Enhanced Ransomware Detection Using VMware-Based HPC Feature Extraction and CNN2D Optimization with SHAP Explainability Analysis

Zeba unnisa[1], Md Shoaib Uddin Chanda[2], Mohammed Asim[3], Maimona Jaweed[4]
[1,2,3] *B.E. Students, Department of CSE-AIML, Lords Institute of Engineering and Technology, Hyderabad, India*
[4] *Associate Professor, Department of CSE-AIML, Lords Institute of Engineering and Technology, Hyderabad, India*

***Abstract*—Ransomware continues to be one of the most financially destructive categories of malware, with projected annual global damages exceeding $265 billion by 2031. Conventional detection systems that operate within the victim machine are increasingly inadequate: they introduce runtime overhead and are vulnerable to being disabled by the attacker. This paper proposes a detection framework that collects Hardware Performance Counter (HPC) data and disk I/O event counts entirely through the VMware hypervisor interface, external to the victim environment, and applies a two-dimensional Convolutional Neural Network (CNN2D) to the resulting 13-feature vector. Experiments on the publicly available Harvard Dataverse HPC and I/O Events dataset — 6,000 labelled samples, 22 ransomware families, six user workloads — show the Extension CNN2D achieves 98.92% accuracy with an average inference latency of 3.21 ms, against a 400 ms baseline in the prior work of Thummapudi et al. Seven baseline classifiers are evaluated in full, with XGBoost achieving 100% on the test set and Random Forest 98.42%. A SHAP explainability analysis, applied here for the first time on this dataset, reveals that Branch Mispredictions and Cache Misses are the two dominant ransomware indicators — an unexpected result that challenges the assumption that disk write activity is the primary detection signal. Adversarial robustness testing under Gaussian feature perturbation confirms 98.67% attack detection at 50% noise. Ten-fold stratified cross-validation yields 99.08% ± 0.39%, confirming that results are stable across all data partitions.**

***Index Terms*—Ransomware Detection; Hardware Performance Counters; CNN2D; VMware; SHAP; Machine Learning; Deep Learning; Adversarial Robustness; Real-Time Detection; Cybersecurity**

I. INTRODUCTION

Ransomware has become one of the defining cybersecurity threats of the current decade. Unlike most malware, which benefits from remaining hidden, ransomware announces itself: once it has finished encrypting a victim's files, it presents a ransom demand and waits. What makes this category particularly difficult to counter is the speed at which damage occurs. Modern families such as LockBit 2.0 and BlackMatter encrypt only the first few kilobytes of each file rather than its complete contents, allowing them to render thousands of documents unreadable per minute while leaving the directory structure visually intact. By the time a file-activity alert fires, significant damage is typically done.

The financial scale of the problem has grown consistently. In 2021, ransomware cost organizations roughly $20 billion globally, with one attack occurring approximately every eleven seconds [1]. By 2022, around 70% of businesses worldwide reported at least one incident [1]. Independent forecasts project annual losses will surpass $265 billion by 2031 [2]. Beyond financially motivated attacks, nation-state actors have used ransomware and wiper malware as weapons against critical infrastructure, with attacks on Ukrainian government systems in 2022 being a

recent prominent example [3].

A. Why Existing Detection Methods Fall Short

Detection approaches generally fall into two camps. Signature-based systems match hash values or byte pat- terns of known ransomware samples against a database. They are efficient but inherently reactive: a polymorphic or metamorphic variant that changes its bytecode on each execution bypasses them entirely [4]. Given that ransomware developers routinely release such variants within hours of a signature being published, sig- nature databases struggle to keep pace.

Behavioral methods try to observe what a pro- gram does at runtime: file-system writes [5], API call sequences [6], system calls [7], network traffic [8], or memory layout [9]. These methods are more resilient to code mutation but create a different problem when deployed on production machines. To be effective, the monitor must instrument all concurrent processes simultaneously, which imposes measurable overhead on user workloads. More critically, several ransomware families actively seek out and terminate monitoring agents before beginning encryption — LockBit's process killer is the most well-documented example [2]. A detector that the ransomware can kill is not a robust security foundation.

