Optimizing Honeypot Placement Strategies with Graph Neural Networks for Enhanced Resilience via Cyber Deception

Mohamed Osman
Virginia Commonwealth University
Richmond, VA, USA
osmanmw@vcu.edu

Tamer Nadeem
Virginia Commonwealth University
Richmond, VA, USA
tnadeem@vcu.edu

Ahmed Hemida
DEVCOM Army Research Laboratory
Adelphi, MD, USA
ahmed.h.hemida.ctr@army.mil

Charles Kamhoua
DEVCOM Army Research Laboratory
Adelphi, MD, USA
charles.a.kamhoua.civ@army.mil

ABSTRACT

In the ever-evolving realm of cybersecurity, strategic honeypot placement is pivotal for enhanced network deception. This paper introduces a novel approach, leveraging Graph Neural Networks (GNNs), to optimize honeypot placements, outstripping traditional game-theoretic methods. Traditional techniques often face computational inefficiencies due to the "curse of dimensionality" in complex dynamic environments. Our GNN model, through extensive exploration of architectures including Graph Attention Transformers (GAT) and Graph Transformer models, showcases superior performance. With the integration of game-theoretic edge features, the model achieves a remarkable test accuracy of 92.34%. Additionally, our GNN solution provides a 139x inference speedup over classical methods, underpinning its efficiency and potential to revolutionize cybersecurity strategies.

CCS CONCEPTS

• **Security and privacy** → **Network security**; **Mobile and wireless security**; • **Computing methodologies** → **Machine learning**.

KEYWORDS

Honeypot Placement, Network Deception, Graph Neural Networks, Cyber Security, Graph Attention Transformers, Optimization Strategies.

ACM Reference Format: Mohamed Osman, Tamer Nadeem, Ahmed Hemida, & Charles Kamhoua. 2023. Optimizing Honeypot Placement Strategies with Graph Neural Networks for Enhanced Resilience via Cyber Deception. In Proceedings of the 2nd Graph Neural Networking Workshop 2023 (GNNet '23), Dec. 8, 2023, Paris, France. ACM, NY, NY, USA, 7 pages. https://doi.org/10.1145/3630049.3630169

GNNet '23, December 8, 2023, Paris, France

ACM ISBN 979-8-4007-0448-2/23/12.
https://doi.org/10.1145/3630049.3630169

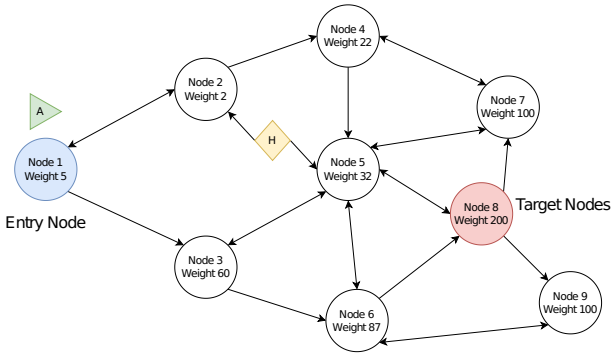

Figure 1: An example scenario with an attacker starting out at the entry node (in blue). A honeypot (H) may be placed along any of the edges to capture the attacker as it tries to reach one of the target nodes (in red). We restrict our scenario to only one honeypot, and attempt to optimize its placement.

1 INTRODUCTION

Modern computer networks have become highly connected and heterogeneous due to the integration of a diverse array of devices and protocols, including Internet of Things (IoT), robots, sensors, and other connected systems. This growing complexity and heterogeneity, although essential for providing sophisticated services and adapting to rapidly changing demands, also pose significant security challenges. Networks now encompass an increasing number of devices with different operating systems, making them more susceptible to interference, cyber attacks, and complex management issues such as patching vulnerabilities. These security concerns become even more critical in military environments, where the Internet of Battlefield Things (IoBT)[11, 14] necessitates the protection of critical nodes and system components.

To counter cyber threats, network administrators, or defenders, often employ cyber deception techniques during the reconnaissance stage of an attack.[24] During this stage, attackers gather information about targeted systems and networks to develop their attack strategies. Cyber deception techniques involve manipulating the network interfaces to disguise the true state of the network, thereby disrupting the attacker's decision-making process. Honeypots, which are decoy systems designed to attract and engage attackers, play a crucial role in enhancing network resilience by

intercepting attacker paths. However, traditional game-theoretic approaches for honeypot placement exhibit limitations and inefficiencies in dynamic environments with multiple attackers and honeypots. Additionally, these methods often rely on computationally intensive iterative algorithms, such as fictitious play, rendering them slow and less practical for real-time applications.

