Adaptive-HMD: Accurate and Cost-Efficient Machine Learning-Driven Malware Detection using Microarchitectural Events

Yifeng Gao*, Hosein Mohammadi Makrani‡, Mehrdad Aliasgari§,
Amin Rezaei§, Jessica Lin*, Houman Homayoun‡, and Hossein Sayadi§

*Department of Computer Science, George Mason University, VA, USA
‡Department of Electrical and Computer Engineering, University of California, Davis, CA, USA
§Department of Computer Engineering and Computer Science, California State University, Long Beach, CA, USA

2021 IEEE 27th International Symposium on On-Line Testing and Robust System Design (IOLTS) | 978-1-6654-3370-9/21/$31.00 ©2021 IEEE | DOI: 10.1109/IOLTS52814.2021.9486701

Abstract*—To address the high complexity and computational overheads of conventional software-based detection techniques, Hardware Malware Detection (HMD) has shown promising results as an alternative anomaly detection solution. HMD methods apply Machine Learning (ML) classifiers on microarchitectural events monitored by built-in Hardware Performance Counter (HPC) registers available in modern microprocessors to recognize the patterns of anomalies (e.g., signatures of malicious applications). Existing hardware malware detection solutions have mainly focused on utilizing standard ML algorithms to detect the existence of malware without considering an adaptive and cost-efficient approach for online malware detection. Our comprehensive analysis across a wide range of malicious software applications and different branches of machine learning algorithms indicates that the type of adopted ML algorithm to detect malicious applications at the hardware level highly correlates with the type of the examined malware, and the ultimate performance evaluation metric (F-measure, robustness, latency, detection rate/cost, etc.) to select the most efficient ML model for distinguishing the target malware from benign program. Therefore, in this work we propose *Adaptive-HMD*, an accurate and cost-efficient machine learning-driven framework for online malware detection using low-level microarchitectural events collected from HPC registers. *Adaptive-HMD* is equipped with a lightweight tree-based decision-making algorithm that accurately selects the most efficient ML model to be used for the inference in online malware detection according to the users' preference and optimal performance vs. cost (hardware overhead and latency) criteria. The experimental results demonstrate that *Adaptive-HMD* achieves up to 94% detection rate (F-measure) while improving the cost-efficiency of ML-based malware detection by more than 5X as compared to existing ensemble-based malware detection methods.

Keywords- Hardware Malware Detection, Machine Learning, Microarchitectural Events, Cost Analysis.

I. Introduction

The fast-growing complexity of modern computing platforms has led to emergence of new security vulnerabilities, making them suitable targets for sophisticated cyber attacks such as malware [1], [2]. Malicious software or malware is the general term for a group of malicious programs developed by cyber attackers to perform harmful tasks like damaging or disabling computer systems, networks, and mobile devices often taking partial control over a digital device's operations or leaking sensitive data and personally identifiable information [3], [4]. According to a recent McAfee Labs threat report more than 67 millions new malware variants have been discovered in only the first quarter of 2019 with nearly 40% increase as compared to the last quarter of 2018 [5].

To cope with with the malware attacks variations and protect the integrity and confidentiality of the authenticated users' information, security researchers have constantly attempted to upgrade the endpoint protection mechanisms (e.g., anti-malware software tools) as the last layer of defense. Such solutions though effective, relied on signature-based or pattern matching analysis categorized as conventional software-based detection techniques that have shown to be inefficient mostly imposing significant complexity and computational overheads on the system. In addition, they can be bypassed using obfuscation or polymorphism methods [2], [6], [7], [8].

To overcome the performance and computational overhead of traditional malware detection techniques, Hardware Malware Detection (HMD) has emerged by employing low-level microarchitectural events of running applications on the target system. These events are collected through Hardware Performance Counters (HPCs) registers [9], [10], [11] that are special-purpose registers implemented in modern microprocessors to capture the hardware-related events of profiled applications. HMD methods have shown the suitability of standard machine learning (ML) algorithms applied on HPCs information in detecting patterns of malicious applications [9], [10], [11], [12], [13], [14]. On the other side, the considerable growth of mobile platforms and Internet-of-Things (IoT) devices development has further intensified the impact of malware threats. In particular, there are some important factors influencing the security vulnerability of embedded systems and IoTs such as limited available resources, low computational capacity, and significant number of computing nodes in the network [8], [15], [16] that need to be considered in the design process of the adopted HMD method in modern platforms.