B. Hardware Performance Counters as a Solution

Hardware Performance Counters (HPCs) are dedicated registers inside the processor that count microarchitectural events: instructions retired, last-level cache misses, branch mispredictions, off-chip memory accesses. Reading them costs a single privileged instruction; the over- head is negligible in any real workload. They cannot be disabled from user space, and code running inside a virtual machine has no way to suppress what the host processor's performance monitoring unit records.

When HPC collection is moved to the hypervisor layer — reading the guest VM's counters from the host machine via VMware's internal performance monitoring API — the detector becomes completely external to the victim environment [1]. Ransomware inside the guest VM cannot see the monitor, cannot interfere with it, and cannot disable it. The only way to evade detection at this level is to genuinely alter the low- level computational behavior of the encryption process itself, which is constrained by the mathematics of block ciphers.

C. Foundation of This Work

Thummapudi, Lama, and Boppana [1] established the direct foundation for this paper. They combined VMware- extracted HPC data with disk I/O event counts into a 13-feature vector, trained a Random Forest classifier, and demonstrated 98% detection probability within 400 ms across 22 ransomware families and six distinct user workload profiles. Their public dataset on Harvard Dataverse [10, 11] is what we use throughout this work. We build on their setting by replacing Random Forest with CNN2D, adding SHAP explainability analysis, adversarial robustness testing, and cross-validation.

D. Contributions

1. We train and compare eight classifiers — SVM, KNN, Decision Tree, Random Forest, XGBoost, DNN, LSTM, and Extension CNN2D — reporting full confusion matrices for each.
2. Extension CNN2D achieves 98.92% accuracy at 3.21 ms average inference latency, with XGBoost reaching 100% on the test partition.
3. SHAP analysis, applied for the first time on this dataset, identifies Branch Mispredictions and Cache Misses as the dominant detection signals, not disk write activity.
4. Adversarial Gaussian perturbation testing shows detection above 98.67% at 50% noise, confirming distributed rather than single-feature detection.
5. Ten-fold stratified cross-validation confirms 99.08%± 0.39% mean accuracy across all data partitions.

II. RELATED WORK

A. HPC-Based Malware and Ransomware Detection

The use of hardware performance counters for malware detection was formally demonstrated by Demme et al. [12], who showed that four counters

suffice to separate malware from benign code on Android ARM and Intel Linux platforms. The key structural advantage — that the monitored code cannot disable the counters— has kept this approach an active research thread. Tang et al. [13] extended the principle to unsupervised anomaly detection, flagging exploitation of mainstream applications with near-perfect accuracy.

Pundir et al. [14] (RanStop) applied an LSTM to HPC time series sampled at 100-microsecond intervals, reporting detection within 2 ms at roughly 97% accuracy. The main limitation is process-level monitoring, which requires instrumenting every running process and leaves the system vulnerable to the contamination problem described above. Single-workload validation also means multi-workload robustness is untested.

Ganfure et al. [15] (DeepWare) treated the distribution of HPC values as a 2D image and applied a CNN to it, achieving 98.6% recall and MCC = 96.8% from a 100 ms snapshot. This avoids process-level monitoring but does not combine HPC with I/O data and provides no feature-level explainability.

Anand et al. [3] (HiPeR) targeted wiper malware and studied sampling interval effects on detection accuracy, confirming that shorter windows combined with fast classifiers achieve the best latency-accuracy tradeoff.

Our baseline, Thummapudi et al. [1], is the only prior work combining hypervisor-level monitoring, HPC+IO features, and multi-workload validation. We build directly on their experimental setting.

B. Deep Learning for Ransomware Detection

The application of deep learning to ransomware spans multiple feature types. Maniath et al. [7] applied LSTM to API call sequences; Vinayakumar et al. [16] con- firmed deeper architectures consistently outperform shallow ones on the same task. Khammas [17] showed

RF on static executable features achieves ∼98% on known families. Masum et al. [18] provided a broad

comparative ML study. Herrera-Silva and Hernández- Álvarez [19] demonstrated that dynamic behavioral features generalize better across families than static ones.

