Redefining Trust: Assessing Reliability of Machine Learning Algorithms in Intrusion Detection Systems

Hossein Sayadi[1], Zhangying He[1], Tahereh Miari[2], Mehrdad Aliasgari[1]

[1]Department of Computer Engineering and Computer Science, California State University, Long Beach, CA, USA
[2]Center for Information Systems and Technology, Claremont Graduate University, CA, USA

***Abstract*—The performance limitations of conventional software-based Intrusion Detection Systems (IDSs) have paved the way for the emergence of hardware-oriented approaches. These approaches harness the power of Machine Learning (ML) algorithms applied to processors' hardware-related data, thereby enhancing the overall system's security and efficiency. However, ensuring the dependability of ML models' decisions is crucial, yet this aspect has been largely overlooked in previous studies. In this paper, we delve into the reliability of machine learning algorithms within hardware-oriented intrusion detection systems, focusing specifically on malware detection. Our investigation aims to bridge the existing gap by shedding light on the trade-offs between performance vs. reliability and robustness levels exhibited by ML models in intrusion detection systems. We conduct a thorough evaluation of ML algorithms in hardware-oriented IDSs, considering factors such as training data size, number of hardware events used, and internal data separability (malware stealthiness). Additionally, we incorporate an effective model observer module to assess prediction probabilities in real-time; thereby, employing a threshold to determine the ML model's confidence for enhanced reliable intrusion detection.**

***Index Terms*—Cybersecurity, Machine Learning, Intrusion Detection System, Reliability Analysis.**

I. Introduction

Intrusion Detection Systems (IDSs) are critical for safeguarding information systems' security and integrity. Malware detection is a key focus within IDSs, aiming to identify and mitigate malicious software that compromises system security and functionality [1], [2]. With the increasing sophistication of malware variants, effective malware detection has become paramount in IDSs. Traditional software-based techniques, relying on pattern matching, signatures, and heuristics, offer protection against known malware strains but struggle to detect emerging threats efficiently [3], [4]. This limitation has driven the development of hardware-oriented approaches, leveraging dedicated hardware components like Hardware Performance Counter (HPC) registers to monitor performance-related data [5]–[10]. By utilizing Machine Learning (ML) algorithms with low resource requirements, hardware-oriented IDSs aim to enhance system efficiency and overall security [11]. The integration of ML algorithms within these approaches holds promise for augmenting intrusion detection capabilities.

The capability of ML models to make dependable decisions is of utmost importance, as they serve as the front-line defense against constantly evolving cyber-attacks [12], [13]. Failure to address reliability concerns jeopardizes the effectiveness and practicality of IDSs. As outlined below, our comprehensive analysis brings to light significant considerations linked to ML-based IDSs which have been overlooked in prior research.

1) Unprecedented Run-Time Situations: While ML-based intrusion detection systems hold great potential, the ability to predict unprecedented run-time situations correctly is imperative, particularly in safety-critical applications. ML-based IDSs struggle to predict and handle unforeseen run-time scenarios effectively. The reliability of ML models can vary significantly, leading to uncertainties in their performance when faced with unseen, stealthy (low malware vs. benign class separability), and/or out-of-distribution intrusion events [14]–[16] .

2) Handling Misclassified Samples: ML-based IDSs often lack robust mechanisms to address wrongly predicted samples [17]. Finding appropriate solutions to handle these samples and minimize their impact is essential for enhancing system performance [18]–[20]. The work in [12] highlights that deep learning models are often fine-tuned within narrow boundaries, prioritizing certain data they were trained on and neglecting run-time variations. However, ensuring reliable decisions in unprecedented conditions in real-time necessitates offering effective assessments to surpass these limitations.

3) Ensuring Dependability and System-Level Robustness: ML-based intrusion detection systems deal with the challenge of achieving run-time dependability and reliability. While current research primarily focuses on testing models with specific datasets, the reliability of these models in real-world, dynamic environments remains uncertain [21]. Additionally, ensuring system reliability requires designing robust fault tolerance mechanisms to handle unprecedented data uncertainty during run-time [22]–[24]. However, existing fault tolerance solutions mainly address system faults and errors, overlooking the limitations of the ML models themselves [17], [25].

