Cyberattack detection through GPML: Graph Processing for Machine Learning

Majed Jaber, Julien Michel, Nicolas Boutry, Pierre Parrend

SoftwareX 31 (2025) 102308

Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Original software publication

Cyberattack detection through GPML: Graph Processing for Machine Learning

Majed Jaber [a,b], Julien Michel [a,b], Nicolas Boutry [a], Pierre Parrend [a,b,*]

[a] Laboratoire de Recherche de l'EPITA (LRE), 14-16 Rue Voltaire, 94270 Le Kremlin-Bicêtre, France
[b] Université de Strasbourg, CNRS, ICube UMR7357, F-67000 Strasbourg, France

ARTICLE INFO

Keywords:
Graph processing
Machine learning
Python-based library
Spectral graph analysis
Graph communities
Attack detection

ABSTRACT

The dramatic increase of complex, multi-step, and rapidly evolving attacks in dynamic networks involves advanced cyber-threat detectors. The GPML (Graph Processing for Machine Learning) library addresses this need by transforming raw network traffic traces into graph representations, enabling advanced insights into network behaviors. The library provides tools to detect anomalies in interaction and community shifts in dynamic networks. GPML supports community and spectral metrics extraction, enhancing both real-time detection and historical forensics analysis. This library supports modern cybersecurity challenges with a robust, graph-based approach.

Code metadata

Current code version	1.1.0
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX-D-25-00317
Permanent link to Reproducible Capsule	https://www.kaggle.com/code/majedjaber/gpml-reproducible-capsule
Legal Code License	ISC https://opensource.org/license/isc-license-txt
Code versioning system used	git
Software code languages, tools, and services used	python
Compilation requirements, operating environments & dependencies	https://github.com/lre-security-systems-team/gpml/blob/main/requirements.txt
If available Link to developer documentation/manual	https://github.com/lre-security-systems-team/gpml/blob/main/README.md
Support email for questions	julien.michel@epita.fr, majed.jaber@epita.fr

1. Motivation and significance

In today's digital landscape, network security faces growing challenges due to the increasing complexity of cyber-threats [1]. Attackers constantly improve their techniques, targeting vulnerabilities across interconnected devices, systems, and services. Traditional security measures often struggle to keep up, as they primarily rely on signature-based detection [2] or rule-based methods [3], which may not capture emerging threats in network traffic. To effectively monitor, analyze, and secure network environments, new approaches are required—approaches that can handle large-scale and dynamic data from highly interconnected environments while addressing the alert fatigue in security operations [4].

Two main approaches emerge from the literature as shown in Table 1: graph analytics, or complex network analysis, which characterizes the connectivity properties on the graph itself, and embedding analysis, which extracts information about node surroundings for each node. Graph analytics leverages Community, Spectral, or Complex Network information to quantify the relative connectivity of node groups, the structure of the connections between nodes, or the position with respect to the whole network. Embedding analysis typically relies on Graph Neural Networks (GNNs) and their variants. In all cases, graph-based models represent entities as nodes and their interactions as edges and enable effective analysis of network traffic. GNNs extend this by learning at the node, edge, and graph levels [5]. While traditional

* Corresponding author at: Laboratoire de Recherche de l'EPITA (LRE), 14-16 Rue Voltaire, 94270 Le Kremlin-Bicêtre, France.
E-mail addresses: majed.jaber@etu.unistra.fr (M. Jaber), julien.michel@epita.fr (J. Michel), nicolas.boutry@epita.fr (N. Boutry), pierre.parrend@epita.fr (P. Parrend).

https://doi.org/10.1016/j.softx.2025.102308
Received 13 May 2025; Received in revised form 2 August 2025; Accepted 10 August 2025
Available online 27 August 2025

Table 1
Comparison of graph analytics and embedding analysis.