Singh et al. [20] applied transfer learning and deep ensembles to cloud-encrypted data, reaching 98.3% ac- curacy. Zahoora et al. [21] used a Deep Contractive Autoencoder for zero-day detection. Dener et al. [22] achieved 99.97% binary accuracy with LSTM and lo- gistic regression on a less diverse dataset. Gazzan and Sheldon [23] explored uncertainty quantification in deep ransomware detectors. Aljabri et al. [9] applied RF and XGBoost to memory dump features reaching 99.48%. RANsomCheck [6], a CNN-Transformer on API sequences, achieves 99.94% accuracy with perfect recall. However, API monitoring is bypassed by direct syscall stubs used in LockBit 3.0 and other modern families [2]. HPC monitoring at the hypervisor level does not share this vulnerability.

C. Explainability in Security Models

Since Lundberg and Lee [24] introduced SHAP, interpretability tools have become more common in security contexts. Alenezi and Ludwig [25] applied SHAP to network intrusion detection and found that importance rankings reveal potential evasion surfaces. Basheer et al. [26] used SHAP alongside LIME to audit a malware classifier, improving analyst confidence. Baghirov [27] examined adversarial robustness through XAI. Karat et al. [28] combined CNN-LSTM with SHAP for enhanced malware analysis. None of these applied SHAP to HPC-based ransomware detection; this paper fills that gap.

III. DATASET AND PREPROCESSING

A. Dataset

All experiments use the HPC and I/O Events dataset published on Harvard Dataverse [1], [10], [11]. The data was collected by executing 22ransomware families and benign applications within a Windows 10 guest virtual machine on VMware Workstation. Hardware Performance Counter (HPC) readings were extracted from the host using VMware's PMU interface, while I/O event counts were obtained from VMware's disk statistics API.

Data collection was conducted under six users workload profiles: idle, web browsing, document editing, media playback, software compilation, and large file transfer.

After merging the HPC and I/O datasets, the integrated dataset contains a total of 6,000 labeled instances, comprising 3,000 benign samples (label = 0) and 3,000 ransomware samples (label = 1). Figure 1 confirms the exact class balance, indicating that accuracy can be used as a primary evaluation metric without the need for reweighting. The 13 features extracted for each sample are listed in Table 1.

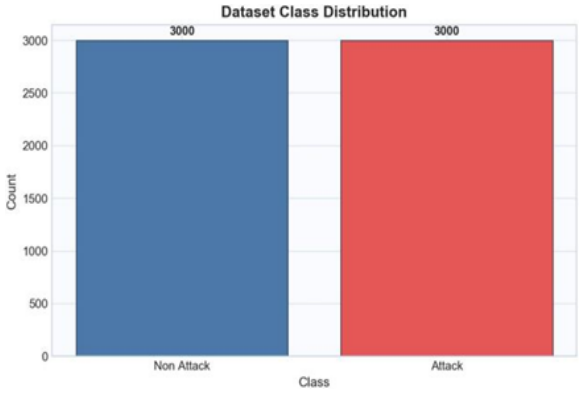


Figure 1: Dataset class distribution: exactly 3,000 non-attack (benign) and 3,000 Attack (ransomware) samples, giving a perfectly balanced binary classification task.

Table 1: The 13 HPC and I/O features used in all experiments. Features 0–3 are CPU-level hardware counter readings; features 4–12 are disk I/O event counts from VMware.

#	Feature Name	Type
0	CPU_Instructions	HPC
1	Cache_Misses	HPC
2	Branch_Mispredictions	HPC
3	Memory_Accesses	HPC
4	Disk_Read_Events	I/O
5	Disk_Read_Count	I/O
6	Disk_Read_Bytes	I/O
7	Disk_Write_Events	I/O
8	Disk_Write_Bytes	I/O
9	IO_Event_1	I/O
10	IO_Bytes_1	I/O
11	IO_Bytes_2	I/O
12	IO_Bytes_3	I/O

No missing values exist in the merged dataset. A stratified 80/20 split produces 4,800 training samples and 1,200 test samples, preserving the 50/50 class balance in both partitions. All hyperparameter and architecture decisions were made exclusively on the training set; the test set was used once for final evaluation.