This paper aims to address the critical challenges surrounding the reliability of ML models in hardware-oriented intrusion detection systems. We present an in-depth analysis of factors impacting the reliability of ML models from multiple perspectives. Additionally, we propose a novel reliability observing module that evaluates the prediction confidence of models and effectively handles unpredictable samples through advanced detection and safety protection mechanisms. To this aim, we comprehensively assess the performance and reliability of widely deployed machine learning algorithms in hardware-oriented malware detection, considering crucial parameters such as training data size, number of hardware events, and malware stealthiness. Furthermore, we introduce a calibration method for selective prediction to enhance the overall reliability of ML algorithms for robust and efficient intrusion detec-

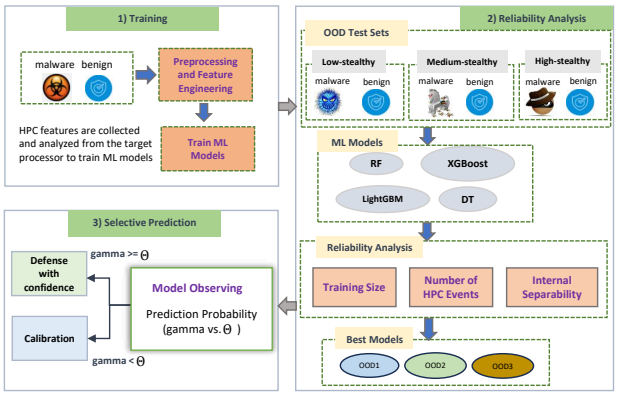

Fig. 1: Proposed framework: Reliability analysis for ML algorithms in IDSs

tion. By achieving a delicate balance between performance and reliability, this research represents a significant advancement toward developing trustworthy and reliable security solutions to effectively counter emerging cyber-attacks.

II. Proposed Reliability Analysis Methodology

A. Data Acquisition and Threat Model

In the data collection phase, we monitored an extensive set of both malware and benign instances, capturing HPC events using the Perf tool on an Intel Xeon X5550 machine [14]. This process involved running applications within controlled Linux Containers (LXC) environment with a sampling time of 10ms [26]. Our dataset consisted of a diverse range of real-world applications, including MiBench [27], SPEC2006 [28], Linux system programs, browsers, and text editors, representing common usage scenarios. For the malware dataset, applications from VirusShare online repository categorized into various types, covering a wide spectrum of real-world malware threats. To organize the data effectively, we partitioned the malware and benign data into four distinct sets and utilized the Mahalanobis Distance [29] metric (detailed in Section II-C) to compute distances among the different malware types. This allowed us to identify closer-distance sets for more effective training. Additionally, we generated three out-of-distribution (OOD) sets comprising zero-day samples not present in the training dataset, enabling thorough evaluation of model performance and generalization on previously unseen data, ensuring reliability and robustness against emerging attack scenarios.

B. Methodology Overview

Figure 1 presents an overview of our proposed methodology for assessing the reliability of ML models in intrusion detection systems and making informed predictions. The evaluation of model reliability is based on our three-fold out-of-distribution test data, allowing us to find the most effective models. Subsequently, the selected reliable model is used for selective prediction.

1) ML Models Implementation: To develop hardware-oriented IDS via reduced HPC data we train twelve ML models with default parameter settings in scikit-learn that have been widely employed in existing research. These algorithms include Random Forest (RF), Decision Tree, Gaussian Naive Bayes (GNB), Logistic Regression (LR), Extra Tree Classifier, Ridge Classifier, K-Nearest Neighbors (KNN), Support Vector Machine (SVM), Bagged Decision Tree, Gradient Boosting, Multi-layer Perceptron (MLP), XGBoost, and LightGBM.