To illustrate our approach, we consider a scenario depicted in Fig. 1. In this specific setting, the network is exposed to a single attacker, entering through a designated entry node. Our primary objective is to ascertain the most strategic edge within this network on which to deploy a honeypot. By doing so, we aim to maximize our chances of intercepting and neutralizing the attacker, thereby safeguarding the network's integrity.

To address these challenges, this study proposes a Graph Neural Network[25] (GNN)-based model for determining the optimal placement of honeypots in a given network topology. Our approach leverages the power of GNNs to process graph-structured network data and generate honeypot placement strategies that maximize the probability of intercepting attacker paths. We investigate the application of various GNN architectures, including Graph Attention Transformers (GAT) variants[6, 29], Graph Convolutional Networks (GCN)[13], and Graph Transformer models[8], to identify the most suitable architecture for the problem at hand.

Furthermore, we explore a variety of input features, such as node values as the only node features, binary features for entry and target nodes, and reward matrix data derived from classical game-theoretic algorithms. We also implemented with different feature embedding approaches. This exploration of input features aims to assess their impact on model performance and identify the most effective feature set for the GNN-based approach.

In addition, our research examines different optimization strategies, such as learning rate curves (one cycle [27], cosine annealing warm restarts [17], ReduceLROnPlateau[1]), optimizers [7, 12, 16, 31], model ema [10], as well as various model sizes, and model architectures incorporating the aforementioned layers tested. This investigation into optimization strategies and model configurations seeks to determine the most effective combination for enhancing the GNN-based model's performance in identifying optimal honeypot placements.

Our research ultimately aims to contribute to the broader goal of utilizing AI capabilities to improve network security and resilience against cyber threats. By developing a more efficient and adaptable solution compared to classical game-theory-based methods for optimal honeypot placement, we strive to enhance network deception and minimize the number of successful attack trials, thereby safeguarding networks and systems against cyber threats.

The rest of the paper is organized as follows. We discuss the related work in Section 2. Then we discuss the game theory model and the attack graph in detail in section 3. In Section 4, we provide the system model and provide the problem statement and optimization model. In Section 5, we provide simulation/experimental results regarding the performance of proposed solution in various scenarios. Finally, we conclude and discuss future work in Sections 6 and 7.

[1]as implemented in PyTorch

2 RELATED WORK

Over the last decade, there has been a remarkable growth in the deployment of Deep Neural Networks (DNNs) across various fields. These networks have shown significant promise in areas such as image recognition [15], speech recognition [1], natural language processing [21], self-driven network protocols [23], synthesizing network traffic [20], among others. Notably, their potential in network security and privacy, especially in tasks like anomaly detection and intrusion detection systems, has been well-established [3, 9, 19, 32]. However, the vulnerability of these networks to adversarial attacks, especially those targeting resources, remains a pressing issue [4].

In the context of this paper, we broadly classify the most related work into two categories: (1) honeypot placement strategies based on game theory, and (2) the application of graph-based neural networks for various tasks. In this section, we review the most relevant literature in both areas and highlight the key differences between our proposed approach and the existing methods.

2.1 Game Theory-based Honeypot Placement Strategies

Game theory has been widely used to model and analyze the interactions between attackers and defenders in cybersecurity, including honeypot placement[2]. In a work by Anwar et al.[2], the authors propose a scalable algorithm for allocating honeypots over an attack graph. They model the problem as a two-person zero-sum strategic game between a network defender and an attacker. The game formulation captures the importance of different nodes in the network, as well as the cost associated with various defense strategies and the cost incurred by the attacker. Furthermore, it considers a practical threat model concerning the available information about the attacker to the network defender. Nash equilibrium defense strategies are analytically characterized, and the complexity of a general game is discussed. The authors propose a scalable algorithm to obtain honeypot allocation strategies in large-scale networks, which we use as a basis for our work.

However, as mentioned earlier, traditional game-theoretic approaches for honeypot placement exhibit limitations and inefficiencies in dynamic environments with multiple attackers and honeypots. Moreover, these methods often rely on computationally intensive iterative algorithms, such as fictitious play, rendering them slow and less practical for real-time applications.

2.2 Graph-based Neural Networks

Graph-based neural networks, including Graph Convolutional Networks (GCN)[13], Graph Attention Transformers (GAT) variants[6, 29], and Graph Transformer models[8], have shown great potential in processing graph-structured data and tackling various tasks, such as node classification, link prediction, and graph classification. The GNNs have been widely employed in numerous application domains, including social networks, recommender systems, and drug discovery[25].