In this paper, we have addressed major challenges involved with an accurate and cost-efficient microarchitectural malware detection that have been ignored in previous works. In particular, existing HMD techniques have ignored presenting a comprehensive and balanced trade-off analysis between performance (e.g., detection rate) and implementation costs (e.g., hardware overhead) of the ML-based detectors at the processor's hardware level. However, our comprehensive examination across different types of malware and machine learning algorithms used for HPC-based malware detection shows that

978-1-6654-3370-9/21/$31.00 ©2021 IEEE

the type of ML algorithm to classify malicious applications in HMD techniques highly relies on two important factors. First, it depends on the type of malicious applications (e.g., Virus, Worm, Rootkit, etc.) and varies across various malware classes. Second, it correlates with the ultimate performance criteria (F-measure, Area Under the Curve, latency, performance/cost, etc.) to select the most efficient ML model for the inference in online malware detection and classification according to the users' preference and optimal performance vs. cost (hardware overhead and latency).

In response, in this work we propose *Adaptive-HMD*, an accurate and cost-efficient machine learning-driven framework for online malware detection using microarchitectural events collected from HPC registers. *Adaptive-HMD* is equipped with an intelligent and lightweight tree-based decision-maker that accurately selects the most efficient ML model to be used for the inference in online HMD according to the adjusted tuning parameters (e.g., malware class, evaluation metric, available resources, etc.). To the best of our knowledge, this is the first work that addresses the challenge of performance vs. cost-efficiency trade-off analysis for ML-based online malware detection using hardware performance counter features. As we will show in this work, given the high implementation cost of complex but accurate ML models (e.g., neural networks) for detecting malware at the hardware level, a cost-sensitive selection model is required that accounts for impact of both detection performance and implementation cost of the base ML models and determines the best malware detector. The results of this work will assist the designers in making effective architectural decisions to develop accurate and cost-effective intelligent countermeasures for securing modern computing systems against emerging cyber attacks, especially in edge platforms and resource-constrained devices.

The remainder of this paper is organized as follows. Section II presents the motivations and challenges of proposing hardware-based online malware detection using ML techniques. Next, the details of the proposed Ml-driven malware detection approach is described in Section III. Further, the comprehensive evaluation of experimental results is presented in section IV. Lastly, Section V concludes this study.

II. Motivations

This section highlights the motivations of proposing an adaptive hardware malware detection framework using illustrative case studies.

A. Importance of Efficient ML Model Selection at Run-Time

To demonstrate the importance of proposing an efficient and cost-sensitive model selection for online hardware malware detection, we examined five different machine learning models including Sequential minimal optimization (SMO), Multi-Layer Perceptron (MLP), One Rule (OneR), and Decision Tree (RepTree and JRip) algorithms in the task of detecting 5,164 contaminated HPC interval samples. Figure 1 shows the pairwise performance comparison between these five ML classifiers. The values in i_{th} rows and j_{th} column indicate the percentage of tested samples that can be successfully identified by the i_{th} model but cannot be correctly recognized by j_{th} model. For instance, according to the value in the 2_{nd} row and 1_{st} column, in around 14.65% of tested HPC intervals, MLP-based classifier correctly recognizes samples that are misclassified by SMO-based malware detector (malware or benign). In addition, according to the figure, there is no perfect ML model for effective HMD. Some samples that could be easily identified by a simple ML model such as OneR are still surprisingly difficult to be identified by more advanced machine learning models. This highlights the necessity of presenting a cost-sensitive decision-making model to build an accurate and robust malware detection that analyzes all base ML models and attempts to select the most efficient ML classifier for recognizing the malicious patterns at run-time based on user's preference and available underlying resources.

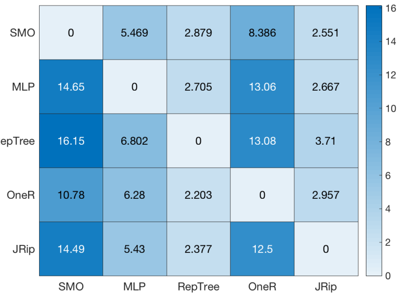


Fig. 1: Performance comparison of different ML models for hardware malware detection

B. Can Ensemble of the ML Models Help?

Ensemble learning is a well-known technique for its ability to greatly enhance classification performance by integrating results from multiple weak classifiers [17], [18]. However, as we show in this work directly applying the ensemble approach by combining multiple ML classifiers for hardware malware detection task incurs significant overhead in terms of latency, area, and complexity to the system. For instance, suppose that we deploy five ML models shown in Figure 1 to build an ensemble classifier to boost up the performance of HMD. As a result, for each input sample the algorithm will need to run all five base classifiers to detect malware. The computation cost of the ensemble-based malware detection is clearly much greater than using a single machine learning model.