Dimension	Graph Analytics (Complex Network Analysis)	Embedding Analysis (GNN-based)
Core Idea	Quantifies structural graph properties	Learns vector representations from graph structure
Focus	Node/edge connectivity, centrality, community, clustering	Context-aware representation of nodes and edges
Typical Techniques	Centrality, modularity, spectral clustering, PageRank	GCN, GraphSAGE, GAT, edge-aware GNNs
Input Requirements	Graph topology (adjacency)	Graph topology + node/edge features
Output	Structural metrics or clusters	Embeddings for downstream tasks (e.g., classification)
Interpretability	High – metrics are well-defined	Lower – embeddings require interpretation
Use Case in Cybersecurity	Detect anomalies, identify influential nodes or subgraphs	Predict attack edges, classify events or behaviors

GNNs rely on node features and topology [6], recent models integrate edge features to better capture interaction-specific information such as packet volume or connection type. Edge-aware variants like E-GraphSage [7] and NE-GConv [8] incorporate these features during aggregation, improving the precision of edge-level predictions. NE-GConv performs binary edge classification, whereas E-GraphSage supports both binary and multi-class outputs.

Dynamic Graph Community (DGC) metrics significantly enhance detection performance in network traffic classification tasks. Enriching the baseline feature set, it captures dynamic interaction patterns in network data. The spectral method in GPML library ($SPECTRA$) analyzes graph states over time using time windows and classifies attacks based on aggregated behaviors. While GNNs aim to identify specific malicious edges, $SPECTRA$ detects broader anomalous evolutions over multiple attack categories.

The GPML library integrates both community and spectral analytics to leverage a graph-based approach to network security analysis. This library transforms raw network traffic data into graph representations [9]. This approach not only enhances the detection of known issues but also reveals hidden or emerging patterns within complex networks. By identifying interconnections through communities and spectral analysis, and monitoring their changes over time, it becomes possible to detect unusual connectivity patterns, isolate suspicious nodes, and proactively respond to potential threats. Tracking both static and dynamic network metrics enables real-time and historical analysis, both essential for network evaluation. GPML library builds on established methods in network graph analysis and leverages widely used libraries like NetworkX [10] for graph operations and Pandas [11] for data handling.

In future studies, the GPML library could serve as a foundational tool for developing AI-driven solutions to detect sophisticated threats.

2. Software description

2.1. Supported metrics

This section explains the community and spectral metrics we introduced for detecting attacks over the network.

- **Community-based Metrics:** The graph community metrics are calculated from a graph community partition at time t, and their dynamicity is calculated from the difference between time $t+1$ and t. Let V_t be the set of nodes of a community at time t.
  - **Stability:** Measures similarity between community states at time t and $t+1$.

    $$\text{Stability} = \frac{|V_t \cap V_{t+1}| - |(V_t \cap \bar{V}_{t+1}) \cup (V_{t+1} \cap \bar{V}_t)|}{|V_t \cup V_{t+1}|}$$

  - **Density:** Probability that a node is adjacent to any other node within the community.
  - **Conductance:** Proportion of communications pointing outside the community.
  - **Degree:** Number of edges going out from a node.
- **Spectral-based Metrics:** The spectral metrics are derived from the spectrum Λ_t of the Laplacian matrix at time t [9]. We denote by $\Lambda_t[i]$ the ith eigenvalue, $i \in [1, n]$, sorted in increasing order, and by $\mathcal{Z}(t)$ the multiplicity of zero in Λ_t.
  - **Connectedness:** Quantifies network interconnectivity using the Laplacian's zero eigenvalue multiplicity $\mathcal{Z}(t)$.

    $$\text{Connectedness} = \exp\left(\frac{1}{\mathcal{Z}(t)} - 1\right)$$

  - **Flooding:** Evaluates average central eigenvalues beyond the zero multiplicity.

    $$\text{Flooding} = \left(\frac{1}{\mathcal{N}} \sum_{i=\mathcal{Z}(t)+1}^{\mathcal{Z}(t)+\mathcal{N}} \Lambda_t[i]\right) - 1$$