In our work, we leverage the power of GNNs to process graph-structured network data and generate honeypot placement strategies that maximize the probability of intercepting attacker paths. We investigate the application of various GNN architectures to

identify the most suitable architecture for the problem at hand. Furthermore, we explore different input features, optimization strategies, and model configurations to optimize model performance in determining optimal honeypot placements.

To the best of our knowledge, our work is the first to propose a GNN-based model for optimizing honeypot placement strategies in network deception and resilience against cyber attacks. By developing a more efficient and adaptable solution compared to classical game-theory-based methods, we strive to enhance network deception and minimize the number of successful attack trials, thereby safeguarding networks and systems against cyber threats.

In summary, our research combines the strengths of both game theory-based honeypot placement strategies and graph-based neural networks to address the limitations and inefficiencies of traditional methods. By incorporating the reward matrices from game theory approaches as edge features in our GNN-based model, we aim to develop an innovative and effective solution for determining the optimal placement of honeypots in network topologies.

3 ATTACK GRAPH & GAME THEORY MODEL

In this section, we define the attack graph and describe the formulation of the game theory model.

3.1 Attack Graph

Attack Graphs (AGs) are widely used in cybersecurity to model potential attacks by mapping out all possible scenarios [22]. These graphs are typically constructed based on network topologies, vulnerabilities, etc [30]. Consequently, an AG can be interpreted differently depending on the specific scenario at hand. In this context, we examine an attack graph composed of N nodes, represented as the graph $G(V, E)$, where $N = |V|$. Each node denotes a vulnerability linked to a host or machine within the network. The presence of an edge $e_{u,v} \in E$ connecting nodes u and v implies the ability to exploit a vulnerability at node v through one at node u. Nodes within the graph are assigned values denoted as w_v, reflecting their significance to the network administrator. Nodes with higher values represent valuable assets within the network, containing critical information and databases essential to the tactical network. It is reasonable to assume that these nodes, due to their elevated values, are particularly attractive to potential adversaries and network attackers. These attackers aim to maximize their expected rewards by strategically choosing from the set of all accessible nodes to compromise. In our model, we posit that the attacker possesses knowledge of the values associated with each node. This assumption holds as attackers often gain access to internal information regarding the network structure and employ network scanning tools for probing [18] during the attack reconnaissance stage.

We make the assumption that the defender lacks precise information about the attacker's location, reflecting a practical threat model. Nonetheless, the defender is aware of the potential entry points the attacker might exploit to penetrate the network. Network records allow the defender to establish a distribution $f_a(\cdot)$ over these entry points, where $V_e \subset V$ represents the entry point set. In essence, the defender can ascertain the probability that an attacker might breach the network through an entry point $u \in V_e$ as $f_a(u)$, with the constraint $\sum_{u \in V_e} f_a(u) = 1$. The objective of the defender is to formulate a proactive strategy for the placement of honeypots within a given network, utilizing the constructed attack graph. To this end, we have transposed this problem into a two-player game theoretical framework, the specifics of our game model are described in the subsequent section.

3.2 Game Theory Model

Our game is defined as a triple ($\mathcal{N}$,$\mathcal{A}$, R), presenting the defender and attacker as the set of players $\mathcal{N} = \{1, 2\}$, their respective action spaces $\mathcal{A} = \mathcal{A}_1 \times \mathcal{A}_2$, and the zero-sum nature of their rewards, $R_1 + R_2 = 0$. The defender's prerogative is the judicious allocation of honeypots within the network, a strategic maneuver captured by the meticulously constructed attack graph. The attacker, conversely, is advancing through the network, with each move carrying the intrinsic risk of exposure and associated costs. The defender's strategy is to decide where to place the honeypots or whether to abstain from placing them to avoid costs P_c. The honeypots are placed on the edges modeling fake services and vulnerabilities to mislead the attacker. The attacker's strategy involves deciding which node to attack next while balancing the risk of exposure and the associated attack cost, A_c. The attacker reward matrix, $R_2 = -R_1$.