As a case study to highlight this issue, Figure 2 illustrates the performance vs. cost (e.g., latency) of three different ensemble methods (Majority Voting, Average Voting, and Stacking) compared with the proposed method, *Adaptive-HMD*. The x-axis and y-axis in Figure 2 represent the model latency (ns) and detection performance rate (F-measure), respectively. As seen, ensemble methods typically deliver large costs in terms of latency with different levels of F-measure as the detection rate metric. For instance, although Stacking method delivers more than 95% detection rate, it has a large latency which makes it less practical for efficient hardware malware detection especially in resource-limited computing systems. Therefore, the objective of the proposed work is achieving a higher or similar performance level as ensemble methods while dramatically reducing the associated overhead and computation cost of hardware malware detection.

III. Overview of *Adaptive-HMD*

In this section, we describe the proposed machine learning-driven approach for accurate and cost-efficient hardware mal-

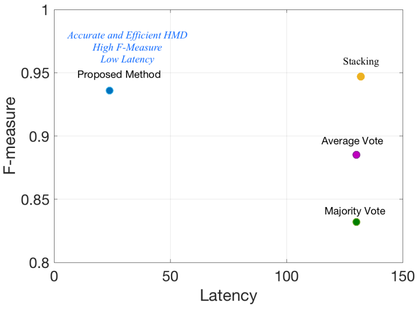


Fig. 2: Performance vs. Cost (e.g., latency) trade-off analysis of ensemble algorithms and the proposed method

ware detection. Figure 3 depicts an overview of the proposed *Adaptive-HMD* framework. As seen, *Adaptive-HMD* consists of two major steps including offline analysis (data collection process) and online detection (model selection and malware detection process) that will be discussed in details below.

A. *Microarchitectural Events Monitoring and Analysis*

The offline analysis step of *Adaptive-HMD* represents the data collection process in which a wide range of benign and malware applications are executed on an Intel Xeon X5550 system running Ubuntu 14.04 with Linux 4.4 Kernel. The target processor hosts 4 HPC registers to profile the applications. Various benign and malware applications are executed. The benign applications include real-world workloads comprising MiBench [19] and SPEC2006 [20], Linux system programs, browsers, and text editors, and malware applications are collected from VirusTotal and VirusShare online repositories including Worm, Virus, Trojan, and Backdoor classes. The microarchitectural events are collected using *Perf* tool at sampling rate of 10ms and are monitored by profiling applications via Linux Containers (LXC) [21]. LXC is considered as an isolated environment which unlike common virtual platforms such as VMWare or VirtualBox, offers access to actual hardware performance counters data instead of emulating HPCs.

Determining the prominent microarchitectural events is an important step for developing accurate and efficient ML classifiers [11], [22], [23]. To effectively select the top HPC features, as shown in Figure 3 we first adopt Correlation Attribute Evaluation to calculate Pearson correlation between each event and target class and rank all the monitored events. Next, we apply Principle Component Analysis (PCA) to find the best HPCs suited for training the ML-based malware detectors. PCA is a class of dimensionally reduction techniques that captures most of the data variation by rotating the original data to a new variable in a new dimension [24], [25]. We use PCA to reduce the features and apply a hierarchical clustering technique to group similar features and identified the top 4 HPCs to capture the behavior of specific class of malware. The feature reduction results indicate that the top 4 HPC events are the same across various classes of malware that include branch instructions, cache references, branch misses, and node-stores.

B. *Adaptive-HMD Training and Testing*

Here, we present the details of training and testing of *Adaptive-HMD* for effective online malware detection using microarchitectural events. As illustrated in Figure 3, the online malware detection process in *Adaptive-HMD* consists of a unified machine learning-based selection model ($model_s$) and N ML-based anomaly detectors (A_i: $i = 0, 1, ..., N-1$). The selection model primarily aims at finding the most accurate and efficient ML-based model from the set of base ML-based anomaly detectors given the observed HPC features for detecting the corresponding class of malware at runtime. In addition, each ML detector A_i in *Adaptive-HMD* is implemented as a binary machine learning classifier to identify whether the observed HPC record is malware or benign.

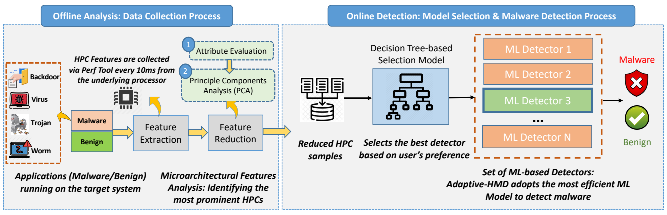

Offline Analysis: Data Collection Process
Online Detection: Model Selection & Malware Detection Process
Backdoor
Virus
Trojan
Worm
Malware
Benign
HPC Features are collected via Perf Tool every 10ms from the underlying processor
1
Attribute Evaluation
2
Principle Components Analysis (PCA)
Feature Extraction
Feature Reduction
Applications (Malware/Benign) running on the target system
Microarchitectural Features Analysis: Identifying the most prominent HPCs
Decision Tree-based Selection Model
Reduced HPC samples
Selects the best detector based on user's preference
ML Detector 1
ML Detector 2
ML Detector 3
...
ML Detector N
Malware
Benign
Set of ML-based Detectors: Adaptive-HMD adopts the most efficient ML Model to detect malware