  - **Wiriness:** Reflects the mean of the highest eigenvalues in the spectrum.

    $$\text{Wiriness} = \frac{1}{\mathcal{N}} \sum_{i=n-\mathcal{N}+1}^{n} \Lambda_t[i]$$

  - **Asymmetry:** Counts the number of significant spectral gaps.

    $$\text{Asymmetry} = \text{Card}\left\{i \geq 2 \mid \Lambda_t[i] - \Lambda_t[i-1] > 10^{-12}\right\}$$

2.2. Software functionalities

The library functionalities can be divided into three main parts:

- Extracting community graph features, as shown in Fig. 1,
- Extracting spectral graph features, as shown in Fig. 2,
- Plotting connectivity graphs.

Additionally, because the library requires correct inputs to function normally, you must make sure your dataset contains the following common features that exist in every traffic network data:

- Timestamp for the arriving packets.
- Source and destination IP addresses.
- For spectral metrics extraction, you need in addition to the above, the total number of packets, size of bytes and the rate of packets. These features exist in every traffic network data logs.

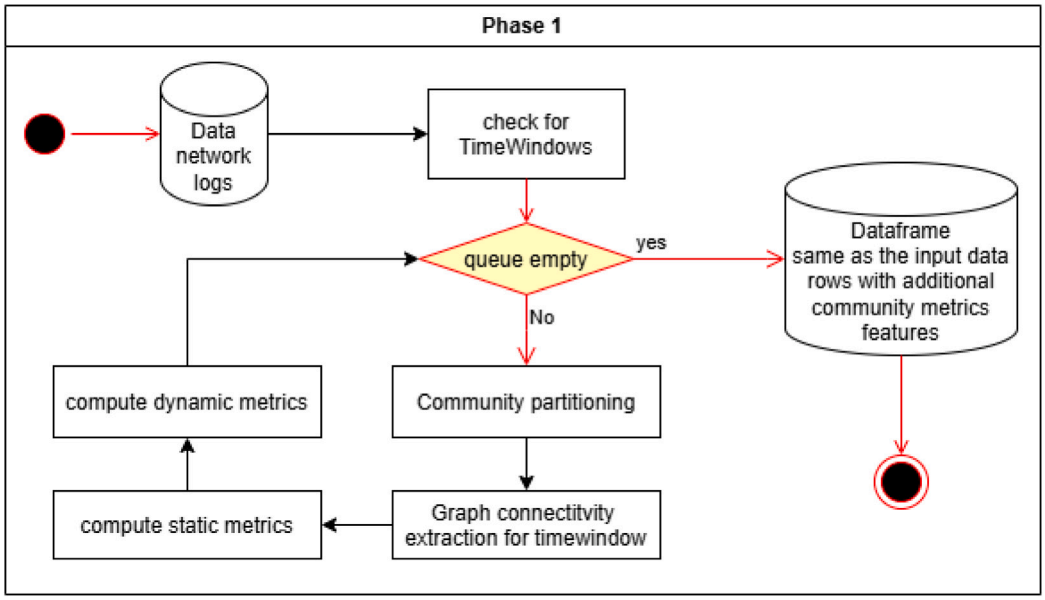

Fig. 1. UML workflow diagram community strategy.

The library provides a set of functionalities that help to add new features for better predictions. The main functionalities are:

```
# Extract time series features grouped by and sorted by
    specific keys
extract_time_series(df, stime, time_unit,
    features_list, sortby_list, groupby_list,
    aggregation_dict)
```

Listing 1: Extraction of time series

In `extract_time_series`, a default time interval of `t=1` s is applied to the dataframe `df` provided by the user. This transformation converts the dataset into a time series format, typically reducing the number of rows compared to the original dataset.

```
extract_community_metrics(dataframe, time_interval,
    date_time, edge_source, edge_dest, label,name,
    community_strategy)
```