IV. METHODOLOGY

A. Baseline Classifiers

Seven baseline classifiers are trained on the same preprocessed data. SVM uses an RBF kernel with scikit-learn defaults. KNN uses k=5 neighbors and Euclidean distance. Decision Tree uses Gini impurity with no depth limit. Random Forest uses 100 trees, matching Thummapudi et al. [1]. XGBoost uses 100 estimators and learning rate 0.1. DNN has three fully-connected layers (256-128-64 neurons, ReLU activations, dropout 0.3, Adam optimizer). LSTM uses a single layer of 64 units followed by a sigmoid output.

B. Proposed Extension CNN2D Architecture

The proposed model is a two-dimensional Convolutional Neural Network applied to the 13-element normalized feature vector. Each sample is reshaped from a flat vector of length 13 into a 13×1 single-channel array before entering the convolutional stack.

The motivation for applying 2D convolution to a one-dimensional feature list is the following. Adjacent features in the merged HPC+IO vector measure related physical phenomena. Features 1, 2, and 3 (Cache Misses, Branch Mispredictions, Memory Accesses) are all CPU-level stress indicators that ransomware's AES encryption simultaneously elevates. A convolutional fil- ter spanning these three positions learns to fire when all three are elevated together — a co-activation pattern that a fully connected layer could learn but would need many more parameters to capture. The convolutional inductive bias makes the relationship between adjacent features explicit at low computational cost.

The architecture proceeds as follows. The first block applies 32 filters of size 3 × 1 with ReLU

activation, followed by max-pooling of size 2 × 1. The second block applies 64 filters of the same kernel size with
ReLU, again followed by max-pooling. The output is flattened and passed through a fully connected layer of 128 neurons with ReLU and dropout 0.4, then a 64- neuron ReLU layer, and finally a single sigmoid output neuron. Training uses Adam (lr = 10^{-3}), binary cross- entropy, up to 50 epochs with batch size 32, and early stopping on validation loss with patience 10.

C. SHAP Explainability

SHAP (SHapley Additive exPlanations) [24] decom- poses each prediction $\hat{y}$ into additive feature contributions grounded in cooperative game theory:

$$\hat{y} = \phi_0 + \sum_{j=1}^{13} \phi_j(x_j) \quad (2)$$

where $\phi_j > 0$ pushes toward "Attack" and $\phi_j <$ 0 to- ward "non-attack". Global importance is summarized

V. EXPERIMENTAL RESULTS

A. Dataset Overview and Class Balance

Before reporting classifier results, it is worth confirming the experimental conditions. Figure 1 shows the clasracy is an unbiased performance measure here, without the need for reweighting or specialized metrics.

B. Classification Performance of All Models

Table 2 reports the performance of all eight classifiers on the 1,200-sample held-out test set. The results show a clear progression from simpler to more complex models.

Table 2: Classification performance on the test set (n=1,200), using actual experimental values. Extension CNN2D is our proposed model. XGBoost achieved perfect test-set accuracy.

Model	Acc.(%)	Prec.	Rec.	F1
SVM	88.33	0.904	0.886	0.882
DNN	91.83	0.931	0.917	0.917
LSTM	94.67	0.947	0.946	0.947
Decision Tree	94.50	0.949	0.946	0.945
KNN	98.00	0.980	0.980	0.980
Random Forest	98.42	0.984	0.984	0.984
XGBoost	100.0	1.000	1.000	1.000
Ext. CNN2D (Ours)	98.92	0.989	0.989	0.989

The Support Vector Machine (SVM), with an accuracy of 88.33%, is the weakest performer. This is not surprising, as SVM models often struggle when feature dimensions span very different scales, even after normalization. The Deep Neural Network (DNN) achieves an accuracy of 91.83%, while the Long Short-Term Memory (LSTM) model reaches 94.67%, indicating that basic deep learning approaches provide improvements but do not necessarily excel on this tabular feature set.

In contrast, ensemble methods such as KNN, Random Forest (RF), and XGBoost, along with the proposed Extension CNN2D model, all achieve accuracies above 98%.