2) Reliability Analysis: Our reliability analysis follows a structured workflow in which we initially explore ML models developed with varying training data sizes, identifying instances of overfitting on out-of-distribution (OOD) test sets. To address this, we partition the training data into thirty independent chunks. Various training data sizes are selected starting from 800 and ending with more than 25K samples. The training dataset contains four types of malware including Trojan, Botnet, Backdoor, and Virus and a set of randomly selected, independent benign samples as the training set. *OOD1* contains three types of malware including Worm, Rootkit, and Ransomware. *OOD2* includes three malware types including Worm, Rootkit, and Adware. *OOD3* contains three types of malware including Worm, Rootkit, and Spyware, and all three sets include the same amount of randomly sampled benign. We trained thirty models for each ML algorithm, which are then evaluated on OOD test sets to determine the model with the highest detection rate (F1-score). Next, we explore the influence of the number of HPC events (1, 2, 3, and 4) on the reliability of ML models. This design parameter is intricately related to the reliability and robustness of the models in hardware-oriented IDSs. Its choice is influenced by the available resources within the target processor architecture, enabling effective run-time detection of attacks [7], [9].

Furthermore, we examine the relationship between the internal separability (stealthiness) of OOD test sets and the models' reliability. The internal data separability is assessed by calculating the Mahalanobis distance between malware and benign samples within each OOD test set. By computing the average distance, we obtain a float number that indicates the level of separability between malware and benign in each set. Finally, we utilize the detection rate of the best ML model on each OOD set as an indicator of reliability score, reflecting the model's dependability during online inference in a given run-time environment. This score serves as an influential factor in determining the prediction probability threshold (gamma) during the calibration process.

3) Model Observer: As shown in Figure 1, we implemented selective prediction to guide predictions into two routes: one with confident defense, and another to reject the prediction by referring it to the next level of further investigation. The best model for each OOD set outputs the predicted class probabilities for each test data, which serves as the model observer. Prediction probability lower than an optimized threshold θ is referred to further investigation while higher than θ is predicted with confidence. For those data that need further investigation, a calibration method such as human-in-the-loop to manually check the incoming data's intrusiveness, or a high performance DNN model (on-device or in the cloud) can be involved. This paper assumes the involvement of humans-in-the-loop capable of accurately distinguishing between malware and benign samples. We then evaluate the overall system performance with the manual calibration and

TABLE I: Internal Mahalanobis Distance for test sets to represent internal data separability (malware stealthiness)

Test Set	Malware	Mahalanobis Distance	Separability
OOD 2	Worm, Rootkit, Adware	0.52	High
OOD 1	Worm, Rootkit, Ransomware	0.40	Medium
OOD 3	Worm, Rootkit, Spyware	0.23	Low

compare it with the original system performance without such calibration strategy in place.

C. Internal Data Separability

In our proposed reliability analysis, we utilize the Mahalanobis distance [29] to model the internal separability of the data. The Mahalanobis distance is a statistical measure commonly employed to assess whether a sample dataset (G1) is drawn from a multivariate normal distribution of another dataset (G2). It quantifies the distance between G1 and G2 by considering the difference between their mean vectors, considering the within-group variation. This metric is computed using the formula shown in Equation 1, where x represents each data point in the observations X to be evaluated, μ denotes the mean vector of the same group, and Σ represents the covariance matrix of the same group. The output of following equation, denoted as D^2 (Mahalanobis distance), provides a metric for assessing the dissimilarity between the two groups.

$$D^2 = (x - \mu)^T \Sigma^{-1} (x - \mu) \quad (1)$$

We quantify the stealthiness (malware benign separability) by calculating the Mahalanobis distance between malware and benign for each OOD test set. We first normalize the features, then use Scikit Learn's $sklearn.covariance.robust\_cov.mahalanobis()$ function [30] to output the Mahalanobis distance matrix for malware and benign. We average all datapoint values to output the average distance score for malware and benign. The internal D^2 is the absolute difference between the average score for all malware and the average score for all benign samples. Table I presents the metrics for the Mahalanobis distance and corresponding malware benign separability level.

III. Evaluation Results and Analysis

In this section, we provide a comprehensive analysis of the results, focusing on the top ML models for each OOD test.

