 2021b; Huang et al., 2021; Fu et al., 2021; Zhou et al., 2021; Xu et al., 2022), detecting outliers with graph-based methods, particularly graph neural networks (GNNs), has recently garnered considerable attention (Ma et al., 2021; Ding et al., 2019b, 2021a; Liu et al., 2022) with many applications such as detecting suspicious activities in social networks (Sun et al., 2022; Dou et al., 2020) and security systems (Cai et al., 2021).

Although there is a long list of libraries for detecting outliers in tabular and time-series data in multiple programming languages (e.g., PyOD (Zhao et al., 2019), SUOD (Zhao

∗. Equal contribution. The work was done when Yingtong Dou was at UIC.

Algorithm	Backbone	GPU	Sampling	Inductive	Reference
SCAN	Clustering	No	No	No	(Xu et al., 2007)
GAE	GNN+AE	Yes	Yes	Yes	(Kipf and Welling, 2016)
Radar	MF	Yes	No	No	(Li et al., 2017)
ANOMALOUS	MF	Yes	No	No	(Peng et al., 2018)
ONE	MF	Yes	No	No	(Bandyopadhyay et al., 2019)
DOMINANT	GNN+AE	Yes	Yes	Yes	(Ding et al., 2019a)
DONE	GNN+AE	Yes	Yes	No	(Bandyopadhyay et al., 2020b)
AdONE	GNN+AE	Yes	Yes	No	(Bandyopadhyay et al., 2020b)
AnomalyDAE	GNN+AE	Yes	Yes	Yes	(Fan et al., 2020)
GAAN	GAN	Yes	Yes	Yes	(Chen et al., 2020)
DMGD	GNN+AE	Yes	Yes	No	(Bandyopadhyay et al., 2020a)
OCGNN	GNN	Yes	Yes	Yes	(Wang et al., 2021)
CoLA	GNN+AE+SSL	Yes	Yes	Yes	(Liu et al., 2021)
GUIDE	GNN+AE	Yes	Yes	Yes	(Yuan et al., 2021)
CONAD	GNN+AE+SSL	Yes	Yes	Yes	(Xu et al., 2022)
GAD-NR	GNN+AE	Yes	Yes	Yes	(Roy et al., 2024)

Table 1: Implemented graph outlier detectors in PyGOD v1.1.0.

et al., 2021a), PyODDS (Li et al., 2020), ELKI (Achtert et al., 2010), OutlierDetection.jl (Muhr et al., 2022), PyTOD (Zhao et al., 2022), TODS (Lai et al., 2021), Telemanom (Hundman et al., 2018)), there is no specialized library for graph outlier detection.

To bridge this gap, we design the first comprehensive **Py**thon **G**raph **O**utlier **D**etection library called PyGOD, with a couple of key technical advancements and contributions. First, it covers a wide array of algorithms with various backbones, including clustering, matrix factorization (MF), generative adversarial networks (GANs), autoencoders (AEs), GNNs, and self-supervised learning (SSL). PyGOD already supports more than fifteen representative algorithms as shown in Table 1. Second, PyGOD implements these detection models with a unified API so that the user only needs to prepare the data in a predefined graph format, at which point all outlier detectors in PyGOD can process the data. Third, PyGOD offers flexible and modularized components of the different outlier detectors implemented, enabling users to customize these detectors according to individual needs. Morover, PyGOD provides many commonly used utility functions to ease the construction of graph outlier detection workflows. Fourth, PyGOD can scale outlier detection to large graphs using sampling and mini-batch processing. With a focus on code clarity and quality, we provide comprehensive API documentation and examples. Additionally, we provide unit tests with cross-platform continuous integration along with code coverage and maintainability checks.

2. Library Design and Implementation

Dependency. PyGOD builds for Python 3.8+ and depends on the popular `PyTorch` (Paszke et al., 2019) and `PyTorch Geometric (PyG)` (Fey and Lenssen, 2019) packages for graph learning on both CPUs and GPUs. Additionally, PyGOD uses `NumPy` (Harris et al., 2020), `SciPy` (Virtanen et al., 2020), and `scikit-learn` (Pedregosa et al., 2011), and `NetworkX` (Hagberg et al., 2008) for data manipulation.