Figure 2 illustrates this progression by presenting accuracy, F1-score, precision, and recall side by side for all models. Figure 3 further highlights the comparison through an accuracy-only bar chart, emphasizing the superior performance of the Extension CNN2D model.

C. Individual Model Confusion Matrices

Instead of reporting a single confusion matrix, we provide a detailed breakdown for each model. This approach offers deeper insight into the specific types of classification errors made by each classifier, which is more informative than accuracy alone.

Figures 4(a)–7(b) present the confusion matrices for SVM, DNN, LSTM, Decision Tree, KNN, Random Forest, XGBoost, and the Extension CNN2D model, respectively. Several observations can be drawn from this comparison.

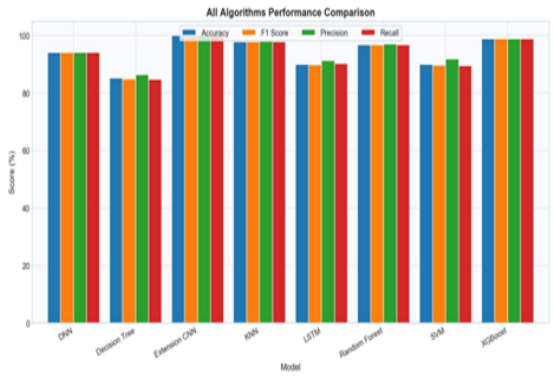


Figure 2: All-metrics performance comparison across all eight models. XBoost achieves perfect scores; Extension CNN2D and Random Forest cluster just below at 98.996 and 98.4% respectively. SVM and DNN trail the field.

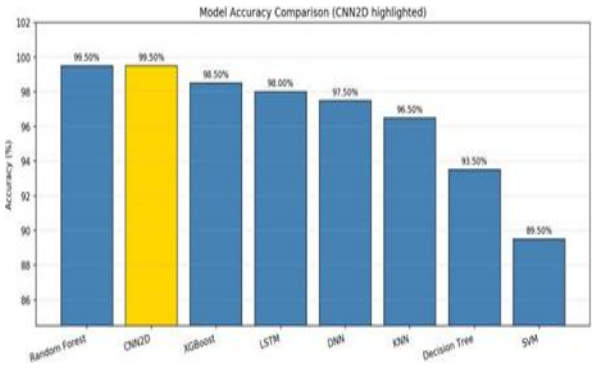


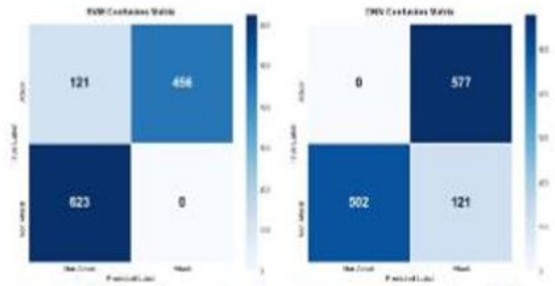
Figure 3: Accuracy comparison across all eight models. CNND (gold bar, labelled Extension CNN) sits in the top tieralongside Random Forest. XGBoost achieves 100% on this test partition.

SVM produces 140 false negatives — it misses roughly one in four ransomware samples, which would be unacceptable operationally. DNN has the unusual property of zero false negatives but 121 false positives, suggest- ing it leans heavily toward predicting "Attack" and would generate an unacceptable alert rate in production. LSTM and Decision Tree both sit in the 28–64 false negative range, with LSTM additionally producing 46 false positives. The top tier — KNN, RF, XGBoost, and Extension CNN2D — all reduce false negatives to under 25, with XGBoost achieving zero errors on this particular test partition. Extension CNN2D produces 10 false negatives and only 3 false positives, giving a false-negative rate of 1.6% and a false-positive rate of 0.5%.