The reward function can easily be expressed as follows,

$$R_1(a_1, a_2) = \begin{cases} -P_c + A_c + Cap * w_v\,; & a_1 = e_{a,v}, a_2 = v \quad \forall v \in \mathcal{V} \\ -P_c + A_c + Esc * w_u\,; & a_1 = e_{a,v}, a_2 = u \forall u \neq v \in \mathcal{V} \\ -P_c\,; & a_1 = e_{a,v}, a_2 = 0 \quad \forall v \in \mathcal{V} \\ 0\,; & a_1 = 0, a_2 = 0 \end{cases} \tag{1}$$

where Cap and Esc represent the rewards for the defender and attacker, respectively. Cap denotes the defender's capture reward, earned when the attacker exploits an edge where a honeypot is placed. In contrast, Esc denotes the attacker's escape reward, acquired when the attacker successfully exploits an edge where no honeypot is positioned, thus evading capture. Finally, taking zero actions signifies that both the defender and attacker are opting to back off, a decision that occurs when either player is confronted with very high action costs. The reward matrix of the game encapsulates all possible attack and defense scenarios along with their corresponding outcomes. It is possible to determine the Nash equilibrium strategies of the game using standard game-solving methodologies, such as linear programming [2].

However, traditional game-solving techniques like linear programming [2] often grapple with the curse of dimensionality. This challenge arises due to the computational complexity that grows exponentially with increasing network size, the number of honeypots, and the diversity of attack paths. To address these limitations, we explore the potential of Graph Neural Networks (GNNs) as illustrated in fig 2. Specifically:

1. **End-to-End Learning Approach**: This method purely utilizes the graph topology to model and predict the optimal strategies. The goal is to minimize the difference between the predicted probabilities from the GNN and those obtained from the game-solving technique. This is depicted on the right side of fig 2.

2. **Hybrid Approach**: Combining information from both the reward matrix and the graph topology, this approach aims to harness the strengths of both the end-to-end and game theory approaches, as portrayed on the left side of fig 2.

In mathematical terms, given the predicted probabilities P_{GNN} from the GNN and the true probabilities P_{true} from the game-solving technique, our objective is to minimize the cross entropy loss:

$$\mathcal{L} = -\sum_i P_{\text{true}}(i) \log P_{\text{GNN}}(i)$$

4 METHODOLOGY

In this section, we present our methodology for developing a GNN-based model to optimize honeypot placement strategies, aimed at enhancing network deception and resilience against cyber attacks. We discuss the experimental design, input feature engineering, model architecture exploration, and optimization strategy evaluation.

4.1 Experimental Setup

Our experiments were conducted on a fixed graph topology with a predetermined number of nodes. The experiments consisted of one attacker, three target nodes, one honeypot, and one entry node. To introduce variability, the rewards for non-target nodes were randomized. We divided the dataset into training and testing sets to assess the performance of the proposed GNN-based models. All the presented experiments are conducted on a dataset containing 100,000 generated graph configurations with a node count set to 30.

4.2 Input Feature Engineering

4.2.1 Node Features. Node features in our framework capture essential information pertaining to the nodes in the graph. Specifically, each node is characterized by the following attributes:

(1) **Node Value** (v): Represents the primary attribute of the node. This can be a continuous scalar value.
(2) **Entry Node Indicator** (e): A binary indicator that designates whether the node is an entry node. It takes a value of 1 if the node is an entry node, and 0 otherwise.
(3) **Target Node Indicator** (t): A binary vector where the i-th element is set to 1 if the node is the i-th target node, and 0 otherwise.

Given these attributes, the node feature vector $\mathbf{x}_i$ for the i-th node can be represented as:

$$\mathbf{x}_i = \begin{bmatrix} v_i \\ e_i \\ \mathbf{t}_i \end{bmatrix} \tag{2}$$

For nodes with continuous values, we further enrich the representation using Fourier features[28]. Let $f(v)$ denote the Fourier transformation of a scalar v. The transformed node value feature becomes $f(v_i)$. For binary attributes (entry and target indicators), we leverage embedding tables, resulting in vectors $\text{emb}_e(e_i)$ and $\text{emb}_t(\mathbf{t}_i)$, respectively.

Therefore, the enhanced node feature vector $\mathbf{X}_i$ becomes:

$$\mathbf{X}_i = \begin{bmatrix} f(v_i) \\ \text{emb}_e(e_i) \\ \text{emb}_t(\mathbf{t}_i) \end{bmatrix} \tag{3}$$

In our exploratory experiments we've empirically found that at larger data scales the use of embedding tables and fourier features enhances accuracy. We reason that this is because increasing their dimensionality allows our model to allocate more parameters to these inputs.

4.2.2 Edge Features. Edges in our graph are enriched with reward matrix data derived from game-theoretic algorithms. Let $\mathbf{R}_{ij}$ represent the reward matrix data for an edge between nodes i and j. The incorporation of these reward matrices as edge features augments the GNN's capability to understand and make strategic decisions on honeypot placements, seamlessly blending game-theoretic insights with graph-based learning.