Fig. 3: Overview of *Adaptive-HMD* for online malware detection

Algorithm 1 *Adaptive-HMD* Training Process

```
Input: Training HPC Interval Samples $H = h_1 \dots h_N$, Label $L = l_1 \dots l_N$. Set of untrained based model $\{A_i | i = 1 \dots M\}$, untrained selection model $model'_s$
Output: Selection Model $model_s$ and based anomaly detectors set $\{A'_i | i = 1 \dots M\}$
for $i \leftarrow 1$ to $M$ do
    $A'_i$ ←FittingModel($H$,$L$ , $A_i$)
    $lat_i$ ←EvaluateLatency($A_i$)
    $area_i$ ←EvaluateAreaOverhead($A_i$)
    for $j \leftarrow 1$ to $N$ do
        $p_{i,j}$ ←CorrectPredictProb($A'_i, h_j$)
    end for
end for
$s \leftarrow score(p, Cost_{lat}, Cost_{area})$
$B$ ←ArgmaxOverI($s$)
$model'_s$=FittingModel($H$,$B$,$model_s$)
return: $model'_s$ and $\{A_i | i = 1 \dots M\}$
```

The training process of *Adaptive-HMD* is described in Algorithm 1 that includes two phrases. Given an input training data with associated labels (malware or benign), the framework first trains each individual base ML-based detector separately. These base detectors include different algorithms including SMO, MLP, OneR, RepTree, and JRip that are selected from diverse branches of machine learning (regression, neural network, decision tree, and rule-based) which are inclusive to model both linear and nonlinear problems and their prediction model can be a binary classifier that is consistent with the malware detection problem.

Next, the algorithm records all prediction results generated from base ML-based detectors (denotes p where $p_{i,j}$ represents the confidence that A_i base classifier can correctly classify j_{th} input sample), as well as the associated latency and area overhead ($Cost_{lat}$ and $Cost_{area}$, respectively). In the second phrase, *Adaptive-HMD* trains a lightweight decision tree-based selection model to identify the best detector. To this aim, *Adaptive-HMD* adopts a JRip classifier, an efficient and accurate decision tree-based classifier, as the selection model. Utilizing JRip minimizes the latency and area overheads caused by selection step in the proposed framework. As we show in this work, *Adaptive-HMD* can obtain better or similar performance as compared to the best base specialized ML-based malware detector. It further enables the user to adjust the cost of the detection mechanism while maintaining a high detection performance rate. Concretely, a criteria score s is evaluated based on the ML-based detector output and the associated cost, per sample for each detector. For each sample, the best detector is determined by the one that has the largest s value (stored in B). Then, the selection model is trained based on the best detector label B and the input samples H. The score function $score(.)$ in *Adaptive-HMD* is defined by:

$$score(h_j.A_i) = P(gt|h_j, A_i) + \alpha C(Cost_{lat}, Cost_{area}) \quad (1)$$

where $P(gt|h_j, A_i)$ indicates the probability that the type of incoming data h_j can be correctly predicted by A_i classifier, and $Cost_{lat}, Cost_{area}$ denotes the latency and area overhead of A_i classifier. In addition, the α parameter is used to determine the overall trade-off between performance and cost of the deployed ML model. When α is large, the model will tend to select a low-cost model algorithm and vice versa. The $score$ function enables *Adaptive-HMD* to adaptively select the most accurate and cost-efficient base ML-based malware detector at run-time according to user's preferences and target evaluation criteria.

Given than the cost has different scale as compared with $P(gt|h_j, A_i)$, before combining the penalty of cost with model, we use a Softmax [26] function to normalize the cost into same value range as $p(gt|h_j, A_i)$ as follows:

$$C(Cost_{lat}, Cost_{area}) = \frac{\exp^{\beta penalty(Cost_{lat}, Cost_{area})}}{\sum \exp^{\beta penalty(Cost_{lat}, Cost_{area})}} \quad (2)$$

in which $penalty(lat_i, area_i)$ which is a weighted sum of latency and area overhead cost is defined as:

$$penalty(Cost_{lat}, Cost_{area}) = \gamma Cost_{lat} + (1 - \gamma) Cost_{area} \quad (3)$$

where γ models user's preference and cost of the detection mechanism (e.g., latency and area overhead). Furthermore, β is a hyper-parameter to adjust the penalty for a high cost model. Concretely, choosing a small β makes the penalty generated from a high cost ML model significantly larger than the low cost model. If β is a large value, $C(.)$ tends to have a similar value across all models. Therefore, the difference of $C(.)$ value between high cost and low cost models is negligible.