Listing 2: Insertion of graph community metrics

The `extract_community_metrics` function takes a *pandas DataFrame* and for a given time window (`time_interval`) and community partitioning strategy (`community_strategy`) as the parameters will produce the corresponding community metrics and insert them back to the *DataFrame* as columns. All the community partitioning strategies, such as the Louvain algorithm, used in this function are imported from existing libraries. The input *DataFrame* is divided in time windows, for each time windows primarily, three underlying method will be called by this function: `gc_metrics_first_order(G)` which calculates metrics by one travel of the graph *G*, `gc_metrics_second_order (forder_metrics_c, forder_metrics_g)` which calculates metrics by one travel over the first order metrics, and `propagate _communities(g1, g2, center, center_t)` which creates the correspondence of communities from two graph at consecutive time windows. Dynamic community metrics are then computed from those metrics and propagated through communities; all of them are added to the output *DataFrame*.

```
extract_spectral_metrics(ts, stime, saddr, daddr, pkts
    , bytes_size, rate, lbl_category, src_pkts,
    dst_pkts, dst_bytes, duration)
```

Listing 3: Extraction of spectral metrics

In `extract_spectral_metrics`, the parameters are provided by the user; however, constructing weighted edges within the network graph extracted over each time window requires selecting a specific feature. For this purpose, `pkts`, `bytes_size`, and `rate` are passed as inputs to weigh the graph during processing across three distinct topologies within each time window. Spectral metrics are then computed for each time window at the midpoint and the end. The output of this function is a new dataframe where each row represents a time window, including its corresponding common features, spectral features, and the associated label.

```
print_graph(dataset, graph_type, label, src_addr,
    dst_addr, sport, dport, url, title, attack_name,
    src_mac, dst_mac)
```

Listing 4: Representing graph based on network features

The `print_graph` function is designed to display the graph in two formats: a non-interactive format using *networkx* and an interactive *HTML* format that allows user interactions.

3. Evaluation

The evaluation highlights the effectiveness of both community and spectral approaches in detecting network threats. The community-based method, referred to as Dynamic Graph Community (DGC) as shown in Fig. 3, demonstrates improved metrics on the UGR16 dataset.[1] For binary predictions, DGC shows Matthews Correlation Coefficient (MCC) of 0.96, Balanced Accuracy of 0.96 and True Positive Rate (TPR) of 0.93, surpassing the baseline with MCC of 0.52 and TPR of 0.44, as well as graph and graph community models on a 5-folds evaluation. In multi-class predictions, DGC achieves high F1-scores, notably 0.89 for anomaly-spam and 0.99 for both scan44 and scan11 attacks, against respectively 0.01 for anomaly-spam, 0.69 for scan44 and 0.11 for scan11 in the baseline. Time performance analysis reveals prediction times from around 1s to 1.5 s, though fitting times increase from 16.15 s to 67.95 s due to model complexity.

We adopt four distinct approaches to classify network traffic data. The first approach, *Classification on Original Data-logs (COD)*, uses the