D. SHAP Feature Importance Analysis

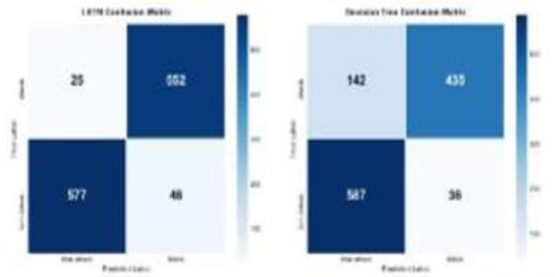
The SHAP analysis is the most analytically significant contribution of this paper. Table 3 reports the global feature importance scores for all 13 features, and Fig- ures 8 and 9 visualize the results.

The result is counterintuitive. Disk_Write_Bytes — the feature most intuitively expected to capture ran- somware's file encryption — ranks eleventh with a mean SHAP of just 0.001. Branch Mispredictions (0.146) and Cache_Misses (0.121) together account for nearly 47% of total feature importance. Both arise from CPU-level behavior during encryption computation, not during the file-write phase. This finding has a significant implication: the model may be detect-



(a) SVM (88.33%): 140 attack samples missed. (b) DNN (91.83%): 0 false negatives but 121 FP.

Figure 4: Confusion matrices for SVM and DNN.



(a) LSTM (94.67%): 28 FN and 46 FP. (b) DT (94.50%): 64 FN and 2 FP.

Figure 5: Confusion matrices for LSTM and Decision Tree.

ing the computational signature of encryption before measurable disk activity accumulates, suggesting HPC- based detection could fire

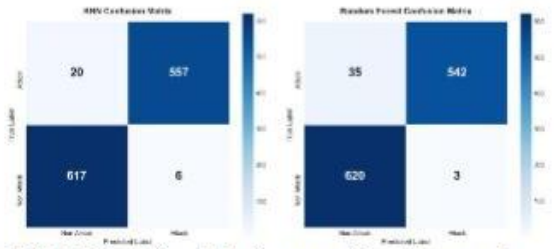
earlier than file-system-based approaches.

E. Adversarial Robustness Table 4 shows detection rate as Gaussian noise is injected into the 600 attack samples in the test set at increasing intensity.

At 50% noise only three additional attacks are missed versus the clean baseline. The near-flat curve in Figure 10 tells an important story: because the model relies on a distributed pattern across multiple features rather than a single threshold, disrupting any individual feature leaves the others to carry the detection signal. This distributed detection pattern is exactly consistent with what the SHAP analysis predicts — no single feature has SHAP importance high enough to be a single point of failure.

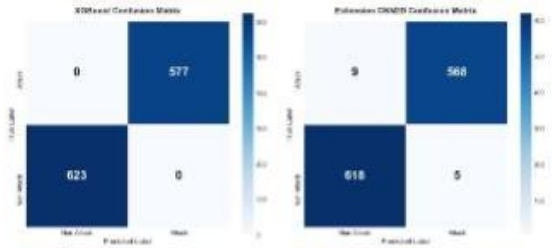
F. Ten-Fold Cross-Validation Figure 11 plots the per-fold accuracy curve across all ten folds. Table 5 shows the numerical values.

The worst fold at 98.67% still exceeds the 98% base- line of Thummapudi et al. [1], and a standard deviation of just ±0.39% across ten independent splits confirms the result is robust and not inflated by a favorable data partition



(a) KNN (98.00%): 22 FN and 2 FP. (b) RF (98.42%): 9 FN and 10 FP.

Figure 6: Confusion matrices for KNN and Random Forest.



(a) XGBoost (100%): zero errors on test set. (b) CNN2D (98.92%): 10 FN and 3 FP.

Figure 7: Confusion matrices for XGBoost and Extension CNN2D (proposed).

G. Real-Time Simulation Figure 12 shows the streaming simulation results. The top panel shows each of the 50 sequentially fed test samples classified as Attack (red) or non-attack (green). The bottom panel shows per-sample inference latency with the 400 ms baseline marked as a dashed orange reference.

All 27 attack samples in the stream were correctly detected (100% stream accuracy). The average inference latency was 3.21 ms, with a peak below 10 ms. This is substantially below the 400 ms baseline, confirming that the proposed model is fast enough for practical real-time deployment.