Algorithm 2 *Adaptive-HMD* Testing Process

```
Input: Testing HPC Interval Sample $h$, Set of based model $\{A_i | i = 1 \ldots M\}$, selection model $model'_s$
Output: Predicted label $l$
$b \leftarrow model'_s$.predict($h$)
$l \leftarrow A'_b$.predict($h$)
return: $l$
```

Algorithm 2 presents the testing process of *Adaptive-HMD*. Intuitively, different from the training process, only one ML-based detector (most efficient) is adopted to distinguish malware from benign class. In particular, given a newly observed sample h, *Adaptive-HMD* first calls the selection model $model'_s$ to predict which ML model in the base model set A is the best classifier for predicting the label. The selected base malware detector A'_b is used to identify whether o is contaminated. In this way, *Adaptive-HMD* does not require to obtain results of *all* the base detectors while offering an accurate and efficient result, which can significantly reduce the cost of the model. In traditional HMD methods where there is no adaptive selection model, since we need to use model's output to make decisions, the total cost by computing all models during the online detection process is $Cost = \sum Cost_i \quad (i = 1 \ldots M)$. Whereas, in the proposed framework, by adding a selection model to adaptively select the best detector, *Adaptive-HMD* avoids computing the latency and area overhead of all base malware detectors in A and it only considers the cost of the best ML classifier for online detection process, hence, achieving a significant cost improvement.

IV. Experimental Results and Evaluation

In this section, we analyze the experimental results of *Adaptive-HMD* and examine the effectiveness of the proposed framework across various evaluation metrics. All the ML-based detectors and the selection model are implemented through Weka data mining toolbox [27].

A. Performance Evaluation Metrics

Evaluating the performance of machine learning classifiers is an important step in implementing effective ML-based countermeasures. The standard evaluation metrics used for performance analysis of *Adaptive-HMD* are listed in Table I. For analyzing the detection rate, malicious application samples are considered as positive instances. Hence, the True Positive Rate (TPR) represents the proportion of correctly identified positive instances, or malicious samples. The True Negative Rate (TNR) also evaluates the specificity that measures the proportion of correctly identified benign files or negative samples. In addition, the False Positive Rate (FPR) is the rate of benign files that are wrongly classified as malware.

The F-measure (F-score) in ML is interpreted as a weighted average of the precision (p) and recall (r). The precision is the proportion of the sum of true positives versus the sum of positive instances and the recall is the proportion of instances that are predicted positive of all the instances that actually belong to the positive class. The Detection Accuracy further measures the rate of the correctly classified positive and negative samples. Area Under the Curve (AUC) is another important evaluation metric for checking any ML model's performance at various threshold settings that corresponds to the area under the Receiver Operating Characteristic (ROC) graph. The AUC metric shows how well a classification model is capable of distinguishing between different classes.

Finally, running ML classifiers at the processor hardware-level requires a comprehensive evaluation of both the performance and computation costs of the detectors. Therefore, to concurrently account for the impact of detection rate and cost-efficiency of the detectors, in this work we propose to examine the performance and computation cost via the detection rate per cost (Efficiency Score) formulated as below:

TABLE I: Evaluation metrics for performance of ML-based detection

Evaluation Metric	Description
True Positive (TP)	Correct positive prediction
False Positive (FP)	Incorrect positive prediction
True Negative (TN)	Correct negative prediction
False Negative (FN)	Incorrect negative prediction
Precision	$P = TP/(FP + TP)$
Recall: True Positive Rate	$TPR = TP/(TP + FN)$
F-measure (F-score)	$Fmeasure = 2 \times (P \times R)/(P + R)$
Detection Accuracy	$ACC = (TP + TN)/(TP + FP + TN + FN)$
Area Under the Curve	$AUC = \int_0^1 TPR(x)dx = \int_0^1 P(A > \tau(x))dx$

$$Efficiency = \frac{FMeasure \times AUC}{Cost_{type}} \tag{4}$$

where $type \in \{latency, area\}$, and AUC and $FMeasure$ are computed via Equations described in Table I.

B. Hardware Implementations of the ML-based Detectors

Application of machine learning for developing efficient hardware-based security countermeasures requires a thorough analysis of detection rate (accuracy, F-measure, etc.) of an algorithm as well as design area and response time (latency) overhead of ML classifiers. In other words, accounting for the impact of both detection rate and implementation cost of the models is a critical evaluation process in selecting the most accurate and cost-efficient detectors. Therefore, we develop the experimented ML-based malware detectors at the hardware level and analyze their associated area and latency overheads. To this aim, we leverage Vivado HLS compiler to develop the HDL implementation of the classifiers on Xilinx Virtex 7 FPGA. This analysis not only helps us to have a realization of the complexity of the proposed framework in terms of a number of logic gates (which can be similarly proportional to an ASIC implementation) but also assists in analyzing the design implementation overhead and cost in a heterogeneous FPGA+CPU architecture which is emerging in SoC designs.