A. Run-Time Data Separability vs. Model Reliability

Figure 2 depicts the seven top-performing ML models performance for different levels of malware and benign separability. Overall, the data/class separability strongly impacts ML models' performance. The detection rates for low separability scenarios (green) are between 0.81 to 0.85, while the high separability (red) achieves over 90% F1-score, with some reaching to 0.95-0.96. XGBoost model performs the best consistently achieving the highest F1-score across all three OOD test sets. For highly stealthy data with low separability, LightGBM, Random Forest, and XGBoost demonstrate comparable performance (0.85). The observations highlight that even the best ML models in IDSs can experience significant performance degradation due to the characteristics and sophistication of the attack. In such cases, trusting the ML model becomes a concern. To address this, conducting iterative run-time reliability tests are suggested to select robust models meeting a predefined threshold and incorporating an option for further investigation if the model's reliability is questionable.

TABLE II: Optimal training data sizes to reach highest performance (F-score) for top-performing ML models

Separatability	RF	LightGBM	XGBoost	DT
High	24,436	22,751	21,065	24,436
Medium	16,853	24,436	12,640	22,751
Low	5,056	24,436	25,279	24,436

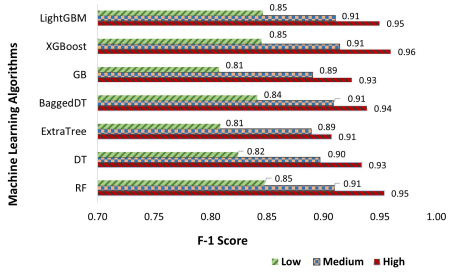

Fig. 2: Top ML models' performance for 3 levels of data internal class separability from low (stealthier) to high

Takeaway 1: *High-performance MLs can reliably detect less-stealthy intrusions (high separability) with distinguishable characteristics but struggle with stealthy intrusions (low separability). Intrusion detection systems must accurately assess the reliability and adapt to the specific attack scenario.*

B. Training Data Size vs. Model Reliability

Table II presents the training data sizes to reach highest performance (F-score) for top MLs for malware detection. For the high-separability OOD test set, Random Forest, LightGBM, and XGBoost achieve F1-scores of 0.955, 0.957, and 0.961, respectively, with corresponding best training data sizes of 24,436, 22,751, and 21,065. Similarly, for the low-separability OOD test set, F1-scores are 0.853, 0.852, and 0.860, with best training sizes of 5,056, 24,436, and 25,279. In addition, Figure 3 highlights the optimal training data sizes that avoid overfitting for medium and low separability data (marked as circled on the graph). As seen, stealthy intrusions require more data to train effective ML models, while less stealthy data tend to overfit with increased data. It is practical to search for the optimal training size that fits ML models based on different intrusion characteristics, as fine-tuning the models in IDSs with the appropriate training size enhances their reliability.

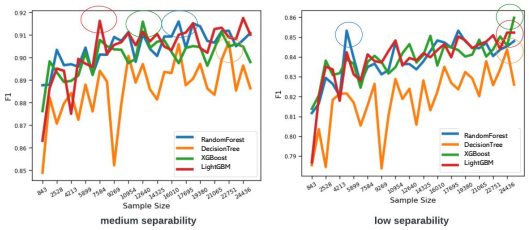

Fig. 3: Training data sizes without over-fitting (circled on the graph)

Takeaway 2: *Stealthy intrusions (low internal data separability) demand a larger amount of training data for effective ML models across all algorithms in IDSs.*

Takeaway 3: *ML algorithms have varying data requirements. XGBoost is data-hungry, while Random Forest needs the least. With ample data, XGBoost performs well with a small model size. In scenarios with moderate data, RF outperforms other algorithms with a medium-sized model.*

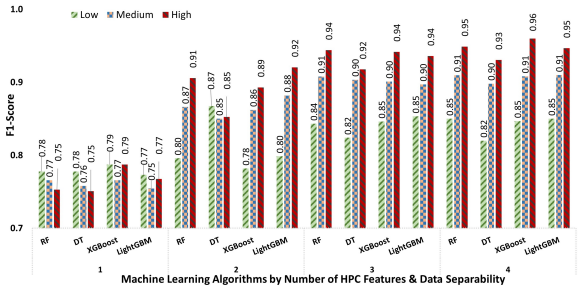

Fig. 4: Number of HPC features impact ML Models' reliability in three levels of separability OOD sets from low to high