API Design. As shown in Figure 1, inspired by the API design of `scikit-learn` (Buitinck et al., 2013) and `PyOD` (Zhao et al., 2019), all detection algorithms in PyGOD inherit from

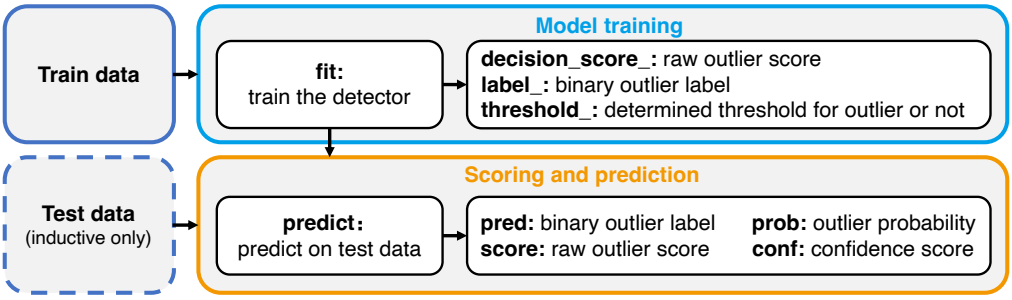

Figure 1: Demonstration of PyGOD's unified API design.

a base class with the same API interface: (*i*) `fit` trains the detector, gets the outlier scores (higher means more outlying) and outlier prediction on the input data, and generates necessary statistics for prediction (in the inductive setting); (*ii*) `predict` leverages the trained model to predict on the input data in the inductive setting (no input data are required in the transductive setting). `predict` returns a binary prediction (0 for normal samples and 1 for outliers), a raw outlier score, a probability of a sample being an outlier (using the method by Kriegel et al. (2011)), and a confidence score (Perini et al., 2020) based on users' needs. The usage of the above APIs is demonstrated in Code Demo 1.

```
from pygod.utils import load_data                # import data function
data = load_data("inj_cora")                     # load built-in dataset

from pygod.detector import DOMINANT              # import the detector
model = DOMINANT(num_layers=4)                   # initialize the detector
model.fit(data)                                  # train with data

pred, score = model.predict(data,                # predict labels by default
                            return_score=True)   # and raw outlier scores

from pygod.metric import eval_roc_auc, eval_f1 # import the metric
eval_f1(data.y.bool(), pred)                     # evaluate by F1
eval_roc_auc(data.y.bool(), score)               # evaluate by AUC
```

Code Demo 1: Using DOMINANT (Ding et al., 2019a) on Cora (Morris et al., 2020).

Streamlined Graph Learning with PyG. We choose to develop PyGOD on top of the popular `PyG` library for multiple reasons. First, this reduces the complexity of processing graph data for users. That is, PyGOD only requires the input data to be in the standard graph data format in PyG[1]. Second, most of the detectors use GNNs (Kipf and Welling, 2017) as their backbone (see Table 1), where PyG already provides an optimized implementation. Third, PyG is the most popular GNN library with advanced functions like graph sampling and distributed training. Under the PyG framework, we implement mini-batch

1. PyG data object: `https://pytorch-geometric.readthedocs.io/en/latest/modules/data.html`

processing and/or sampling for selected models to accommodate learning with large graphs as shown in Table 1.

Modularized components and helpful utility functions. To minimize code redundancy and improve reusability, PyGOD employs modularization in the implementation of deep detectors, dividing different components into `nn.conv`, `nn.encoder`, `nn.decoder`, and `nn.functional`. Additionally, a set of helpful utility functions is designed to facilitate graph outlier detection. In terms of tasks, PyGOD includes `utils.to_edge_score` and `utils.to_graph_score` to enable the adaptation of any node level model to edge level and (sub)graph level outlier detection. In terms of evaluation, PyGOD offers common metrics for graph outlier detection in the `metric` module. In terms of data, PyGOD provides built-in example data sets through `utils.load_data`. Moreover, it offers outlier generator methods in the `generator` module for injecting both contextual and structural outliers (Ding et al., 2019a). This serves as a solution for model evaluation and benchmarking. For more details, please refer to PyGOD documentation[2].

3. Library Robustness and Accessibility

Robustness and Quality. While building PyGOD, we follow the best practices of system design and software development. First, we leverage the continuous integration by *GitHub Actions*[3] to automate the testing process under various Python versions and operating systems. In addition to the scheduled daily test, both commits and pull requests trigger the unit testing. Notably, we enforce all code to have over 99% coverage[4]. By following the `PEP8` standard, we enforce a consistent coding style and naming convention, which facilitates community collaboration and code readability.