Table II, reports the hardware implementation results of base ML classifiers in *Adaptive-HMD*. Given that malware detection latency depends on the underlying processor frequency that executes the detection model, the latency is considered as the number of clock cycles (cycles @10 ns). Moreover, the area overhead is calculated using the total number of utilized LUTs, FFs, and DSP units in the FPGA. As seen from Table II, as expected the MLP classifier has led to a significant area and latency overhead, as compared to other ML classifiers.

TABLE II: Hardware implementation cost of base ML-based anomaly detectors used in *Adaptive-HMD*

ML Model	Latency (Cycles @10 ns)	Area (LUTs + FFs + DSPs)
SGD	22	2466
MLP	102	25667
RepTree	3	377
OneR	1	292
Jrip	2	156

As mentioned before, the selection model in *Adaptive-HMD* is a JRip classifier, and all five ML models listed in Table II are used as base malware detectors. *Adaptive-HMD*'s cost consists of two parts including the cost of the selection model and the cost of the selected base model. Since *Adaptive-HMD* only calls one model for malware detection, we evaluate the average running cost of *Adaptive-HMD* by taking latency and area overheads into consideration using the following equation:

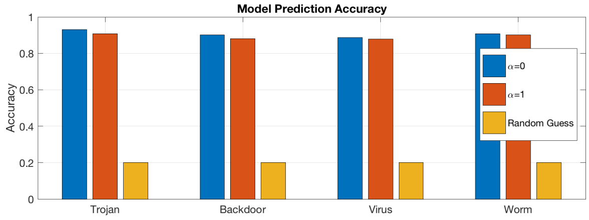


Fig. 4: Selection model prediction accuracy in *Adaptive-HMD*

$$Cost_{type} = \sum_{model \in \mathcal{M}} \frac{N_{model}}{N} Cost_{type,model} + Cost_{type,s} \tag{5}$$

where N denotes the total number of HPC interval samples and $model \in \mathcal{M} = \{SGD, MLP, RepTree, OneR, JRip\}$. N_{model} denotes the total number of times that a model is chosen by the selection model. $Cost_{type,model}$ and $Cost_{type,s}$ define the latency/area overhead of base classifier and the cost of selection model, respectively. Similarly, $area_{model}$ and $area_s$ denote the area overhead for the base model and selection model, respectively.

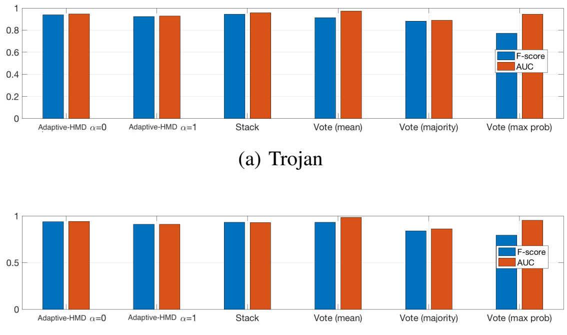
C. Selection Model Accuracy in Adaptive-HMD

Figure 4 illustrates the prediction accuracy of model selection in *Adaptive-HMD*. We examine the effectiveness of *Adaptive-HMD* across all four different types of malware. The performance is evaluated by the accuracy of the proposed model to correctly choose the best base model. Given that there exists no prior HMD work that has focused on the proposed problem, here we compare *Adaptive-HMD* against random guess technique to demonstrate the nontrivial effectiveness and improvement of our proposed HMD. According to the figure, both *Adaptive-HMD* with $\alpha = 0$ and $\alpha = 1$ can achieve approximately 90% accuracy in all four types of malware detection tasks whereas random guess can only achieve 20% accuracy. The result highlights the effectiveness of *Adaptive-HMD* in correctly selecting the best ML model to be used for accurate and efficient detection of anomalies based on the input HPC features at run-time.

D. Performance and Cost Analysis in Adaptive-HMD

To further evaluate the performance of *Adaptive-HMD*, we compare it with classical ensemble machine learning models that are also deployed in recent HMD works such as [13]. Ensemble learning model in machine learning primarily aims at combining multiple base classifiers to improve the classification accuracy [17], [28]. We compare the four different baselines including Stacking [29], Majority Vote, Max Probability Vote, and Vote Average-based Ensemble model [28]. For the purpose of thorough and fair comparison, all algorithms adopt the same set of based models used in *Adaptive-HMD*.