[1] https://nesg.ugr.es/nesg-ugr16/

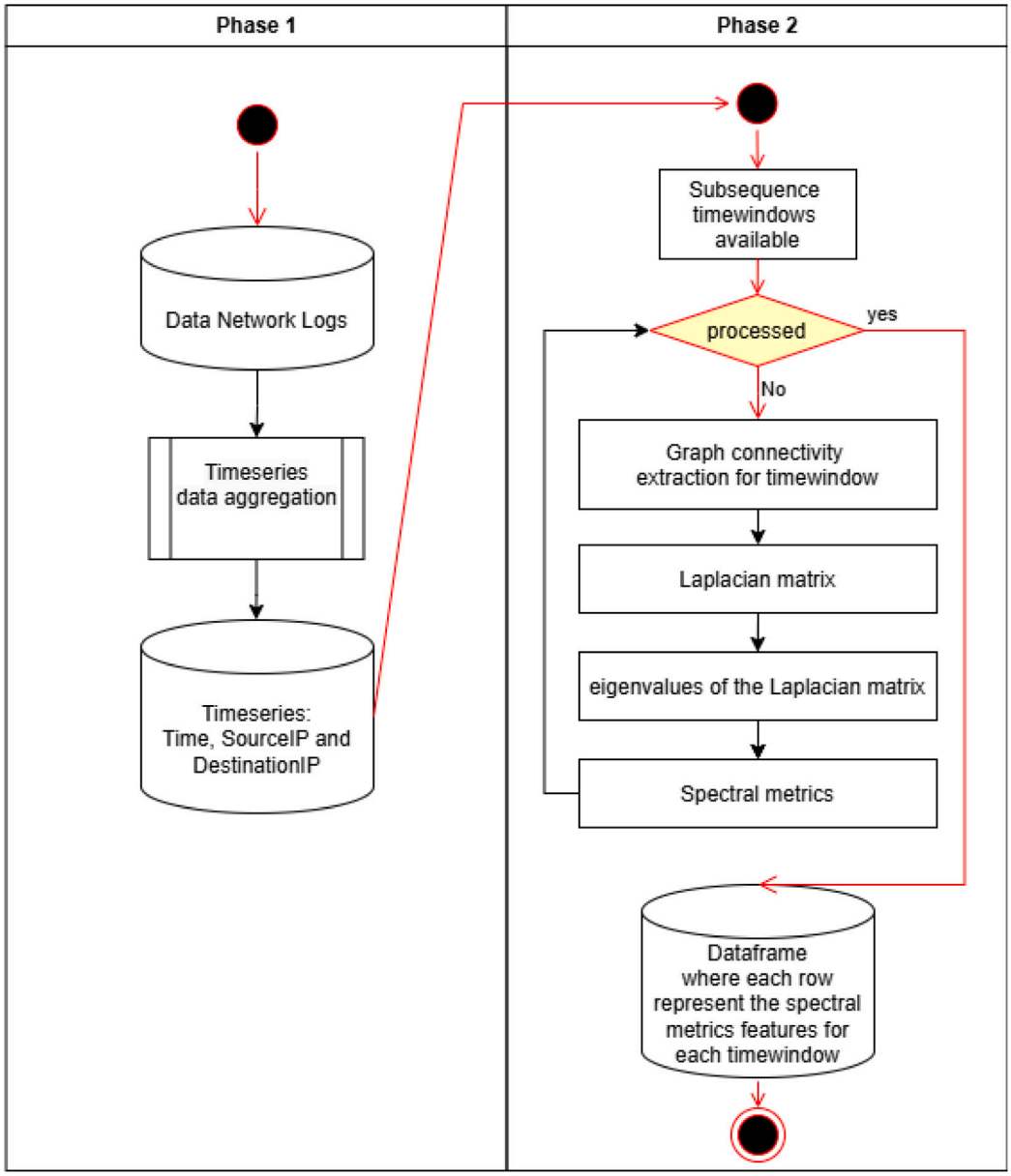

Fig. 2. UML workflow diagram for spectral graph strategy.

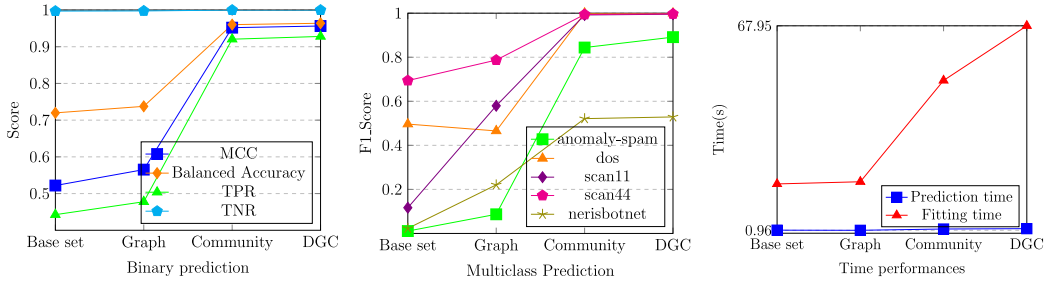

Fig. 3. Comparison of graph community approaches with baseline on UGR16 dataset with 5-folds evaluation using XGboost. Base set is original dataset feature space, the other ones are the same dataset enriched with incrementally: graph metrics, graph community metrics and dynamic graphe community metrics.