Accessibility and Community. PyGOD comes with detailed API documentation rendered by `Read the Docs`. The documentation includes an installation guide as well as interactive examples in `Jupyter notebooks`. To facilitate community contribution, the project is hosted on `GitHub` with a friendly contribution guide and issue reporting mechanism. At the time of publishing, PyGOD has been widely used in numerous real-world applications including Twitter bot detection (Feng et al., 2022) and financial fraud detection (Huang et al., 2022), with more than 1,200 GitHub stars and 20,000 PyPI downloads.

4. Conclusion and Future Plans

In this paper, we present the first comprehensive library for graph outlier detection, called PyGOD. PyGOD supports a wide range of detection algorithms with a unified API, rich documentation, and robust code design. These features make it valuable for both academic research and industry applications. The development plan of PyGOD will focus on multiple aspects: (*i*) enabling detectors to acquire domain knowledge by incorporating different amounts of supervision signals; (*ii*) optimizing its scalability with the latest advancement in graphs (Jia et al., 2020); and (*iii*) incorporating automated machine learning to enable intelligent model selection and hyperparameter tuning (Zhao et al., 2021b).

2. Documentation: `https://docs.pygod.org/`
3. Continuous integration by GitHub Actions: `https://github.com/pygod-team/pygod/actions`
4. Code coverage by Coveralls: `https://coveralls.io/github/pygod-team/pygod`

Acknowledgments

The authors who are affiliated with the University of Illinois Chicago are supported in part by NSF under grant III-2106758 and POSE-2346158. Philip S. Yu and Hao Peng are the corresponding authors.

References

E. Achtert, H.-P. Kriegel, L. Reichert, E. Schubert, R. Wojdanowski, and A. Zimek. Visual evaluation of outlier detection models. In *Database Systems for Advanced Applications (DASFAA)*, pages 396–399. Springer, 2010.

C. C. Aggarwal. An introduction to outlier analysis. In *Outlier Analysis*, pages 1–34. Springer, 2017.

S. Bandyopadhyay, N. Lokesh, and M. N. Murty. Outlier aware network embedding for attributed networks. In *Annual AAAI Conference on Artificial Intelligence (AAAI)*, volume 33, pages 12–19, 2019.

S. Bandyopadhyay, S. Vishal Vivek, and M. Murty. Integrating network embedding and community outlier detection via multiclass graph description. *Frontiers in Artificial Intelligence and Applications (FAIA)*, 325:976–983, 2020a.

S. Bandyopadhyay, S. V. Vivek, and M. Murty. Outlier resistant unsupervised deep architectures for attributed network embedding. In *ACM International Conference on Web Search and Data Mining (WSDM)*, pages 25–33, 2020b.

L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. VanderPlas, A. Joly, B. Holt, and G. Varoquaux. API design for machine learning software: experiences from the scikit-learn project. In *ECML-PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122, 2013.

L. Cai, Z. Chen, C. Luo, J. Gui, J. Ni, D. Li, and H. Chen. Structural temporal graph neural networks for anomaly detection in dynamic graphs. In *ACM International Conference on Information & Knowledge Management (CIKM)*, pages 3747–3756, 2021.

Z. Chen, B. Liu, M. Wang, P. Dai, J. Lv, and L. Bo. Generative adversarial attributed network anomaly detection. In *ACM International Conference on Information & Knowledge Management (CIKM)*, pages 1989–1992, 2020.

K. Ding, J. Li, R. Bhanushali, and H. Liu. Deep anomaly detection on attributed networks. In *SIAM International Conference on Data Mining (SDM)*, pages 594–602. SIAM, 2019a.

K. Ding, J. Li, and H. Liu. Interactive anomaly detection on attributed networks. In *ACM International Conference on Web Search and Data Mining (WSDM)*, pages 357–365, 2019b.

K. Ding, J. Li, N. Agarwal, and H. Liu. Inductive anomaly detection on attributed networks. In *International Conference on International Joint Conferences on Artificial Intelligence (IJCAI)*, pages 1288–1294, 2021a.

K. Ding, Q. Zhou, H. Tong, and H. Liu. Few-shot network anomaly detection via cross-network meta-learning. In *The Web Conference (WWW)*, pages 2448–2456, 2021b.