C. Number of Hardware Events vs. Model Reliability

In Figure 4, the impact of the number of HPC events used for training on the models' F1-scores is demonstrated across three OOD sets. It is observed that the number of features required for training a stable ML model reaches saturation, typically requiring a maximum of four features in this case. However, this can vary for different datasets and ML algorithms, emphasizing the importance of identifying the optimal number of features for each IDS. The detection rate shows rapid growth when transitioning from one to two events, peaking at four events, with marginal improvement observed when using four events instead of three. This trend holds across various test sets. Using two events can yield a 92% detection rate in high-separability, 88% in medium-separability, and 87% in low-separability sets, that can provide a low-cost solution, especially for resource-constrained applications.

Takeaway 4: *ML algorithms exhibit varying hardware feature requirements to develop reliable IDS based on data characteristics, reaching a saturation point where additional features offer diminishing returns.*

Takeaway 5: *While four events yield highest detection rates, our analysis suggests that a reduced number of events can be more cost-effective. Using just two HPCs slightly decreases performance (approximately 5%), but proves advantageous for resource-constrained platforms such as embedded systems and biomedical devices with limited hardware support.*

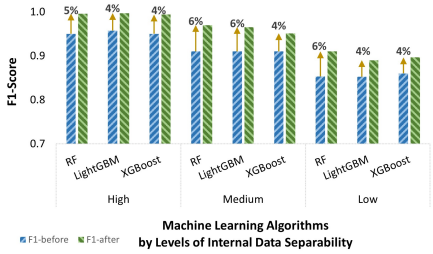

Fig. 5: Detection rate comparison of MLs in IDS before and after calibration

D. Calibration Strategy Analysis

We employ an optimized threshold based on ML models' prediction probabilities to perform selective prediction for calibration. Incoming events during run-time with prediction probabilities exceeding the threshold are confidently predicted by the ML models, while those below the threshold are referred for further investigation. Figure 5 illustrates the F1-scores before and after calibration for the top three ML algorithms (LightGBM, Random Forest, and XGBoost) across three OOD test sets. After calibration, Random Forest obtains optimal performance, with a 100% F1-score in the high-separability set and a 97% score in the medium separability set. For the high stealthy (low internal data separability) OOD test set, it demonstrates a 6% improvement, reaching a 91% detection rate. LightGBM also benefits from calibration using the prediction probability threshold, while XGBoost performs strongly even without calibration.

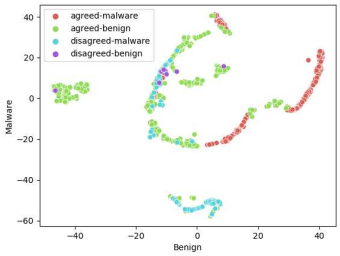

Fig. 6: t-Distribution of observed data: Reliable vs. unreliable predictions by three ML models in hardware IDS for low separability (stealthier) OOD set

To gain insights into the distribution of data in the low-separability OOD test set, we examine the data points correctly and incorrectly predicted by strong ML models (RF, LightGBM, and XGBoost) and their majority vote count. The data points, categorized as agreed and disagreed malware and benign, are visualized using t-SNE visualization method [31] in Figure 6. The plot depicts the challenges posed by stealthy intrusions, particularly for disagreed malware and benign where their data points are intertwined, leading to model unreliability. Future research should emphasize the development of automated techniques within ML-based IDSs to identify and calibrate concealed data points effectively.

Takeaway 6: *ML models exhibit varied responses to calibration methods, resulting in different performance outcomes after calibration. In IDSs, deploying and evaluating calibration methods becomes essential to achieve optimal performance.*

Takeaway 7: *ML models cannot be blindly trusted in intrusion detection systems. They prove to be unreliable in detecting more sophisticated attacks. Incorporating a model observer becomes crucial in assessing the trustworthiness of an ML model and making informed decisions on its reliability.*

IV. Concluding Remarks

In this work, we addressed important challenges concerning the reliability of ML models for intrusion detection using low-level hardware events. We introduced and assessed essential parameters impacting the reliability of ML algorithms in hardware-oriented IDSs, demonstrating their impact on model performance in uncertain run-time conditions. Furthermore, we proposed a lightweight model observer to evaluate the confidence level of model predictions, enabling the selection of effective defenders when the model exhibits high confidence. Conversely, when ML reliability is compromised, alternative options such as human intervention or higher-cost detection methods can be employed. Our analysis demonstrated that the proposed calibration method enhances the overall reliability and performance of the ML-based IDS, resulting in a 4% to 6% improvement in recognizing stealthy malware attacks.