4.2.3 Feature Processing and Integration. To prepare the data for the GNN-based model, we process the node and edge features as follows:

(1) **Node Feature Processing**: Each node feature $\mathbf{X}_i$ is passed through a node embedder. The node embedder consists of a Fourier transformation for the node value, and embeddings for binary features, combined linearly to produce the final node representation.
(2) **Edge Feature Processing**: Edge features $\mathbf{R}_{ij}$ are directly transformed into a fixed-size vector using a linear layer, producing the edge representation.

Once processed, these features are fed into the GNN. Within the GNN, node representations are updated through various layers, while the edge features provide additional context for these updates. We show a simplified overview of this method in Fig 2.

4.3 Optimizing and Accelerating the Simulator

We optimized the simulator from [2] by creating a parallelized, just-in-time compiled version of their algorithm for calculating rewards. We observed that in practice, fictitious play was approximately 200 times slower than this optimized algorithm. This observation motivated our approach to use the reward matrix as edge features, transforming the problem into approximating only the fictitious play module instead of both the reward matrix and fictitious play. By optimizing the reward matrix calculation, we were able to significantly reduce the computational overhead. Fig 2 shows a block diagram of a simulator and the two proposed GNN input schemes.

4.4 Model Architecture Exploration

We investigated various GNN architectures to identify the most suitable one for the problem at hand. We considered Graph Attention Transformers (GAT) variants [6, 29], Graph Convolutional Networks (GCN) [13], and Graph Transformer models [8]. We experimented with different model sizes and model architectures that incorporated the aforementioned layers to determine the best-performing configuration for identifying optimal honeypot placements.

To further enhance the performance of the GNN-based model, we examined various optimization strategies, such as learning rate curves (one cycle [27], cosine annealing warm restarts [17], and ReduceLROnPlateau), optimizers [7, 12, 16, 31], and model ema [10].

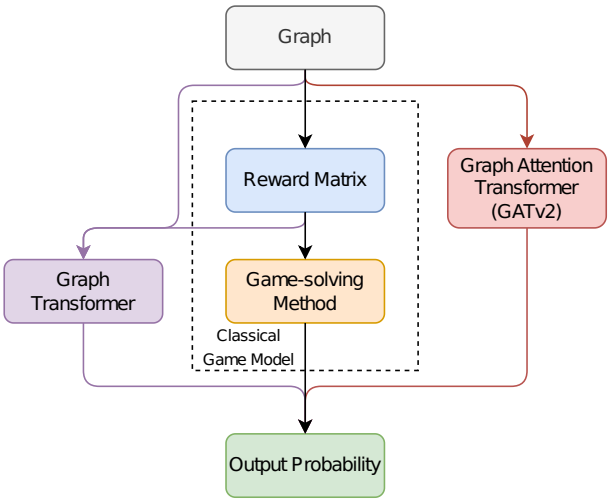

Figure 2: In the middle a block diagram of the simulator is shown. On the left and right, different GNN input configurations are shown.

Our goal was to determine the most effective combination of optimization strategies and model configurations to maximize the GNN-based model's ability to identify optimal honeypot placements.

The best configuration without edge features utilized GATv2 [6] and included skip connections for every layer, layer normalization [5], and the SwiGLU [26] activation function. Edge embeddings were generated by concatenating node embeddings, which were then used for making predictions.

The best configuration using edge features was the edge feature variant of the Graph Transformer [8]. The model implementation followed the architecture presented in the paper. Mean edge features were used to make predictions.

5 EXPERIMENTAL RESULTS

In this section, we present the experimental results and performance evaluation of the proposed GNN-based models for optimal honeypot placement in a network topology. We evaluate our models using two primary metrics: accuracy and cross-entropy loss. Accuracy reflects the proportion of correctly predicted honeypot placements, while cross-entropy loss measures the discrepancy between the predicted probability distribution and the target distribution. Lower cross-entropy loss and higher accuracy indicate better performance.

We perform an 80-20 train-test split and perform no model selection. We train for 100 epochs for all experiments. Further, to show how our technique scales, we perform experiments with the training set size set to 80k, 10k and 2k samples. Test set size is static at 20k samples. All experiments are run on a machine with a 7950X CPU and NVidia RTX 4090 GPU. A batch size of 1024 is used.

5.1 Model Performance Evaluation

We compared the performance of our best performing models with and without edge features. Table 1 summarizes the test accuracy and cross-entropy loss values for these models.