Figure 5 demonstrates the performance results of *Adaptive-HMD* on malware detection under F-score and AUC metrics as compared with ensemble-based detectors. Due to space limitation, in this figure we only report the results for Trojan and Worm detection. It is observed that *Adaptive-HMD* achieves

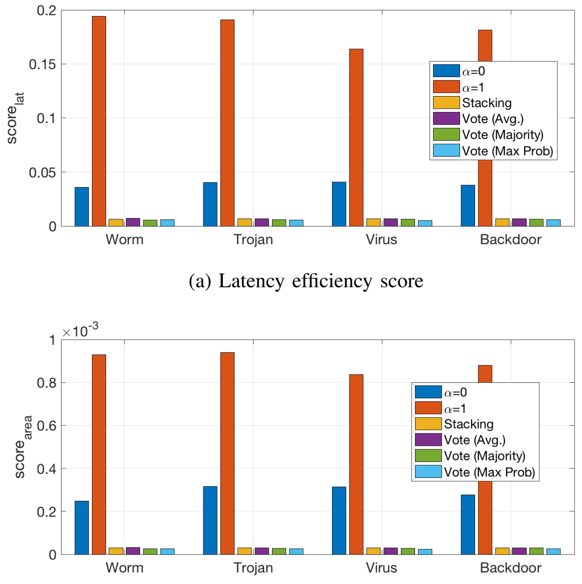


(a) Trojan

(b) Worm

Fig. 5: F-measure and AUC results of *Adaptive-HMD* in detecting Trojan and Worm as compared with ensemble-based detectors

similar performance compared with the ensemble models. The results indicate that compared with the three Voting-based ensemble approaches, *Adaptive-HMD* achieves a similar or better detection performance across all types of malware tested under F-measure, while observing slightly performance degradation under AUC evaluation. Furthermore, the efficiency score analysis of *Adaptive-HMD* with two best α scores against all baseline models is shown in Figure 6. As seen, by taking the implementation cost into consideration, *Adaptive-HMD* achieves a significant improvement as compared with all baseline models. In addition, compared with *Adaptive-HMD* with $\alpha = 0$ in which there is no cost penalty, $\alpha = 1$ achieves a better efficiency score. This is because *Adaptive-HMD* with $\alpha = 1$ has the capability to avoid selecting the high cost model to further achieve the best accuracy per cost performance.



(a) Latency efficiency score

(b) Area overhead efficiency score

Fig. 6: Efficiency trade-off analysis for different malware types

We also evaluate the performance of *Adaptive-HMD* with respect to different α values. The performance vs. α comparison results for Trojan and Worm malware detection in *Adaptive-HMD* are reported in Tables II and III. In these two malware detection tasks, $\alpha = 0$ can achieve the best detection rate (AUC, F-score, recall, and precision) but at the cost of significant latency and area overheads. As the α increased, the computation cost of *Adaptive-HMD* is decreased under the cost of losing accuracy performance. This is because *Adaptive-HMD* tends to strongly avoid using the high cost model when α is large. This selection strategy may reduce some degree of accuracy performance when the penalty is large. Hence, α value highly depends on the user's preference, available resources in the underlying processor, and target evaluation criteria (performance vs. cost). The results indicate that when $\alpha = 1$ *Adaptive-HMD* achieves a significant cost-efficiency improvement while maintaining similar level of detection rate for the best ML model.

TABLE III: *Adaptive-HMD* performance with different α in Trojan

α	Latency	Area	AUC	F-score	recall	precision
0	22.051	2808.579	0.946	0.939	0.950	0.928
1	4.490	911.211	0.927	0.923	0.942	0.905
5	4.151	873.924	0.921	0.918	0.934	0.902
10	4.139	871.692	0.925	0.924	0.942	0.906
100	4.000	876.000	0.853	0.853	0.858	0.848

TABLE IV: *Adaptive-Shield* performance with different α in Worm

α	Latency	Area	AUC	F-score	recall	precision
0	24.645	3541.834	0.941	0.936	0.904	0.970
1	4.274	893.665	0.910	0.911	0.908	0.915
5	4.106	872.932	0.926	0.914	0.892	0.937
10	4.064	871.127	0.906	0.903	0.884	0.924
100	4.000	876.000	0.874	0.877	0.855	0.899

V. Conclusion

According to the recent security analysis reports, malicious software attacks have witnessed an increase at an alarming rate in numbers, variants, and harmful purposes. Hardware Malware Detection (HMD) techniques have presented the suitability of Machine Learning (ML) classifiers applied on applications' low-level features to distinguish malware from benign programs with reduced complexity as compared to traditional software-based methods. Existing HMD solutions have mainly used standard ML algorithms to recognize the patterns of malicious software without offering an adaptive malware detection model to address the trade-off between the detection accuracy and cost-efficiency. In response, in this work we propose *Adaptive-HMD*, an accurate and cost-efficient machine learning-based framework to facilitate online malware detection using low-level microarchitectural features collected from processor's performance counter registers. *Adaptive-HMD* is based on a lightweight and cost-sensitive tree-based decision-maker that accurately determines the most efficient ML classifier to be adopted for the inference in online malware detection according to the users' preference and optimal performance vs. implementation cost. The experimental results indicate that our proposed approach could achieve up to closely 94% detection rate while significantly lowering the implementation cost of hardware malware detection.