V. Acknowledgment

This work is supported by the National Science Foundation under Award No. 2139034.

REFERENCES

[1] G. Gu, P. A. Porras, V. Yegneswaran, M. W. Fong, and W. Lee, "Bothunter: Detecting malware infection through ids-driven dialog correlation." in *USENIX Security Symposium*, vol. 7, 2007, pp. 1–16.

[2] M. Alazab, K. Soman, P. Poornachandran, A. Al-Nemrat, and S. Venkatraman, "Deep learning approach for intelligent intrusion detection system," *Ieee Access*, vol. 7, pp. 41 525–41 550, 2019.

[3] H. Sayadi, H. Wang, T. Miari, H. M. Makrani, M. Aliasgari, S. Rafatirad, and H. Homayoun, "Recent advancements in microarchitectural security: Review of machine learning countermeasures," in *2020 IEEE 63rd International Midwest Symposium on Circuits and Systems (MWSCAS)*. IEEE, 2020, pp. 949–952.

[4] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, "Survey of intrusion detection systems: techniques, datasets and challenges," *Cybersecurity*, vol. 2, no. 1, pp. 1–22, 2019.

[5] J. Demme, M. Maycock, J. Schmitz, A. Tang, A. Waksman, S. Sethumadhavan, and S. Stolfo, "On the feasibility of online malware detection with performance counters," *ACM SIGARCH Computer Architecture News*, vol. 41, no. 3, p. 559–570, jun 2013.

[6] A. Tang, S. Sethumadhavan, and S. J. Stolfo, "Unsupervised anomaly-based malware detection using hardware features," in *Research in Attacks, Intrusions and Defenses*, A. Stavrou, H. Bos, and G. Portokalidis, Eds. Cham: Springer International Publishing, 2014, pp. 109–129.

[7] H. Sayadi, N. Patel, S. M. P.D., A. Sasan, S. Rafatirad, and H. Homayoun, "Ensemble learning for effective run-time hardware-based malware detection: A comprehensive analysis and classification," in *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*, 2018, pp. 1–6.

[8] B. Singh, D. Evtyushkin, J. Elwell, R. Riley, and I. Cervesato, "On the detection of kernel-level rootkits using hardware performance counters," in *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, ser. ASIA CCS '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 483–493.

[9] H. Sayadi, H. M. Makrani, S. M. P. Dinakarrao, T. Mohsenin, A. Sasan, S. Rafatirad, and H. Homayoun, "2smart: A two-stage machine learning-based approach for run-time specialized hardware-assisted malware detection," *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 728–733, 2019.

[10] M. Ozsoy, C. Donovick, I. Gorelik, N. Abu-Ghazaleh, and D. Ponomarev, "Malware-aware processors: A framework for efficient online malware detection," in *2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA)*, 2015, pp. 651–661.

[11] Z. He, H. M. Makrani, S. Rafatirad, H. Homayoun, and H. Sayadi, "Breakthrough to adaptive and cost-aware hardware-assisted zero-day malware detection: A reinforcement learning-based approach," in *2022 IEEE 40th International Conference on Computer Design (ICCD)*. IEEE, 2022, pp. 231–238.

[12] D. Tran, J. Z. Liu, M. W. Dusenberry, D. Phan, M. Collier, J. Ren, K. Han, Z. Wang, Z. E. Mariet, H. Hu, N. Band, T. G. J. Rudner, Z. Nado, J. van Amersfoort, A. Kirsch, R. Jenatton, N. Thain, E. K. Buchanan, K. P. Murphy, D. Sculley, Y. Gal, Z. Ghahramani, J. Snoek, and B. Lakshminarayanan, "Plex: Towards reliability using pretrained large model extensions," in *First Workshop on Pre-training: Perspectives, Pitfalls, and Paths Forward at ICML 2022*, 2022.