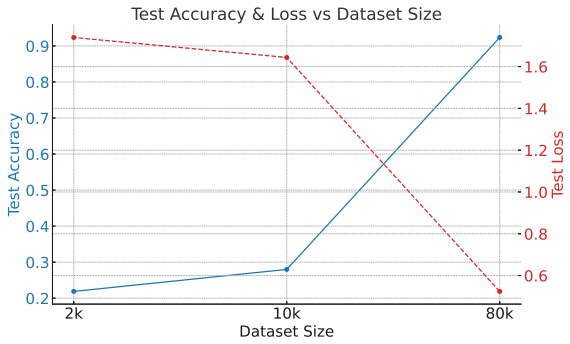

Figure 3: Graph showing accuracy and loss at different data scales.

As shown in Table 1 and Fig 3, our best performing model with edge features, the Graph Transformer, achieved a test accuracy of 92.34% and a cross-entropy loss of 0.5248. In contrast, the best performing model without edge features, GATv2, had a lower test accuracy of 48.47% and a higher cross-entropy loss of 1.737. These results demonstrate the effectiveness of incorporating edge features derived from game-theoretic algorithms into our GNN-based model, leading to a substantial improvement in performance.

5.2 Impact of Input Features and Architectures

Our experiments revealed that extending the input features by adding binary features for entry and target nodes, as well as incorporating reward matrix data as edge features, significantly improved model performance. Furthermore, embedding tables and fourier features empirically led to modest improvements, especially at large data scales.

In terms of model architectures, our exploration indicated that GATv2 with skip connections, layer normalization, and the SwiGLU activation function performed best without edge features. Meanwhile, the edge feature variant of the Graph Transformer achieved the best performance when edge features were included.

5.3 Optimization Strategy Performance

Our evaluation of various optimization strategies revealed that the optimal combination for maximizing the GNN-based model's ability to identify honeypot placements consisted of the following: a one cycle learning rate curve[27], the LAMB optimizer[31], and model ema[10]. This combination proved to be the most effective in enhancing the model's performance.

Table 1: Performance Comparison of GNN-based Models

Model	Test Accuracy	Test Cross-entropy
GATv2 (no edge features)	48.47%	1.737
Graph Transformer (edge features) 80k	92.34%	0.5241
Graph Transformer (edge features) 10k	27.98%	1.6436
Graph Transformer (edge features) 2k	21.88%	1.7392

5.4 Inference Latency and Energy Consumption

For the game-theoretic solution, extensive testing revealed an average inference time of 1250 seconds when analyzing 100,000 samples. It's important to note that this classical approach was executed on a powerful multi-core system consisting of 16 CPU cores and 32 threads (Ryzen 7950X). However, as the number of attackers, honeypots, and nodes increases, the inference time of the game-theoretic solution experiences exponential growth. In contrast, our proposed GNN-based solution exhibited remarkable efficiency, achieving an inference time of approximately 9 seconds when processing the same volume of samples. This performance was measured on a single Nvidia RTX 4090 and it represents a speedup of almost 139x. Importantly, the GNN-based solution demonstrated polynomial scaling, meaning that its inference time will not increase exponentially with the growth of system complexity. This scalability advantage becomes especially pronounced when dealing with a larger number of attackers, honeypots, or network nodes.

To understand the environmental impact and energy efficiency of both solutions, we adopted a measure of efficiency described as "samples processed per second per watt". For the game-theoretic solution running on a CPU with a peak power target (PPT) of 230W and thermal design power (TDP) of 170W, we assumed a worst-case scenario where the CPU is consuming its full PPT of 230W. Under this assumption, the CPU processed 100,000 samples in 1250 seconds, resulting in an efficiency rate of approximately 0.348 samples per second per watt. In contrast, our GNN-based approach on the RTX 4090 GPU, with a power consumption of 450W, processed the same number of samples in just 9 seconds. This gives an impressive efficiency of approximately 24.691 samples per second per watt. The stark difference in these rates underscores the power-efficient nature of our GNN-based approach, positioning it as a more environmentally friendly option, especially in scenarios demanding high computational resources. Lastly, both the multiprocessing version of the simulator and our proposed GNN reported high levels of hardware utilization, meaning that they efficiently use the available resources of the hardware. As we further optimize the system, we expect to approach the theoretical maximum power consumption of the hardware due to increased utilization. This expected trend further justifies our decision to use worst-case power consumption scenarios for our efficiency estimations.

6 DISCUSSION

The GNN-based model proposed in this paper showcases the viability of leveraging advanced machine learning architectures to tackle long-standing problems in cybersecurity. However, like all solutions, the approach presented here comes with inherent challenges and potential avenues for improvement.

