Graph Neural Networks on Phase Space Graphs for Cybersecurity

Parker Cole, Ryan Benton, Ralf Riedel, David Bourrie, Rebecca Clark and Todd McDonald
School of Computing, University of South Alabama, Mobile, USA

phc1721@jagmail.southalabama.edu

Abstract: Non-linear phase space analysis may be used to represent time-series data as graph data with transitions between states in the time domain. By studying these transitions, we can predict anomalies within the system. Previous research has demonstrated success in learning from phase graphs for malware and seizure detection. These solutions either require extracting global features or converting the graph into an image for convolutional neural networks (CNNs), which adds complexity and limits the potential expressiveness of a graph. To sidestep current limitations, this study proposes Graph Neural Networks (GNNs) for analyzing phase graphs. Unlike CNNs, which must transform graph data into an image representation that may not preserve isomorphism, GNNs operate on the graph data itself through message-passing mechanisms that naturally preserve graph structure. Four GNN architectures were evaluated on two cybersecurity datasets: the Canadian Institute for Cybersecurity Intrusion Detection System (CICIDS) 2017 dataset and a power usage dataset for rootkit detection. The CICIDS dataset was processed at three density levels by varying the symbol parameter. Findings reveal GNNs can be used successfully with phase space graphs, that the type of GNN impacts classification accuracy, and that variance in phase space parameters also impacts accuracy. GIN achieved the most consistent performance across all experiments, while graph density significantly affected results.

Keywords: Graph neural networks, Phase space analysis, Cybersecurity, Intrusion detection, Rootkit detection, Time-series classification

1. Introduction

Time series data in computing has been used to represent diverse processes in the frequency and time domains. Sensor readings, for instance, are a very common source of time series data, where machine learning (ML) has assisted in understanding patterns hidden in data (Fu et al, 2024). Despite its importance and availability, time-series data remains one of the most challenging types of data for machine learning (Jadon et al, 2021). Methods for processing time series data may mitigate such challenges and possibly increase the predictive power of ML algorithms (Zhao et al, 2017), potentially reducing the time and capital investments needed to create effective models.

Phase space analysis is a method that can be adapted to model time-series data as graphs representing states and transitions. The phase space graph of a system may capture the changes in the underlying dynamics and may be representative of its behavior over time (Protopopescu and Hively, 2005). This method has proven to be a promising and historically effective way to predict states within time series data, such as predicting heart disease based on neurological data (Stuckey et al, 2018).

Current time series methods using phase space have their respective limitations. The standard mathematical approach of finding optimal parameters and thresholds requires extensive time and is difficult to scale without domain-level knowledge (Stuckey et al, 2018). Machine-learning approaches have included integration with Convolutional Neural Networks (CNNs) and ML-based solutions for parameter selection (Callegari, 2024). These methods are challenging because they must transform graph-data into an image representation that will preserve isomorphism of its temporal information by the pooling layers of a CNN.

The proposed solution is the application of Graph Neural Networks (GNNs), which operate on graph data directly through message-passing mechanisms where data from all vertices are exchanged with neighboring nodes, naturally preserving graph structure without requiring intermediate transformations. Due to limited information about GNNs on phase space graphs, there is a lack of understanding which GNNs perform best, and what strategies work best for converting time series data for GNNs using phase space (Neupane et al, 2022).

This research aimed to address this shortcoming by presenting two experiments which apply GNNs to cybersecurity data processed using phase space. Additionally, this research aimed to observe the impact of changing the size of the phase space graphs by generating variants of one dataset and analyzing the results.

2. Background

2.1 Rootkits and Side-channel Analysis

A rootkit is software that stays dormant on a device until a malicious party activates it to gain escalated privileges, more formally defined as "a software that is developed to enable its associated programs to conceal

their presence from analysis and detection while maintaining privileged access to a target system" (Kim et al, 2012, p.134). Detection of rootkits has become an evolving arms race due to their concealment capabilities (Lobo et al, 2010). Operating systems organize access levels into a "ring" system where user-level access is ring three and kernel level access is ring zero (Kim et al, 2012). If a rootkit obtains ring zero permissions, it can manipulate kernel-level functionalities, obscuring detection of other code on the machine.