[13] Z. He, T. Miari, H. M. Makrani, M. Aliasgari, H. Homayoun, and H. Sayadi, "When machine learning meets hardware cybersecurity: Delving into accurate zero-day malware detection," in *2021 22nd International Symposium on Quality Electronic Design (ISQED)*. IEEE, 2021, pp. 85–90.

[14] Z. He, A. Rezaei, H. Homayoun, and H. Sayadi, "Deep neural network and transfer learning for accurate hardware-based zero-day malware detection," in *Proceedings of the Great Lakes Symposium on VLSI 2022*, 2022, pp. 27–32.

[15] V. Nagarajan, A. Andreassen, and B. Neyshabur, "Understanding the failure modes of out-of-distribution generalization," *ArXiv*, vol. abs/2010.15775, 2020.

[16] H. Sayadi, Y. Gao, H. Mohammadi Makrani, T. Mohsenin, A. Sasan, S. Rafatirad, J. Lin, and H. Homayoun, "Stealthminer: Specialized time series machine learning for run-time stealthy malware detection based on microarchitectural features," in *Proceedings of the 2020 on Great Lakes Symposium on VLSI*, 2020, pp. 175–180.

[17] L. Myllyaho, M. Raatikainen, T. Männistö, J. K. Nurminen, and T. Mikkonen, "On misbehaviour and fault tolerance in machine learning systems," *Journal of Systems and Software*, vol. 183, p. 111096, 2022.

[18] R. El-Yaniv and Y. Wiener, "On the foundations of noise-free selective classification," *Journal of Machine Learning Research*, vol. 11, pp. 1605–1641, 2010.

[19] Z. Bosnić and I. Kononenko, "An overview of advances in reliability estimation of individual predictions in machine learning," *Intelligent Data Analysis*, vol. 13, no. 2, pp. 385–401, 2009.

[20] Z. Xu and J. H. Saleh, "Machine learning for reliability engineering and safety applications: Review of current status and future opportunities," *Reliability Engineering System Safety*, vol. 211, p. 107530, 2021.

[21] J. M. Zhang, M. Harman, L. Ma, and Y. Liu, "Machine learning testing: Survey, landscapes and horizons," *IEEE Transactions on Software Engineering*, vol. 48, no. 1, p. 1–36, jan 2022.

[22] I. D. Kivlichan, Z. Lin, J. Liu, and L. Vasserman, "Measuring and improving model-moderator collaboration using uncertainty estimation," 2021.

[23] S. X. Wu and W. Banzhaf, "The use of computational intelligence in intrusion detection systems: A review," *Applied Soft Computing*, vol. 10, no. 1, pp. 1–35, 2010.

[24] H. Sayadi, H. Farbeh, A. M. H. Monazzah, and S. G. Miremadi, "A data recomputation approach for reliability improvement of scratchpad memory in embedded systems," in *2014 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*. IEEE, 2014, pp. 228–233.

[25] L. E. Lwakatare, I. Crnkovic, and J. Bosch, "Devops for ai – challenges in development of ai-enabled applications," in *2020 International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, 2020, pp. 1–6.

[26] M. Helsely, "Lxc: Linux container tools," in *IBM developer works technical library*, 2009.

[27] M. R. Guthaus *et al.*, "Mibench: A free, commercially representative embedded benchmark suite," in *IISWC'01*, Dec 2001, pp. 3–14.

[28] J. L. Henning, "Spec cpu2006 benchmark descriptions," *SIGARCH Comput. Archit. News*, vol. 34, no. 4, pp. 1–17, Sep. 2006.

[29] P. C. Mabalanobi, "On tests and measures of group divergenc," *Journal of the Asiatic Society of Bengal*, vol. 26, pp. 541–588, 1930.

[30] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[31] L. van der Maaten and G. Hinton, "Visualizing high-dimensional data using t-sne," *Journal of Machine Learning Research*, vol. 9, no. nov, pp. 2579–2605, 2008.