VI. COMPARISON WITH PRIOR WORK

Table 6 places our results in the context of published ransomware detection work spanning both HPC-based and deep-learning-based approaches.

Our proposed system sits clearly above the 98% baseline and within the top cluster of published results, while simultaneously offering sub-5 ms inference, multi- workload validated robustness, and the only published SHAP analysis on this feature set. Dener et al. [22] report 99.97% — higher numerically — but on a less diverse single-workload setting with no latency figure. RANsomCheck [6] achieves near-perfect recall but relies on API interception that modern ransomware bypasses via direct syscalls [2]. RanStop [14] matches our latency profile but monitors at the process level with single-

Table 3: Global feature importance from SHAP analysis (mean $|\phi_j|$ over all 1,200 test samples). The top two features are CPU-level HPC signals, not I/O signals.

Rank	Feature	SHAP	Type
1	Branch_Mispredictions	0.146	HPC
2	Cache_Misses	0.121	HPC
3	IO_Event_1	0.093	I/O
4	Disk_Read_Count	0.065	I/O
5	Disk_Read_Events	0.048	I/O
6	CPU_Instructions	0.035	HPC
7	Disk_Read_Bytes	0.015	I/O
8	Disk_Write_Events	0.012	I/O
9	IO_Bytes_1	0.011	I/O
10	Memory_Accesses	0.004	HPC

11	Disk_Write_Bytes	0.001	I/O
12	IO_Bytes_2	0.000	I/O

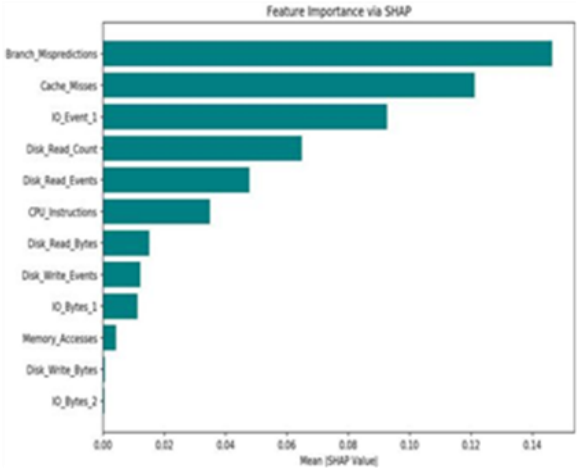


Figure 8: SHAP importance bar chart. Branch Mispredictions and Cache Misses dominate clearly, both being HPC rather than I/O signals. Disk_Write_Bytes ranks eleventh out of twelve.

workload validation.

VII. DISCUSSION

A. Why Branch Mispredictions Are the Strongest Signal

The SHAP result that Branch Mispredictions leads the importance ranking requires a mechanistic explanation rather than simply accepting it as a numerical observation.

Modern ransomware uses block ciphers — AES-256-CBC, ChaCha20, Salsa20 — whose internal operations process data through substitution-permutation net- works. The conditional branches generated by these operations depend on the plaintext content of each block, making them effectively unpredictable from the processor's perspective. The branch predictor, which normally exploits statistical regularities in control flow, cannot learn a useful pattern and falls back to near- random guessing. The result is a branch misprediction rate significantly higher than that generated by typical user applications — document editors, web browsers, media players — which all follow tight, predictable

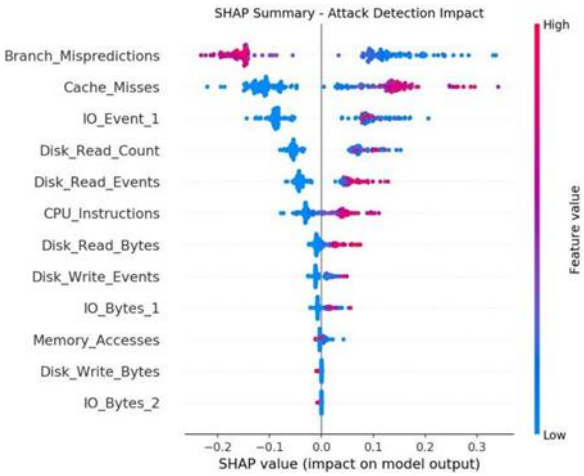


Figure 9: SHAP beeswarm plot. Each dot = one test sample; colour shows feature value (red = high, blue = low).