Scaling to Larger Network Topologies: Scaling to Larger Network Topologies: A primary constraint of our GNN model arises from its dependency on a specific reward matrix size, which is defined by the simulator we utilize. This results in a model with an input linear layer tailored to match the reward matrix's dimensions. Consequently, any alteration in the network topology inherently changes the reward matrix size, necessitating a corresponding adjustment to the model's input layer. While the core of our model remains invariant to topological changes, its ability to adapt to varied topologies without re-instantiating the input layer poses a limitation. The model exhibits commendable adaptability when weights fluctuate within a singular topology. However, its performance under completely diverse topologies is yet to be explored. Furthermore, although adapting to a different reward matrix size is technically straightforward—by creating a new input linear layer—this approach's practicality in dynamic real-world scenarios remains unproven. We leave further exploration on this critical point to future work.

Multiple Attackers and Honeypots: Another layer of complexity arises when considering scenarios with multiple attackers and honeypots. The dynamic interactions among various attackers, each potentially having different objectives and strategies, will significantly complicate the game-theoretical framework. Incorporating this into a GNN structure would require a more intricate modeling of node and edge features.

Generalization Across Diverse Networks: While the model demonstrated promising results on the fixed graph topology, its performance on different and more diverse network structures remains to be tested. This is crucial, as real-world networks can significantly vary in their configurations.

Future work should also investigate the integration of more advanced reinforcement learning techniques. These can potentially eliminate the need for targets from fictitious play and allow the model to learn optimal honeypot placements through direct interactions with the environment.

7 CONCLUSION

This paper presented an innovative GNN-based approach to optimize honeypot placements within network topologies, achieving promising results when compared to traditional game-theory-based methods.

The inclusion of edge features, derived from game-theoretic algorithms, proved instrumental in amplifying the performance of the GNN-based model. This not only underscores the potency of Graph Neural Networks in processing graph-structured data but also emphasizes the synergy achieved by marrying classical game-theoretic insights with advanced machine learning techniques.

While the initial results are encouraging, this paper also paves the way for future research. Exploring more dynamic environments, testing the robustness of the model across diverse network structures, and integrating advanced learning techniques are just a few of the exciting avenues to be pursued.

This work serves as a stepping stone towards a future where AI capabilities play an instrumental role in fortifying networks against cyber threats, ensuring a safer digital ecosystem for all.

ACKNOWLEDGMENTS

This work was supported in part by the U.S. National Science Foundation under Grant OAC-2212424; and in part by the Commonwealth Cyber Initiative, an investment in the advancement of cyber research and development, innovation, and workforce development.

REFERENCES

[1] Mahbubul Alam, Manar D. Samad, and et. al. 2019. Survey on Deep Neural Networks in Speech and Vision Systems. arXiv:1908.07656 [cs.CV]

[2] Ahmed H Anwar, Charles Kamhoua, and Nandi Leslie. 2020. Honeypot allocation over attack graphs in cyber deception games. In *2020 International Conference on Computing, Networking and Communications (ICNC)*. IEEE, 502–506.

[3] Mohammed Ayyat, Tamer Nadeem, and Bartosz Krawczyk. 2023. Class-Aware Neural Networks for Efficient Intrusion Detection on Edge Devices. In *2023 20th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*. 204–212. https://doi.org/10.1109/SECON58729.2023.10287462

[4] Mohammed Ayyat, Santosh Kumar Nukavarapu, and Tamer Nadeem. 2022. Dynamic Deep Neural Network Adversarial Attacks for Edge-based IoT Devices. In *GLOBECOM 2022 - 2022 IEEE Global Communications Conference*. 61–67. https://doi.org/10.1109/GLOBECOM48099.2022.10001235

[5] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450* (2016).

[6] Shaked Brody, Uri Alon, and Eran Yahav. 2022. How Attentive are Graph Attention Networks?. In *International Conference on Learning Representations*.

[7] Xiangning Chen, Chen Liang, Da Huang, Esteban Real, Kaiyuan Wang, Yao Liu, Hieu Pham, Xuanyi Dong, Thang Luong, Cho-Jui Hsieh, et al. 2023. Symbolic discovery of optimization algorithms. *arXiv preprint arXiv:2302.06675* (2023).

[8] Vijay Prakash Dwivedi and Xavier Bresson. 2020. A generalization of transformer networks to graphs. *arXiv preprint arXiv:2012.09699* (2020).

[9] Mahmoud Said ElSayed, Nhien-An Le-Khac, Marwan Ali Albahar, and Anca Jurcut. 2021. A novel hybrid model for intrusion detection systems in SDNs based on CNN and a new regularization technique. *Journal of Network and Computer Applications* 191 (2021), 103160. https://doi.org/10.1016/j.jnca.2021.103160

[10] Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. 2018. Averaging weights leads to wider optima and better generalization. *arXiv preprint arXiv:1803.05407* (2018).

