Supplementary Materials for 'Black-Box Robustness Probing of Graph Neural Networks for VLSI Circuit Netlists'

Rupesh Raj Karn, Johann Knechtel Ozgur Sinanoglu
Center for Cyber Security, New York University, Abu Dhabi, UAE.
Email: {rupesh.k, johann, ozgursin}@nyu.edu

I. Preliminaries

A. Black-Box Probing

In MLaaS, users have access only to input–output queries and are unable to inspect internal weights, architecture, or training details [1]. This lack of transparency poses significant challenges, especially in critical applications such as circuit analysis where the reliability of a model is paramount. Black-box probing is necessary for several reasons:

- ▷ *Vulnerability Assessment:* Probing through input–output queries enables us to indirectly estimate the model's local sensitivity and infer areas of potential weakness, such as regions where minor perturbations lead to dramatic changes in output [2].
- ▷ *Informed Model Selection:* With a growing number of ML-as-a-Service providers, users need a systematic way to compare models [3]. Evaluating various robustness metrics (e.g., Jacobian-based sensitivity, prediction margins, local Lipschitz constants) through black-box probing offers a quantitative basis for selecting models that demonstrate superior stability under perturbations.
- ▷ *Guiding Defense Mechanisms:* The insights from black-box probing serve as a guide for designing defenses [4]. For example, a model's sensitivity to specific input features suggests that the model can be regularized, or additional filtering can be introduced during inference.
- ▷ *Operational Assurance:* In industrial applications such as integrated circuit analysis, even small misclassifications can have cascading effects [3]. Black-box probing provides an independent diagnostic tool to ensure that deployed models maintain their robustness in the face of real-world variations or potential adversarial attacks.

B. MLaaS for Circuit Domain

The rise of MLaaS [2] has revolutionized circuit design and analysis, with multiple platforms now offering ML-powered solutions. Notable examples include CLEO (Rose–Hulman Institute's circuit learning repository) [5], Cadence Cerebrus (for ML-enhanced verification) [6], and Synopsys' ML-driven tools (enabling cloud-based power analysis and yield prediction) [7]. These developments highlight MLaaS's growing adoption in the circuit domain. Such examples illustrate the MLaaS field's rapid expansion. While our work does not specifically target the robustness evaluation of these vendor tools, our framework provides a mechanism for black-box GNN robustness assessment and can serve as a valuable diagnostic layer for evaluating future MLaaS services once GNNs and their variants are deployed for circuit analysis tasks.

II. Probing Methodology

A. Trojan Injection

We adopt well-known templates from Trust-Hub https://trust-hub.org/ to embed three types of Trojans: *Countermux*, *FSMor*, and *Andxor*. Each of these Trojans operates with distinct mechanisms to remain stealthy under normal conditions while enabling malicious functionality once triggered.

Countermux: This Trojan leverages a hidden multiplexer (MUX) whose select line activates only under rare trigger conditions. Once activated, the Trojan reroutes signals through a malicious path, thereby altering outputs.

FSMor: This Trojan exploits finite state machines (FSMs) by inserting additional hidden states or transitions that cannot be reached during regular operation. When the hidden state is triggered, the FSM transitions into malicious behavior, such as bypassing security checks or disabling functional logic.

Andxor: This Trojan manipulates the circuit using additional AND/XOR logic gates that remain dormant for most inputs. On rare and specific input patterns, the Trojan logic is enabled, resulting in corrupted computations.

These Trojans are inserted to Verilog netlist files from industry-standard benchmarks including ISCAS'85 and EPFL.

B. Containerize GNN Model

To simulate a real-world MLaaS environment, it is essential that our trained GNN model be deployed as a standalone service that users (D) can access as a black box. To achieve this, we encapsulate the entire GNN model along with its dependencies (including Python scripts, required libraries, and pre-trained weights) into a Docker container [8]. The containerization process involves the following steps:

1) *Docker Image Creation:* We create a `Dockerfile` that specifies the base image (e.g., `python:3.9`), installs the required Python packages (e.g., `dgl`, `sklearn`, etc.), and copies the model code and weight files into the container. The resulting Docker image contains all the components needed for inference.

2) *Deployment on a Public Cloud:* To emulate an MLaaS environment, we deploy the Docker container on a cloud platform namely Amazon Web Services (AWS). Using its service–Amazon Elastic Container Service (ECS) [9], the container is launched and exposed via a RESTful API endpoint. The end user then sends input data (circuit graphs) to this endpoint and receives predictions (Ⓓ without any visibility into the underlying model implementation.
3) *Accessing the Deployed Model:* With the container running on AWS, the user interacts with the model through standard HTTP requests. This abstraction perfectly mirrors the MLaaS paradigm, where clients consume ML functionalities as a third-party service without needing direct access to model internals.