Y. Dou, Z. Liu, L. Sun, Y. Deng, H. Peng, and P. S. Yu. Enhancing graph neural network-based fraud detectors against camouflaged fraudsters. In *ACM International Conference on Information & Knowledge Management (CIKM)*, pages 315–324, 2020.

H. Fan, F. Zhang, and Z. Li. AnomalyDAE: dual autoencoder for anomaly detection on attributed networks. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 5685–5689, 2020.

S. Feng, Z. Tan, H. Wan, N. Wang, Z. Chen, B. Zhang, Q. Zheng, W. Zhang, Z. Lei, S. Yang, et al. TwiBot-22: towards graph-based Twitter bot detection. In *Annual Conference on Neural Information Processing Systems (NeurIPS)*, volume 35, pages 35254–35269, 2022.

M. Fey and J. E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

T. Fu, C. Xiao, X. Li, L. M. Glass, and J. Sun. MIMOSA: multi-constraint molecule sampling for molecule optimization. In *Annual AAAI Conference on Artificial Intelligence (AAAI)*, volume 35, pages 125–133, 2021.

A. A. Hagberg, D. A. Schult, and P. J. Swart. Exploring network structure, dynamics, and function using networkx. In G. Varoquaux, T. Vaught, and J. Millman, editors, *Python in Science Conference*, pages 11 – 15, Pasadena, CA USA, 2008.

C. R. Harris, K. J. Millman, S. J. Van Der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, et al. Array programming with NumPy. *Nature*, 585(7825):357–362, 2020.

K. Huang, T. Fu, W. Gao, Y. Zhao, Y. H. Roohani, J. Leskovec, C. W. Coley, C. Xiao, J. Sun, and M. Zitnik. Therapeutics data commons: machine learning datasets and tasks for drug discovery and development. In *Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track (NeurIPS)*, 2021.

X. Huang, Y. Yang, Y. Wang, C. Wang, Z. Zhang, J. Xu, L. Chen, and M. Vazirgiannis. DGraph: a large-scale financial dataset for graph anomaly detection. In *Annual Conference on Neural Information Processing Systems (NeurIPS)*, volume 35, pages 22765–22777, 2022.

K. Hundman, V. Constantinou, C. Laporte, I. Colwell, and T. Soderstrom. Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 387–395, 2018.

Z. Jia, S. Lin, M. Gao, M. Zaharia, and A. Aiken. Improving the accuracy, scalability, and performance of graph neural networks with ROC. In *Annual Conference on Machine Learning and Systems (MLSys)*, volume 2, pages 187–198, 2020.

T. N. Kipf and M. Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016.

T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017.

H.-P. Kriegel, P. Kroger, E. Schubert, and A. Zimek. Interpreting and unifying outlier scores. In *SIAM International Conference on Data Mining (SDM)*, pages 13–24. SIAM, 2011.

K.-H. Lai, D. Zha, G. Wang, J. Xu, Y. Zhao, D. Kumar, Y. Chen, P. Zumkhawaka, M. Wan, D. Martinez, and X. Hu. TODS: an automated time series outlier detection system. In *Annual AAAI Conference on Artificial Intelligence (AAAI)*, pages 16060–16062, 2021.

J. Li, H. Dani, X. Hu, and H. Liu. Radar: residual analysis for anomaly detection in attributed networks. In *International Conference on International Joint Conferences on Artificial Intelligence (IJCAI)*, volume 17, pages 2152–2158, 2017.

Y. Li, D. Zha, P. Venugopal, N. Zou, and X. Hu. PyODDS: an end-to-end outlier detection system with automated machine learning. In *Companion Proceedings of the Web Conference (WWW)*, pages 153–157, 2020.

Z. Li, Y. Zhao, X. Hu, N. Botta, C. Ionescu, and G. Chen. ECOD: unsupervised outlier detection using empirical cumulative distribution functions. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, pages 1–1, 2022. doi: 10.1109/TKDE.2022.3159580.

K. Liu, Y. Dou, Y. Zhao, X. Ding, X. Hu, R. Zhang, K. Ding, C. Chen, H. Peng, K. Shu, et al. BOND: benchmarking unsupervised outlier node detection on static attributed graphs. In *Annual Conference on Neural Information Processing Systems (NeurIPS)*, volume 35, pages 27021–27035, 2022.