raw dataset with minimal filtering, excluding only time and sequence-related features to avoid bias from temporal patterns. The second approach, *Classification on Time-Series (CTS)*, introduces a time-series transform mechanism to segment traffic logs data into series of data and extract basic quantitative features like packet counts and rates, aiming to capture evolving traffic behaviors without relying on topological

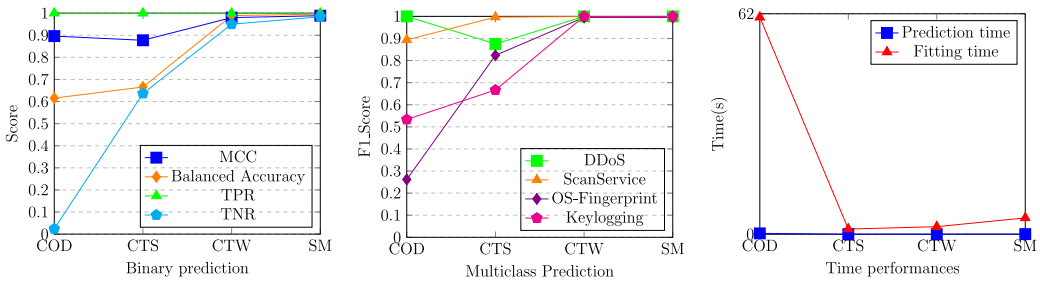

Fig. 4. Comparison of spectral graph approaches with baseline on Botnet dataset.

Table 2
Comparison study between E-GraphSage, EN-GConv and SPECTRA over Botnet IoT dataset for binary classification.

	E-GraphSage	EN-GConv	SPECTRA
F1 Score	0.9888	0.3322	**0.9944**
MCC	0.7772	0.3892	**0.9936**
ADR (%)	97.86	97.14	**99.52**
Precision	**0.9993**	0.2004	0.9936

structures. The third approach, *Classification on Time-Windowing (CTW)*, enhances CTS by aggregating data within each time window into new datasets labeled based on the presence of attacks, improving temporal context while still focusing on quantitative attributes. The fourth and final approach, *Spectral Metrics (SM)*, constructs graphs within each time window and splits them into two sub-windows to observe structural changes. It then extracts spectral metrics from Laplacian matrices of these subgraphs, providing topological insights and enabling a graph-based representation of dynamic network behavior for classification. These approaches are detailed in our previous work [9] for further details.

The spectral approach, represented as SM (Spectral Metrics) as shown in Fig. 4, proves highly effective on the Botnet dataset.[2] For binary predictions, SM reaches near-perfect metrics, including an MCC of 0.91 out of approximate of 0.9 for the COD, Balanced Accuracy of 0.97 out of 0.61 for COD, TPR of 0.93, and TNR of 0.99 out 0.01 for COD. In multi-class predictions, SM achieves excellent F1-scores, such as 0.99 for ScanService and 0.91 for DDoS attacks. Despite higher fitting times at 62 s for COD, prediction times remain efficient across all configurations.

For a full-fledged graph community metrics analysis, extraction time is 65.3 s for 54.10^3 occurrences, 4.10^3 seconds for $5.38.10^6$ occurrences, and 70.10^3 seconds for 53.10^6 occurrences, so the process is roughly linear. For analysis on a given time window only, extraction time is 9.8 s for 53.10^3 occurrences, and 10.10^3 sec for 53.10^6 for 5 min slots; and 13.8 s for 53.10^3 occurrences, and 23.10^3 sec for 53.10^6 for 20 min slots. The extraction is one on a machine with Intel Core i5-10300H processor at 2.5 MHz and 15,8 Gb RAM.