Side channel analysis may aid in minimizing rootkit threats by examining data from computer hardware leakages, like voltage or CPU-generated heat. This is effective because any level of access produces hardware leakage. Kyte et al (2012) describe an effective side-channel method to identify hardware virtualization-based rootkits using energy consumption data.

2.2 Phase Space Analysis

The means by which this research converts time-series data into graphs for GNNs is via the phase space algorithm. This algorithm converts time series data into graph data by utilizing the fundamentals behind Takens' Embedding Theorem (Robinson, 2005), which provides a method for time-delay embedding, allowing for the representation of time series data as high-dimensional graphs of state changes. The algorithm created by Protopopescu and Hively (2005) replaces the continuous numeric data with enumerable symbols that represent ranges, divides the data into "cut sets", and then generates links between these cut sets. Figure 1 illustrates this process.

The algorithm has a few key parameters that control the output, being the number of base-cases, number of points per cut-set, the number and size of the symbols, the method of symbolization, and the pattern in which links are assigned (McDonald et al, 2022).

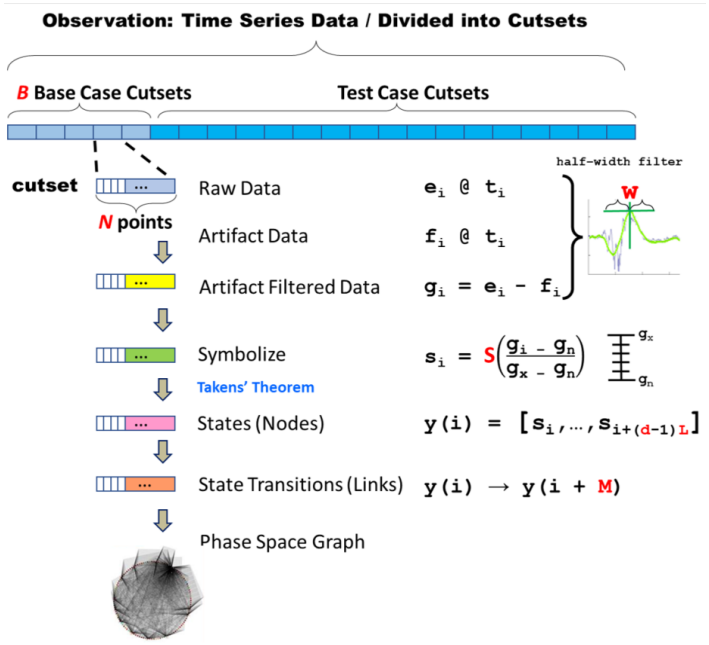

Figure 1: Phase space algorithm process showing transformation from time-series data to graph representation (reproduced from Callegari, 2024)

The phase space algorithm transforms continuous numeric data into discrete graph data (Luckett et al, 2019a). In order to achieve this, it must have a binning step that discretizes these continuous values into ranges known as "bins". The number of total bins is controlled by the "symbols" parameter. A higher symbol value results in more nodes and more connections throughout the resulting phase space graph. This is because when the data is encoded into a higher number of symbols there is a higher number of permutations of state changes, which results in fewer redundancies and more unique edges and nodes in the end graph datasets.

Existing work has been done with this algorithm to successfully prove that as an embedding method it is able to at least maintain some aspects of the original dataset to a degree where it can be effectively used to classify and make predictions about dynamical systems (Callegari, 2024). A study in healthcare using non-linear phase space analysis to time-series data showed promise with wave-type data classification (Stuckey et al, 2018).

2.3 Graph Neural Networks

Time series data being portrayed as graph data is a well-established concept in the machine learning community, and there exists a history of converting time-series data into graphs for the use in GNNs (Jin et al, 2024). By viewing temporal data as a series of moments that occur after the previous event yet before the next event, it is natural to attempt to portray this data in some form of graph. One such experiment used graph neural networks with sensor data to detect anomalous events (Deng and Hooi, 2021). Phase space analysis allows for a different projection of this data by treating time as a semi-cyclic dimension that allows for better observation regarding repetitive or cyclical behavior within the system (Orlik, 2012).