Y. Liu, Z. Li, S. Pan, C. Gong, C. Zhou, and G. Karypis. Anomaly detection on attributed networks via contrastive self-supervised learning. *IEEE Transactions on Neural Networks and Learning Systems (TNNLS)*, 2021.

X. Ma, J. Wu, S. Xue, J. Yang, C. Zhou, Q. Z. Sheng, H. Xiong, and L. Akoglu. A comprehensive survey on graph anomaly detection with deep learning. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 2021.

C. Morris, N. M. Kriege, F. Bause, K. Kersting, P. Mutzel, and M. Neumann. TUDataset: a collection of benchmark datasets for learning with graphs. In *ICML Workshop on Graph Representation Learning and Beyond*, pages 1–11, 2020.

D. Muhr, M. Affenzeller, and A. D. Blaom. OutlierDetection.jl: a modular outlier detection ecosystem for the julia programming language. *arXiv preprint arXiv:2211.04550*, 2022.

A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. PyTorch: an imperative style, high-performance deep learning library. In *Annual Conference on Neural Information Processing Systems (NeurIPS)*, volume 32, 2019.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. Scikit-learn: machine learning in Python. *Journal of Machine Learning Research (JMLR)*, 12:2825–2830, 2011.

Z. Peng, M. Luo, J. Li, H. Liu, Q. Zheng, et al. ANOMALOUS: a joint modeling approach for anomaly detection on attributed networks. In *International Conference on International Joint Conferences on Artificial Intelligence (IJCAI)*, pages 3513–3519, 2018.

L. Perini, V. Vercruyssen, and J. Davis. Quantifying the confidence of anomaly detectors in their example-wise predictions. In *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD)*, pages 227–243. Springer, 2020.

A. Roy, J. Shu, J. Li, C. Yang, O. Elshocht, J. Smeets, and P. Li. GAD-NR: graph anomaly detection via neighborhood reconstruction. In *ACM International Conference on Web Search and Data Mining (WSDM)*, 2024.

L. Sun, Y. Dou, C. Yang, K. Zhang, J. Wang, S. Y. Philip, L. He, and B. Li. Adversarial attack and defense on graph data: a survey. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 2022.

P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, et al. SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nature Methods*, 17(3):261–272, 2020.

X. Wang, B. Jin, Y. Du, P. Cui, Y. Tan, and Y. Yang. One-class graph neural networks for anomaly detection in attributed networks. *Neural Computing and Applications*, 33(18): 12073–12085, 2021.

X. Xu, N. Yuruk, Z. Feng, and T. A. Schweiger. SCAN: a structural clustering algorithm for networks. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 824–833, 2007.

Z. Xu, X. Huang, Y. Zhao, Y. Dong, and J. Li. Contrastive attributed network anomaly detection with data augmentation. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*, 2022.

X. Yuan, N. Zhou, S. Yu, H. Huang, Z. Chen, and F. Xia. Higher-order structure based anomaly detection on attributed networks. In *IEEE Big Data Conference*, pages 2691–2700, 2021.

Y. Zhao, Z. Nasrullah, and Z. Li. PyOD: a Python toolbox for scalable outlier detection. *Journal of Machine Learning Research (JMLR)*, 20(96):1–7, 2019.

Y. Zhao, X. Hu, C. Cheng, C. Wang, C. Wan, W. Wang, J. Yang, H. Bai, Z. Li, C. Xiao, et al. SUOD: accelerating large-scale unsupervised heterogeneous outlier detection. In *Annual Conference on Machine Learning and Systems (MLSys)*, pages 463–478, 2021a.

Y. Zhao, R. Rossi, and L. Akoglu. Automatic unsupervised outlier model selection. In *Annual Conference on Neural Information Processing Systems (NeurIPS)*, volume 34, pages 4489–4502, 2021b.

Y. Zhao, G. H. Chen, and Z. Jia. TOD: GPU-accelerated outlier detection via tensor operations. In *International Conference on Very Large Data Bases (VLDB)*, volume 16, 2022.

S. Zhou, Q. Tan, Z. Xu, X. Huang, and F.-l. Chung. Subtractive aggregation for attributed network anomaly detection. In *ACM International Conference on Information & Knowledge Management (CIKM)*, pages 3672–3676, 2021.