A comparison evaluation between $SPECTRA$, E-GraphSage and EN-GConv is performed for Botnet [12] and TonIoT,[3] as shown in Table 2.

DGC and SM address advanced network security challenges through complementary strengths. DGC excels in analyzing large-scale networks by identifying clusters based on community-driven patterns, which helps detect anomalies like insider threats or advanced persistent threats. On the other hand, SM leverages mathematical rigor to uncover structural insights, identifying hidden substructures such as covert communication channels or botnets.

[2] https://research.unsw.edu.au/projects/bot-iot-dataset
[3] https://research.unsw.edu.au/projects/toniot-datasets

4. Impact

The software opens up avenues for addressing new research questions throughout the pipeline for attack detection. One area of exploration involves graph-based models. Researchers can compare various graph structures and complex network metrics, evaluating their utility in the context of cybersecurity. Additionally, Table 3 shows a comparison between graph convolutional models (GCNs), where learning operates directly on graph data, and graph-derived features, where learning operates on tabular data, can reveal insights into their detection capabilities, time performance, complexity, and support for parallelization. A promising direction involves exploring how traditional graph metrics and knowledge graphs complement each other to improve detection methods.

For attack detectors, the software facilitates characterizing detectors for specific types of attacks, defining weak signal analysis to improve detection capabilities, and advancing feature engineering tailored to attack scenarios. Explainability remains a critical focus, providing transparency and interpretability in the detection process.

The software significantly enhances the pursuit of existing research questions in several ways. It enables the automated extraction of derived features, streamlining the process of preparing data for machine learning models. Similarly, it automates the computation of graph metrics, reducing manual effort and increasing efficiency. Moreover, the systematized visualization and evaluation framework provided by the software improves the ability to assess and interpret machine learning approaches for attack detection, enabling more robust and reproducible research outcomes.

The GPML library is structured to support extensibility and reusability across various graph-based machine learning tasks. Its modular architecture-separating graph representation, processing strategies, and model logic allows experts to add new community, spectral or other detection algorithms, metrics, or embedding methods without modifying core components. It is also dataset-agnostic, making it applicable to multiple domains such as cybersecurity or social networks. It integrates easily into Jupyter notebooks or Python scripts.

5. Conclusions

The GPML (Graph Processing for Machine Learning) library provides a robust and scalable solution for addressing the challenges posed by advanced cyber-threats. By leveraging graph-based methodologies, it enables the detection of complex attack patterns, community dynamics, and emerging anomalies in network environments. The library integrates community and spectral graph analysis with powerful Python tools, offering advanced functionalities for temporal graph processing, feature extraction, and visualization. These capabilities not only enhance threat detection but also support deeper exploration of network behaviors, making GPML a valuable resource for researchers and practitioners in cybersecurity.

Future work can extend the GPML library along several practical and research-oriented directions. One key area is the integration of real-time graph stream processing. This would allow GPML to operate

Table 3
Comparison of graph convolutional models (GCNs) and graph-derived features.

Comparison Dimension	GCNs (Graph Convolutional Networks)	Graph-Derived Feature Models
Input Data Type	Raw graph structure with node/edge attributes	Tabular data derived from graph metrics
Learning Context	End-to-end on graph topology	On extracted features (e.g., centralities)
Detection Capability	High with complex patterns	Effective for known attack structures
Runtime Performance	Higher training cost, slower inference	Fast training and inference
Model Complexity	High (multi-layer neural network)	Low (tree-based or classical models)
Parallelization	Node- or batch-level	Sample-level (easier scaling)
Explainability	Low (harder to trace decisions)	High (feature importance interpretable)
Integration with Traditional Graph Metrics	Implicit (via learned embeddings)	Direct (features used explicitly)
Integration with Knowledge Graphs	Requires adaptation or hybrid modeling	Easily supports semantic features