C. Robustness Evaluation Metrics

After repeated HTTPS querying of the deployed GNN container, we have accumulated a substantial set of test input-output pairs, denoted as

$$X_{\text{test}} \in \mathbb{R}^{N_{\text{test}} \times d} \quad \text{and} \quad Y_{\text{pred}} \in \mathbb{R}^{N_{\text{test}} \times c},$$

where d is the dimensionality of the node features (recall that each node $v_i \in V$ has $\mathbf{x}_i \in \mathbb{R}^d$) and c is the number of classes. The following standard metrics are then used in Ⓔ:

a) Jacobian Sensitivity [10]: The Jacobian matrix $\mathbf{J}_i = \frac{\partial f}{\partial \mathbf{x}_i}$ quantifies how changes in input node features (e.g., logic level, timing, or structural attributes) affect output logits. The Frobenius norm:

$$\|\mathbf{J}_i\|_F = \sqrt{\sum_{j=1}^{c}\sum_{k=1}^{d}\left(\frac{\partial f_j(\mathbf{x}_i)}{\partial x_{ik}}\right)^2} \tag{1}$$

captures the overall sensitivity. In circuit graphs, a large $\|\mathbf{J}_i\|_F$ suggests instability in classification when minor changes occur in feature representations of functional blocks or subgraphs.

b) Local Lipschitz Constant [11]: The Lipschitz constant estimates the steepness of the decision surface around $\mathbf{x}_i$, formalized as

$$L_i = \sup_{\|\delta\mathbf{x}\|=1} \|\mathbf{J}_i\,\delta\mathbf{x}\| \approx \|\mathbf{J}_i\|_2 \tag{2}$$

In circuits, this reflects how sharply decision boundaries respond to incremental perturbations in node features—relevant for detecting sensitivity in critical logic paths.

c) Hessian-Based Curvature Measure [12]: The curvature of the model's output around $\mathbf{x}_i$ reveals how second-order interactions influence predictions. The largest eigenvalue of the Hessian:

$$\lambda_{\max}\left(\nabla^2 f(\mathbf{x}_i)\right) \tag{3}$$

indicates how dramatically the model's decision can shift. For circuits, high curvature signifies fragility near decision boundaries associated with key functional gates or timing nodes.

d) Prediction Margin [13]: This measures how confidently a node is classified:

$$M(\mathbf{x}_i) = f_{y_{\text{pred}}}(\mathbf{x}_i) - \max_{j \neq y_{\text{pred}}} f_j(\mathbf{x}_i) \tag{4}$$

Small margins reflect uncertainty—particularly important in circuits where classification ambiguity indicates logic redundancy, metastability, or feature overlap in the training set.

e) Adversarial Robustness Radius [14]: Defined as the smallest input perturbation causing a change in predicted label:

$$\rho(\mathbf{x}_i) = \min\{\|\delta\mathbf{x}\| : f(\mathbf{x}_i + \delta\mathbf{x}) \neq f(\mathbf{x}_i)\} \tag{5}$$

In circuit terms, this can be interpreted as the minimum variation in local subgraph features (e.g., gate delays, net capacitance) that shifts functional classification.

f) Stability Under Input Noise [15]: To assess resistance to stochastic perturbations (e.g., noise in measurement or layout extraction), we define:

$$S(\mathbf{x}_i) = \mathbb{E}_{\nu\sim\mathcal{N}(0,\sigma^2 I)}\left[\|f(\mathbf{x}_i + \nu) - f(\mathbf{x}_i)\|\right] \tag{6}$$

This metric indicates how reliably the GNN preserves output under random noise injection, which frequently occurs in practice during circuit synthesis or physical design.

While these metrics originate from standard robustness literature, their application to circuit GNNs provides domain-specific insights—particularly in identifying unstable sub-circuits, critical feature dimensions, and topology-sensitive vulnerabilities in classification tasks.