The first model considered was the standard Graph Convolution Network (GCN), one of the earlier networks to be developed (Zhang et al, 2019). GCNs operate on the principle of graph message passing. Message passing is a mechanism where data from all vertices are exchanged with neighboring nodes to disperse graph-wide information between neighbors. Each node contains a feature vector with information about its neighbors, resulting in each node having a vector that represents neighborhood-level information. Message passing is a common feature in graph networks and is the basis of several models (Xu et al, 2019).

The Graph Isomorphism Network (GIN) is a prime focus in this study due to it performing the best among benchmarks (Errica et al, 2022). The GIN model was developed to address graph isomorphism, when equivalent graphs may be evaluated as different due to their representation. GIN uses a sum-aggregation function, which is more expressive in capturing isomorphic graph data (Bouritsas et al, 2023). The GIN model is proven to be as expressive as the Weisfeiler-Lehman graph isomorphism test, described as "a powerful test known to distinguish a broad class of graphs" (Xu et al, 2019, p.2). GIN updates the feature nodes as the GCN does via message passing, with the addition of producing feature vectors more suited for isomorphism tests. This expressiveness is particularly relevant for phase space graphs, where structurally equivalent graphs representing the same dynamical states must be recognized as such regardless of their representation.

EdgeCNN is similar to GCN but performs message passing between edges rather than nodes, which can be effective with connected graphs but less helpful for sparse ones (Yang et al, 2019). GraphSAGE presents the advantage of inductive learning, learning sub-patterns within training graphs to generalize later (Hamilton et al, 2018). This is important for phase space graphs due to their sensitivity to construction parameters.

2.4 Graph Density Considerations

Although there is no formal definition for "dense" and "sparse" graphs (Bollobas and Riordan, 2010), the distinction is important for this work. GNNs encode information into each node based on connected neighbors via message passing (Errica et al, 2022). A sparse graph with few edges yields simpler node-vectors with less information per node, analogous to a CNN trained on low-resolution images. Conversely, overly dense graphs provide rich contextual information at the expense of higher dimensional complexity, increasing the chance of over-fitting and training time.

2.5 Comparison with Traditional Approaches

Traditional machine learning approaches to phase space graph classification have followed two strategies. The first extracts global features such as graph density or spectral properties and feeds these into classifiers (Luckett et al, 2019b), requiring careful feature engineering. The second, explored by Callegari (2024), converts phase space graphs into image representations for CNNs. As described by Luckett et al (2019b), the graph adjacency matrix encodes node and edge information into a rasterized format for convolutional neural networks. This approach benefits from deep learning but faces the challenge of isomorphism: two equal graphs may be represented differently and considered as not equal.

GNNs address these limitations by operating directly on graph structures, preserving node connectivity and edge relationships through message-passing operations, eliminating the conversion step entirely.

3. Methodology

3.1 Datasets

The rootkit dataset consists of power level readings from several trials where a computer's activity was recorded while running Linux and Windows operating systems with or without a rootkit (McDonald et al, 2022). The combined dataset consists of data from 50 trials and 2,220 phase space graphs, comprised of exactly 50% benign and 50% malicious samples. The providers only supplied the data in graph format; the raw data was not provided.

The CICIDS 2017 dataset (University of New Brunswick, 2024) is a well-established benchmark for network intrusion detection containing labeled malicious and benign network traffic (Neupane et al, 2022). To observe the impact of the symbol count parameter, which controls graph size and connectivity by binning numeric data into different numbers of symbols, three variants were generated with values of two, three, and five symbols, representing sparse, nominal, and dense graph-sets. The grand total number of graphs generated was 3,918. The feature titled "BWHDRL" was chosen as the only feature used, because it has been used in the past to achieve 99% accuracy (Callegari, 2024). Unlike the rootkit dataset, the CICIDS graph datasets are comprised of 95% nonevent and 5% event classes. Table 1 summarizes the phase space parameters used for both datasets.