on live network data, enabling the detection of anomalies and attack patterns as they unfold, rather than post hoc. This can be achieved by incorporating incremental graph algorithms and memory-efficient data structures that support streaming edge updates and dynamic feature recalculations. Another direction is the development of AI-driven threat detection pipelines that use graph embeddings generated by GPML as input to deep learning models such as GNNs. For instance, embeddings extracted from temporal communities or spectral signatures could feed into classifiers trained to recognize specific attack tactics or anomalies, improving detection accuracy across diverse environments. In addition, expanding support for distributed graph processing (e.g., integration with Dask or PySpark) could allow scalability to large-scale infrastructures such as enterprise networks or cloud environments. GPML can also be extended to support interoperability with cybersecurity platforms (e.g., SIEM tools) to facilitate direct deployment in operational settings. These extensions will strengthen GPML's role in adaptive, automated threat detection and open up opportunities for further research in cyber situational awareness and resilient defense systems.

CRediT authorship contribution statement

Majed Jaber: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Julien Michel:** Writing – review & editing, Writing – original draft, Validation, Software, Methodology, Investigation, Data curation, Conceptualization. **Nicolas Boutry:** Writing – review & editing, Validation, Supervision, Methodology, Funding acquisition, Formal analysis, Conceptualization. **Pierre Parrend:** Writing – review & editing, Validation, Supervision, Software, Project administration, Methodology, Investigation, Funding acquisition, Conceptualization.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Pierre Parrend reports financial support was provided by EPITA College of Informatics and Advanced Techniques. Majed Jaber reports a relationship with La Région Grand Est that includes: funding grants. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors acknowledge the support of the Région Grand-Est and École Pour l'Informatique et les Techniques Avancées (EPITA), France for the joint funding of the XDGMed Project.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.softx.2025.102308.

References

[1] Uma M, Padmavathi G. A survey on various cyber attacks and their classification. Int J Netw Secur 2013;15(5):390–6.
[2] Kothamali PR, Banik S. Limitations of signature-based threat detection. Rev Intel Artif Med 2022;13(1):381–91.
[3] Sarker IH, Janicke H, Ferrag MA, Abuadbba A. Multi-aspect rule-based AI: Methods, taxonomy, challenges and directions toward automation, intelligence and transparent cybersecurity modeling for critical infrastructures. Internet Things 2024.
[4] Tariq S, Baruwal Chhetri M, Nepal S, Paris C. Alert fatigue in security operations centres: Research challenges and opportunities. ACM Comput Surv 2025.
[5] Wu Z, Pan S, Chen F, Long G, Zhang C, Yu PS. A comprehensive survey on graph neural networks. IEEE Trans Neural Netw Learn Syst 2020;32(1):4–24.
[6] Liu J, Ong GP, Chen X. GraphSAGE-based traffic speed forecasting for segment network with sparse data. IEEE Trans Intell Transp Syst 2020;23(3):1755–66.
[7] Lo WW, Layeghy S, Sarhan M, Gallagher M, Portmann M. E-graphsage: A graph neural network based intrusion detection system for iot. In: NOMS 2022-2022 IEEE/iFIP network operations and management symposium. IEEE; 2022, p. 1–9.
[8] Altaf T, Wang X, Ni W, Liu RP, Braun R. NE-GConv: A lightweight node edge graph convolutional network for intrusion detection. Comput Secur 2023;130.
[9] Jaber M, Boutry N, Parrend P. Graph-based spectral analysis for detecting cyber attacks. In: Proceedings of the 19th international conference on availability, reliability and security. 2024, p. 1–14.
[10] Hagberg A, Conway D. Networkx: Network analysis with python. 2020, URL: https://Networkx.Github.Io.
[11] McKinney W, et al. Pandas: a foundational python library for data analysis and statistics. Python High Perform Sci Comput 2011;14(9):1–9.
[12] Koroniotis N, Moustafa N, Sitnikova E, Turnbull B. Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-iot dataset. Future Gener Comput Syst 2019;100:779–96.