D. Implications for Defense Strategies

The comprehensive sensitivity evaluation not only informs the selection of robust GNN models from different MLaaS providers but also directly guides the development of effective defense strategies. In particular, our framework highlights several key areas for targeted improvement:

- ▷ *Model Calibration and Smoothing:* Metrics such as the prediction margin and adversarial robustness radius provide clear indications of a model's output confidence and its resilience against small input perturbations. A low prediction margin or robustness radius suggests that the decision boundaries are overly sensitive. MLaaS providers can address this by implementing training techniques that emphasize smoothness along the decision boundary – for example, through adversarial training, gradient regularization, or model distillation. These methods help to increase the margin and, in turn, the robustness against adversarial manipulation.
- ▷ *Sensitivity-Aware Training:* First-order metrics (Jacobian norm and local Lipschitz constant) and the second-order Hessian curvature capture how rapidly the model's output changes with respect to its input. Incorporating regularizers that penalize large Jacobian norms or limit the Lipschitz constant during training can lead to more gradual response surfaces. As a result, outputs become less volatile in the face of small input perturbations, which is particularly advantageous for critical applications like circuit analysis.
- ▷ *Real-Time Anomaly Detection:* The stability under input noise metric reflects the invariance of model predictions when subjected to random perturbations. In deployment, MLaaS providers can monitor these sensitivity measures in real-time. Sudden deviations from typical stability ranges may signal an adversarial attack or unforeseen input anomalies, prompting additional verification procedures.
- ▷ *Integrated Metric-Based Evaluation:* Since our analysis reveals that no single metric suffices to capture all aspects of robustness, combining first- and second-order sensitivity measures with confidence indicators (prediction margin and robustness radius) enables a richer risk profile. This integrated framework not only helps in selecting the most reliable models but also informs continuous monitoring

strategies. Providers should prioritize models that consistently show high prediction margins and robustness radii alongside low sensitivity metrics.

Overall, by targeting these areas—smoother decision boundaries, controlled sensitivity, and enhanced output confidence—MLaaS providers can significantly improve the resilience of their services. For users, this means that models selected on the basis of these consolidated metrics are more likely to withstand adversarial conditions and deliver stable, reliable performance in real-world circuit analysis scenarios.

REFERENCES

[1] Q. Weng, W. Xiao, Y. Yu, W. Wang, C. Wang, J. He, Y. Li, L. Zhang, W. Lin, and Y. Ding, "{MLaaS} in the wild: Workload analysis and scheduling in {Large-Scale} heterogeneous {GPU} clusters," in *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*, 2022, pp. 945–960.

[2] B. Wu, X. Yuan, S. Wang, Q. Li, M. Xue, and S. Pan, "Securing graph neural networks in mlaas: A comprehensive realization of query-based integrity verification," in *2024 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2024, pp. 2534–2552.

[3] K. M. Patel, "Machine learning as a service (mlaas) selection for iot environments," Ph.D. dissertation, Curtin University, 2024.

[4] A. Abomakhelb, K. A. Jalil, A. G. Buja, A. Alhammadi, and A. M. Alenezi, "A comprehensive review of adversarial attacks and defense strategies in deep neural networks," *Technologies*, vol. 13, no. 5, 2025.

[5] "Cleo – circuits learned by example online," 2023, accessed: 2025-05-21. [Online]. Available: https://www.rose-hulman.edu/cleo/

[6] "Cadence cerebrus – ml for electronic design automation," Cadence Design Systems White Paper, 2019, demonstrates ML-based enhancements for circuit verification and optimization.

[7] "Leveraging machine learning in circuit design," Synopsys Technical Journal, 2018, illustrates ML-driven solutions for power analysis and yield prediction.

[8] D. Merkel *et al.*, "Docker: lightweight linux containers for consistent development and deployment," *Linux j*, vol. 239, no. 2, p. 2, 2014.

[9] P. Acuña, "Amazon ec2 container service," in *Deploying Rails with Docker, Kubernetes and ECS*. Springer, 2016, pp. 69–98.

[10] A. Ezertas and S. Eyi, "Performances of numerical and analytical jacobians in flow and sensitivity analysis," in *19th AIAA Computational Fluid Dynamics*, 2009, p. 4140.

[11] M. Jordan and A. G. Dimakis, "Exactly computing the local lipschitz constant of relu networks," *Advances in Neural Information Processing Systems*, vol. 33, pp. 7344–7353, 2020.

[12] Z. Yao, A. Gholami, Q. Lei, K. Keutzer, and M. W. Mahoney, "Hessian-based analysis of large batch training and robustness to adversaries," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[13] C. Zheng, V. Malbasa, and M. Kezunovic, "Regression tree for stability margin prediction using synchrophasor measurements," *IEEE Transactions on Power Systems*, vol. 28, no. 2, pp. 1978–1987, 2012.

[14] C. Qin, J. Martens, S. Gowal, D. Krishnan, K. Dvijotham, A. Fawzi, S. De, R. Stanforth, and P. Kohli, "Adversarial robustness through local linearization," *Advances in neural information processing systems*, vol. 32, 2019.

[15] X. Li, P. Yang, Y. Gu, X. Zhan, T. Wang, M. Xu, and C. Xu, "Deep active learning with noise stability," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 12, 2024, pp. 13 655–13 663.