References

[1] H. Wang and et al., "Mitigating cache-based side-channel attacks through randomization: A comprehensive system and architecture level

analysis," in *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2020, pp. 1414–1419.

[2] Z. He and et al., "When machine learning meets hardware cybersecurity: Delving into accurate zero-day malware detection," in *2021 22nd International Symposium on Quality Electronic Design (ISQED)*, 2021, pp. 85–90.

[3] A. Bettany and M. Halsey, "What is malware?" in *Windows Virus and Malware Troubleshooting*. Springer, 2017, pp. 1–8.

[4] H. Sayadi and et al., "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.

[5] M. Labs, "Mcafee labs threats report," Aug August 2019.

[6] G. Jacob and et al., "Behavioral detection of malware: from a survey towards an established taxonomy," *Journal in Computer Virology*, vol. 4, no. 3, pp. 251–266, Aug 2008.

[7] A. Souri and R. Hosseini, "A state-of-the-art survey of malware detection approaches using data mining techniques," *Human-centric Computing and Information Sciences*, vol. 8, no. 1, pp. 1–22, 2018.

[8] S. M. P. Dinakarrao and et al., "Cognitive and scalable technique for securing iot networks against malware epidemics," *IEEE Access*, vol. 8, pp. 138 508–138 528, 2020.

[9] J. Demme and et al., "On the feasibility of online malware detection with performance counters," in *International Symposium on Computer Architecture (ISCA)*. ACM, 2013, pp. 559–570.

[10] A. Tang and et al., "Unsupervised anomaly-based malware detection using hardware features," in *International Workshop on Recent Advances in Intrusion Detection (RAID)*. Springer, 2014, pp. 109–129.

[11] H. Sayadi and et al., "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.

[12] B. Singh and et al., "On the detection of kernel-level rootkits using hardware performance counters," in *ASIA Conference on Computer and Communications Security (ASIACCS)*, 2017, pp. 483–493.

[13] K. N. Khasawneh and et al., "Ensemble learning for low-level hardware-supported malware detection," in *International Workshop on Recent Advances in Intrusion Detection (RAID)*, 2015, pp. 3–25.

[14] H. Sayadi and et al., "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.

[15] A. Mosenia and N. K. Jha, "A comprehensive study of security of internet-of-things," *IEEE Transactions on Emerging Topics in Computing*, vol. 5, no. 4, pp. 586–602, Oct 2017.

[16] H. Sayadi and et al., "Customized machine learning-based hardware-assisted malware detection in embedded devices," in *17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, 2018, pp. 1685–1688.

[17] T. G. Dietterich, "Ensemble methods in machine learning," in *Multiple Classifier Systems*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 1–15.

[18] H. M. Makrani and et al., "Energy-aware and machine learning-based resource provisioning of in-memory analytics on cloud," in *Proceedings of the ACM Symposium on Cloud Computing*, 2018, pp. 517–517.

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

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

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

[22] H. Liu and et al., *Feature selection for knowledge discovery and data mining*. Springer Science & Business Media, 2012, vol. 454.

[23] G. Yan and et al., "Exploring discriminatory features for automated malware classification," in *Detection of Intrusions and Malware, and Vulnerability Assessment*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 41–61.

[24] S. Wold and et al., "Principal component analysis," *Chemometrics and Intelligent Laboratory Systems*, vol. 2, no. 1, pp. 37 – 52, 1987, proceedings of the Multivariate Statistical Workshop for Geologists and Geochemists.

[25] A. E. Serpen G., "Host-based misuse intrusion detection using pca feature extraction and knn classification algorithms," in *Intelligent Data Analysis*, 2018.

[26] B. Gao and L. Pavel, "On the properties of the softmax function with application in game theory and reinforcement learning," *arXiv preprint arXiv:1704.00805*, 2017.

[27] M. Hall and et al., "The weka data mining software: An update," vol. 11, no. 1, p. 10–18, 2009.

[28] J. Kittler, "Combining classifiers: A theoretical framework," *Pattern analysis and Applications*, vol. 1, no. 1, pp. 18–27, 1998.

[29] S. Džeroski and B. Ženko, "Is combining classifiers with stacking better than selecting the best one?" *Machine learning*, vol. 54, no. 3, pp. 255–273, 2004.