Table 1: Phase space parameters for both datasets

Parameter	Rootkit	CICIDS
Base cases	5	5
Nodes per cut set	1970	1970
Filter width	65	65
Symbols	3	2, 3, 5
Time delay	44	44

3.2 Data Preprocessing

For the rootkit dataset, phase space graphs were obtained in preprocessed form from McDonald et al (2022) using the parameters listed in Table 1. No additional normalization or outlier removal was applied, as the phase space algorithm inherently discretizes continuous values into symbolic bins through its symbolization step, reducing sensitivity to individual outliers. For the CICIDS dataset, the raw BWHDRL feature values were processed through the phase space algorithm at three symbol levels. The symbolization step serves as an implicit normalization, mapping continuous values into discrete ranges. No explicit outlier removal was performed on the raw features prior to phase space conversion, as the binning process accommodates extreme values within the outermost symbol ranges.

3.3 Model Configurations

In order to best compare the GNNs used in this study, each model was configured the same way with as similar sizes as possible. Each model used the GNN as the input layer with six hidden layers of size 256, found via experimentation to be the point where models did not benefit from a size increase after 20 epochs of training. To address the tendency for models to continuously predict only one class, a dropout rate of 25% was set. Past the GNN layer, every model's output was put through a 25% dropout layer, batch normalization, then a linear layer which halved the logits from 256 to 128 before a binary classification head.

3.4 Experimental Design

All datasets were divided into a 70/15/15 split of training, testing, and validation data. All samples were stratified so that each section contained the same class distribution percentages within a margin of 1%. When applicable, datasets were also stratified by trials to ensure models would not learn trial-specific quirks. Each dataset variant was assigned four untrained GNN models (GraphSAGE, EdgeCNN, GCN, and GIN), each trained over 20 epochs.

For all experiments, the models were tasked with binary classification, where class zero refers to benign data and class one refers to malicious or event data. The evaluation metric selected was balanced accuracy, the average of recall obtained on each class. This metric prevents class imbalances from affecting results; if a model consistently predicts only one class, balanced accuracy for a two-class problem equals 50%, no better than guessing.

4. Results

4.1 Rootkit Detection Experiment

With the data split into a standard 70/15/15 split for training, testing, and validation, and the data stratified to ensure all data from a particular recording session stayed within the same group, all four of the GNN models were trained for 20 epochs, the point after which no significant model improvement occurred. The balanced accuracy of the models against the validation data (weighted equal to class distribution) was recorded after each

epoch (without the ability to learn from the validation data) and plotted for final analysis. Figure 2 presents these results.

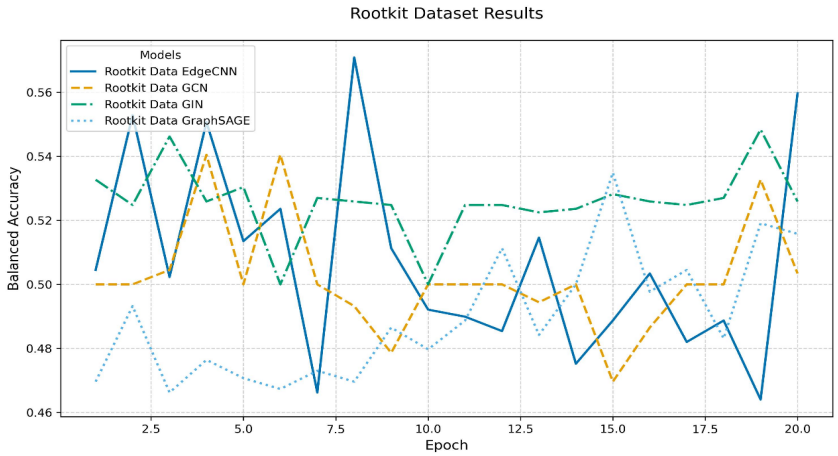

Figure 2: Model performance on rootkit dataset showing balanced accuracy over 20 training epochs

Although EdgeCNN regularly fell below other models, it did have the final highest balanced accuracy by a substantial amount. All other models had highly variable performances over time, indicating a tendency to overfit or underfit. The GIN model, while lower in the end, had the most consistent increase in model performance. This would be the recommended choice for consistent results on this dataset. EdgeCNN's sporadic per-epoch performance implies that the model continuously under-fit or over-fit the data and was sensitive to the noise within the data. It is for this reason that EdgeCNN would likely be more performant on a larger dataset and would not likely be the first choice for training with smaller datasets.