[11] Charles A Kamhoua. 2018. Game theoretic modeling of cyber deception in the internet of battlefield things. In *2018 56th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE, 862–862.

[12] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[13] Thomas N Kipf and Max Welling. [n. d.]. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations*.

[14] Alexander Kott, Ananthram Swami, and Bruce J West. 2016. The internet of battle things. *Computer* 49, 12 (2016), 70–75.

[15] Zewen Li, Wenjie Yang, Shouheng Peng, and Fan Liu. 2020. A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects. arXiv:2004.02806 [cs.CV]

[16] Ilya Loshchilov and Frank Hutter. [n. d.]. Decoupled Weight Decay Regularization. In *International Conference on Learning Representations*.

[17] Ilya Loshchilov and Frank Hutter. [n. d.]. SGDR: Stochastic Gradient Descent with Warm Restarts. In *International Conference on Learning Representations*.

[18] G. Lyon. 2008. *Nmap Network Scanning: Official Nmap Project Guide to Network Discovery and Security Scanning*. Insecure.Com, LLC. https://books.google.com/books?id=a_PkPQAACAAJ

[19] Santosh Nukavarapu and Tamer Nadeem. 2022. iKnight–Guarding IoT Infrastructure Using Generative Adversarial Networks. *IEEE Access* 10 (2022), 132656–132674. https://doi.org/10.1109/ACCESS.2022.3224583

[20] Santosh Kumar Nukavarapu, Mohammed Ayyat, and Tamer Nadeem. 2022. MirageNet - Towards a GAN-based Framework for Synthetic Network Traffic Generation. In *GLOBECOM 2022 - 2022 IEEE Global Communications Conference*. 3089–3095. https://doi.org/10.1109/GLOBECOM48099.2022.10001494

[21] Daniel W. Otter, Julian R. Medina, and Jugal K. Kalita. 2019. A Survey of the Usages of Deep Learning in Natural Language Processing. arXiv:1807.10854 [cs.CL]

[22] Xinming Ou, Wayne F Boyer, and Miles A McQueen. 2006. A scalable approach to attack graph generation. In *Proceedings of the 13th ACM conference on Computer and communications security*. 336–345.

[23] Hannaneh Barahouei Pasandi and Tamer Nadeem. 2021. Towards a Learning-Based Framework for Self-Driving Design of Networking Protocols. *IEEE Access* 9 (2021), 34829–34844. https://doi.org/10.1109/ACCESS.2021.3061729

[24] Neil C Rowe and Han C Goh. 2007. Thwarting cyber-attack reconnaissance with inconsistency and deception. In *2007 IEEE SMC Information Assurance and Security Workshop*. IEEE, 151–158.

[25] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. 2009. The Graph Neural Network Model. *IEEE Transactions on Neural Networks* 20, 1 (2009), 61–80. https://doi.org/10.1109/TNN.2008.2005605

[26] Noam Shazeer. 2020. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202* (2020).

[27] Leslie N Smith and Nicholay Topin. 2019. Super-convergence: Very fast training of neural networks using large learning rates. In *Artificial intelligence and machine learning for multi-domain operations applications*, Vol. 11006. SPIE, 369–386.

[28] Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. 2020. Fourier features let networks learn high frequency functions in low dimensional domains. *Advances in Neural Information Processing Systems* 33 (2020), 7537–7547.

[29] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *International Conference on Learning Representations*.

[30] Shengwei Yi, Yong Peng, Qi Xiong, Ting Wang, Zhonghua Dai, Haihui Gao, Junfeng Xu, Jiteng Wang, and Lijuan Xu. 2013. Overview on attack graph generation and visualization technology. In *2013 International Conference on Anti-Counterfeiting, Security and Identification (ASID)*. IEEE, 1–6.

[31] Yang You, Jing Li, Sashank Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, Kurt Keutzer, and Cho-Jui Hsieh. [n. d.]. Large Batch Optimization for Deep Learning: Training BERT in 76 minutes. In *International Conference on Learning Representations*.

[32] Xiaoxuan Zhang, Jing Ran, and Jize Mi. 2019. An Intrusion Detection System Based on Convolutional Neural Network for Imbalanced Network Traffic. In *2019 IEEE 7th International Conference on Computer Science and Network Technology (ICCSNT)*. 456–460. https://doi.org/10.1109/ICCSNT47585.2019.8962490