4.2 Network Intrusion Detection Experiment

Sparse graphs (2 symbols): All models tended to be biased in the sparse dataset and heavily affected by the 95% and 5% class dispersion. The flat line at 50% means that all models were obtaining 100% accuracy with class A and 0% accuracy with class B which resulted in a flat straight line for all models. This caused every model to consistently make predictions for class A. This is relatively expected; due to the sparseness of the graph there remains little room for variation or ability to distinguish different graphs and the highly complex space that they represent. It is likely that no learning is occurring and that class B, or malicious data, lacked any unique characteristics at this granularity of analysis.

Medium graphs (3 symbols): In the "medium" or nominally populated graph with three symbols, we see more necessary complexity, which appears to provide information for the models to somewhat discriminate and classify the two different classes, with a substantial increase in both activity and average accuracy for all models. The GIN model has the most consistent performance on this dataset variant, with the GCN overtaking it at times but with more drastic changes in accuracy across several epochs. The GraphSAGE model was still quite prone to over-fitting, resulting in large drastic spikes upward and downward on the graph. The GCN provided more stable performance than GraphSAGE however it also failed to surpass GIN in any given epoch barring one exception with epoch 12.

Dense graphs (5 symbols): Similar to the increase from sparse to medium density graphs, the dense graphs with five symbols consistently showed better final results for most models. The GIN model obtained the highest average accuracy at approximately 65%, with the EdgeCNN slightly behind with a score just under 61%. The GCN and GraphSAGE models still seemed to be affected by over-fitting and went back to single-class predictions multiple times during the experiment with final results well below the performance of GIN. Figure 3 presents the dense dataset results, and Table 2 shows the final confusion matrix outcomes.

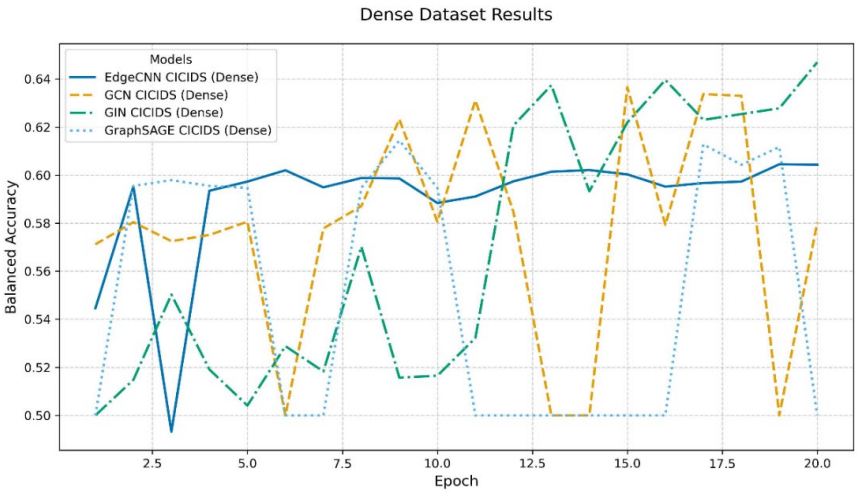

Figure 3: Graph neural network performance on dense CICIDS dataset (5 symbols)

Table 2: Confusion matrix results for dense CICIDS dataset

Model	True Neg	False Neg	False Pos	True Pos	Bal. Acc
EdgeCNN	355	137	25	73	0.641
GCN	301	124	79	86	0.601
GIN	370	149	10	61	0.632
GraphSAGE	380	210	0	0	0.500

As evident in the graph above, the GIN network took more epochs to learn this more complex data. However, the GIN still eventually reaches the top level of accuracy and stabilizes unlike most other models. The EdgeCNN model was able to get a slightly lower accuracy with moderate consistency as well, despite its erratic performance on the previous variant of the dataset. GraphSAGE's consistent 50% balanced accuracy with zero true positives indicates complete failure to identify malicious samples in the dense variant.

4.3 CNN Baseline Comparison Experiment

To provide a direct comparison between GNN-based classification and the CNN image-conversion approach described in Section 2.5, a baseline experiment was conducted using a CNN on phase space graph data. The ABkdSS-single rootkit dataset, consisting of 480 system-call phase space graphs split evenly between benign and malicious classes, was used. Each graph's adjacency matrix was converted into a grayscale image representation and fed into a standard CNN architecture for binary classification. The dataset was organized into eight equal folders for stratified evaluation.

The CNN achieved 93% training accuracy but only 59.4% test accuracy, exhibiting a training-test gap of over 33 percentage points, as summarized in Table 3. Per-folder test accuracy ranged from 37.5% to 87.5%, indicating high variance across data partitions. This substantial overfitting pattern contrasts with the GNN results, where models such as GIN maintained more consistent performance between training and evaluation. The CNN's overfitting can be attributed to the information loss inherent in converting graph adjacency matrices to rasterized images: the flattening process discards structural relationships between nodes, forcing the CNN to learn from pixel patterns that may not faithfully represent the underlying graph topology.

Table 3: CNN baseline results on ABkdSS-single rootkit dataset

Metric	Value
Training accuracy	93.0%
Test accuracy	59.4%
Training-test gap	33.6 pp

Metric	Value
Per-folder range	37.5% - 87.5%
Samples	480

5. Discussion

5.1 GNN Architecture Performance

Based on the result of the experiments, there is evidence to suggest that GNNs perform better on phase space data when the symbols parameter is a higher value. The models consistently reached a level that is better than random, implying that GNNs and phase space can be used in conjunction with one another and could evolve into a new way of analyzing time-series cybersecurity data. In all experiments, GIN and GraphSAGE never fell below 50% accuracy. In the majority of experiments, all models reached better than random performance before the end of training.

Based on the data, there is evidence to suggest that GIN offers the best performance consistently on all of the variants of the datasets tested. While EdgeCNN can reach comparable results on specific instances, and outperform GIN at certain epochs, the GIN eventually reaches a higher accuracy than the other models. GIN's consistent performance can be attributed to its theoretical properties: its sum-aggregation function is injective, meaning it maps distinct multisets of neighbor features to distinct representations (Xu et al, 2019). This is critical for phase space graphs where subtle differences in state transition patterns distinguish benign from malicious behavior. By contrast, GCN uses mean-aggregation, which can conflate structurally different neighborhoods when they share similar average feature values, which may explain GCN's more erratic performance on these datasets.

EdgeCNN eventually won out on the rootkit dataset, however its per-epoch performance was sporadic, implying sensitivity to noise. EdgeCNN would likely be more performant on a larger dataset.

GraphSAGE demonstrated poor ability to distinguish between classes on the CICIDS dataset. This may be attributed to its sampling-based aggregation, which subsamples neighbors during training rather than considering all connections. For phase space graphs where discriminative patterns depend on the complete neighborhood structure, this subsampling likely discards critical transition information.

5.2 Impact of Graph Density

A key finding is the strong relationship between phase space symbol count and classification performance. The complete failure of all models on sparse graphs (2 symbols) indicates insufficient discriminative information at low granularities. The progressive improvement with medium (3 symbols) and dense (5 symbols) graphs suggests higher symbol counts capture finer state distinctions necessary for differentiating benign from malicious behavior.

With the hindsight of what was learned from the CICIDS experiments, the rootkit data experiments seem to be hindered by the low symbol count parameter from the phase space portion of the preprocessing. Due to the direct correlation between this symbol variable and the resulting sparseness of the graph, this variable was modulated to see its impact on GNN performance with cybersecurity data, to decrease the probability of having a flawed embedding technique. It was infeasible during this research to explore the full parameter space for phase graph generation, and thus predetermined parameter sets from prior work were used.

5.3 Comparison with Existing Methods

The CNN baseline experiment (Section 4.3) provides a direct empirical comparison. On the rootkit dataset, the CNN achieved 59.4% test accuracy despite 93% training accuracy, a 33.6 percentage point overfitting gap. By comparison, GIN on the rootkit dataset maintained more consistent training-to-validation performance with less severe overfitting. The CNN's high variance across data partitions (37.5% to 87.5% per-folder accuracy) further illustrates the instability of the image-conversion approach, where the flattening of adjacency matrices into pixel grids discards the structural relationships that distinguish benign from malicious graphs.

Callegari (2024) reported 99% accuracy using CNNs on image-converted CICIDS phase space graphs, though direct comparison requires caution as that work used different experimental configurations and multi-feature approaches. The GNN approach offers practical advantages by eliminating the graph-to-image conversion step, preserving the original graph topology, and avoiding the isomorphism challenge where structurally equivalent

graphs may produce different image representations. These results, combined with the CNN baseline comparison, establish that GNNs provide a more structurally faithful approach to phase space graph classification, with GIN identified as the most promising architecture.

5.4 Computational Considerations

All models converged within 20 epochs, with per-epoch training times on the order of seconds for the rootkit dataset and minutes for the CICIDS variants. GIN and GCN trained faster per epoch than EdgeCNN and GraphSAGE. GNN inference is computationally lightweight once trained, though the upstream phase space conversion step introduces latency that must be considered in deployment.

5.5 Limitations

Due to the nature of these experiments, conclusive rules cannot be stated about model performances, however this work has shown evidence that can be leveraged to find more optimal parameters for the phase space algorithm in conjunction with GNNs. Several limitations should be acknowledged. First, only a limited set of symbol values was explored, and optimal parameters may vary across datasets. Second, generalizability to other domains has not been established. Third, the study did not include formal statistical significance testing; future work should incorporate cross-validation with paired statistical tests. Finally, the GNN models can only operate on one singular graph, but the phase space algorithm maps each feature to a separate graph. Methods to combine graphs exist but fall beyond the scope of this work.

6. Conclusion

This study demonstrates that GNNs provide a viable approach for analyzing phase space graphs in cybersecurity applications. There is evidence to support the claim of GNNs being consistently better than random on various datasets when combined with phase space preprocessing. GIN emerged as the most consistently effective architecture, attributed to its injective sum-aggregation function that preserves structural distinctions critical for phase space classification. Graph density significantly impacts classification performance, with higher symbol counts in phase space preprocessing yielding improved results.

While the absolute accuracy levels achieved are modest compared to CNN-based approaches, GNNs offer the advantage of operating directly on graph structures without information-losing conversion steps. Future research should explore optimal symbol parameters across broader ranges, methods for combining multiple features in multi-variate datasets, incorporation of statistical validation frameworks, and validation on additional cybersecurity domains beyond rootkit detection and network intrusion.

Ethics Declaration: This research used publicly available datasets and did not require ethical clearance.

AI Declaration: AI tools were used to assist with manuscript preparation, citation conversions, and formatting. The research methodology, experiments, analysis, and conclusions are entirely the work of the authors.

References

Bollobas, B. and Riordan, O. (2010) 'Sparse graphs: metrics and random models', arXiv:0812.2656.

Bouritsas, G., Frasca, F., Zafeiriou, S. and Bronstein, M.M. (2023) 'Improving graph neural network expressivity via subgraph isomorphism counting', IEEE Transactions on Pattern Analysis and Machine Intelligence, 45(1), pp. 657-668.

Callegari, C. (2024) Network Intrusion Detection Using Nonlinear Phase Space Analysis, Doctoral dissertation. Available at: https://www.proquest.com/docview/3126842334

Deng, A. and Hooi, B. (2021) 'Graph neural network-based anomaly detection in multivariate time series', Proceedings of the AAAI Conference on Artificial Intelligence, 35(5), pp. 4027-4035.

Errica, F., Podda, M., Bacciu, D. and Micheli, A. (2022) 'A fair comparison of graph neural networks for graph classification', arXiv:1912.09893.

Fu, Y. et al (2024) 'Remote sensing time series analysis: A review of data and applications', Journal of Remote Sensing, 4, Article 0285.

Hamilton, W.L., Ying, R. and Leskovec, J. (2018) 'Inductive representation learning on large graphs', arXiv:1706.02216.

Jadon, S., Milczek, J.K. and Patankar, A. (2021) 'Challenges and approaches to time-series forecasting in data center telemetry: A survey', arXiv:2101.04224.

Jin, M. et al (2024) 'A survey on graph neural networks for time series: Forecasting, classification, imputation, and anomaly detection', IEEE Transactions on Pattern Analysis and Machine Intelligence, 46(12), pp. 10466-10485.

Kim, S., Park, J., Lee, K., You, I. and Yim, K. (2012) 'A brief survey on rootkit techniques in malicious codes', Journal of Internet Services and Information Security, 2(3/4), pp. 134-147.

Kyte, I., Zavarsky, P., Lindskog, D. and Ruhl, R. (2012) 'Enhanced side-channel analysis method to detect hardware virtualization based rootkits', Proceedings of the 2012 World Congress on Internet Security, Las Vegas, pp. 192-201.

Lobo, D., Watters, P., Wu, X.W. and Sun, L. (2010) 'Windows rootkits: Attacks and countermeasures', Proceedings of the 2010 Cybercrime and Trustworthy Computing Workshop, Ballarat, pp. 69-78.

Luckett, P., Pavelescu, E., McDonald, T., Hively, L. and Ochoa, J. (2019a) 'Predicting state transitions in brain dynamics through spectral difference of phase-space graphs', Journal of Computational Neuroscience, 46(1), pp. 91-106.

Luckett, P., Watts, T., McDonald, J.T., Hively, L. and Benton, R. (2019b) 'A deep learning approach to phase-space analysis for seizure detection', Proceedings of the 10th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics, Niagara Falls, pp. 190-196.

McDonald, J.T., Clark, R.C., Hively, L.M. and Russ, S.H. (2022) 'Phase space power analysis for PC-based rootkit detection', Proceedings of the ACM Southeast Conference, New York, pp. 82-90.

Neupane, S. et al (2022) 'Explainable intrusion detection systems (X-IDS): A survey of current methods, challenges, and opportunities', IEEE Access, 10, pp. 112392-112415.

Orlik, M. (2012) Self-Organization in Electrochemical Systems I: General Principles of Self-organization. Temporal Instabilities, Springer, Berlin.

Protopopescu, V.A. and Hively, L.M. (2005) 'Phase-space dissimilarity measures of nonlinear dynamics: Industrial and biomedical applications', Recent Research Developments in Physics, 6(2), pp. 649-688.

Robinson, J.C. (2005) 'A topological delay embedding theorem for infinite-dimensional dynamical systems', Nonlinearity, 18(5), pp. 2135-2143.

Stuckey, T.D. et al (2018) 'Cardiac phase space tomography: A novel method of assessing coronary artery disease utilizing machine learning', PLOS ONE, 13(8), e0198603.

University of New Brunswick (2024) CICIDS2017 Dataset. Available at: https://service.tib.eu/ldmservice/dataset/cicids2017-dataset

Xu, K., Hu, W., Leskovec, J. and Jegelka, S. (2019) 'How powerful are graph neural networks?', arXiv:1810.00826.

Yang, S., Gong, Z., Ye, K., Wei, Y., Huang, Z. and Huang, Z. (2019) 'EdgeCNN: Convolutional neural network classification model with small inputs for edge computing', arXiv:1909.13522.

Zhang, S., Tong, H., Xu, J. and Maciejewski, R. (2019) 'Graph convolutional networks: A comprehensive review', Computational Social Networks, 6(1), Article 11.

Zhao, B., Lu, H., Chen, S., Liu, J. and Wu, D. (2017) 'Convolutional neural networks for time series classification', Journal of Systems Engineering and Electronics, 28(1), pp. 162-169.