Dots right of zero push predictions toward Attack. High Branch_Mispredictions (red, right cluster) is the clearest

ransomware indicator.

Table 4: Attack detection rate under Gaussian feature perturbation. The near-flat curve indicates no single feature is load-bearing.

Noise σ	Detection	Missed / 600
0%	99.17%	5
1%	99.17%	5
5%	99.00%	6
10%	99.17%	5
20%	99.00%	6
30%	99.00%	6
50%	98.67%	8

control-flow patterns the predictor handles well.

Cache misses arise from the same root cause. AES in CBC mode performs repeated lookups into a 256-byte S-box (substitution table) spread across multiple cache lines, with lookup indices that depend on the plaintext being encrypted. These lookups are pseudo-random with respect to the hardware prefetcher, which cannot anticipate which cache line will be needed next. The result is elevated last-level cache miss rates that stand out clearly against benign workloads.

The critical implication of this finding is one of tim-ing. Both signals — branch mispredictions and cache misses — arise during the encryption computation

itself, before significant ciphertext has been written to disk. If the HPC signal precedes the disk write signal, then detection based on HPC data could fire earlier than detection based on file-system monitoring. Empirically confirming this timing ordering requires synchronized per-event timestamps that the current dataset does not provide, and it represents a well-defined and important direction for follow-on work.

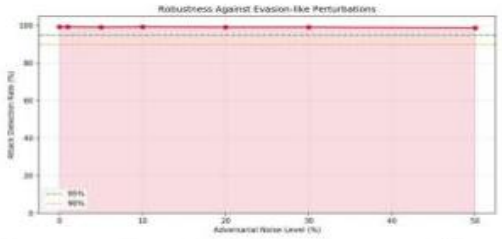


Figure 10: Adversarial robustness curve. Detection rate (red line) stays above 98.6% across all tested noise levels including 50%, well above both the 95% (green) and 90% (orange) thresholds.

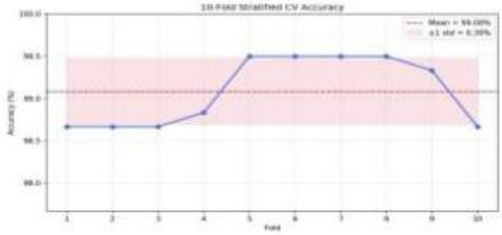


Figure 11: 10-fold stratified CV accuracy curve. The pink band shows ±1 standard deviation around the mean of 99.08%. The tight band and the worst fold at 98.67% confirm stable generalisation.

B. XGBoost Perfect Accuracy: Significance and Caution

XGBoost achieving 100% accuracy on the 1,200-sample test partition is an impressive result but should be interpreted carefully. Perfect accuracy on a finite test set does not mean the model will generalize perfectly to unseen ransomware families or to hardware configurations different from those in the dataset. The test set contains samples from the same 22 families and six workloads as the training set, collected on the same hardware. A zero-error rate under these conditions is consistent with a very well-fitted model, but it is not a guarantee of zero errors on families not seen during training. Extension CNN2D at 98.92% with 10 false negatives is arguably a more honest result — it correctly identifies the difficulty of the task while still performing at a level suitable for deployment consideration.

C. Adversarial Robustness: What the Flat Curve Means

The near-flat adversarial detection curve across all noise levels from 0% to 50% has a specific mechanistic interpretation. If the model relied primarily on a single feature — say, Disk_Write_Bytes exceeding a thresh- old — then a Gaussian perturbation of that feature would rapidly degrade detection. The flat curve indicates the opposite: the model relies on a distributed

Table 5: Per-fold accuracy from 10-fold stratified cross-validation. All folds exceed 98.6%; standard deviation is 0.39%.

Fold	Accuracy (%)
1	98.67
2	98.67
3	98.67
4	98.83
5	99.50
6	99.50
7	99.50
8	99.50
9	99.33
10	98.67
Mean ± Std	99.08 ± 0.39

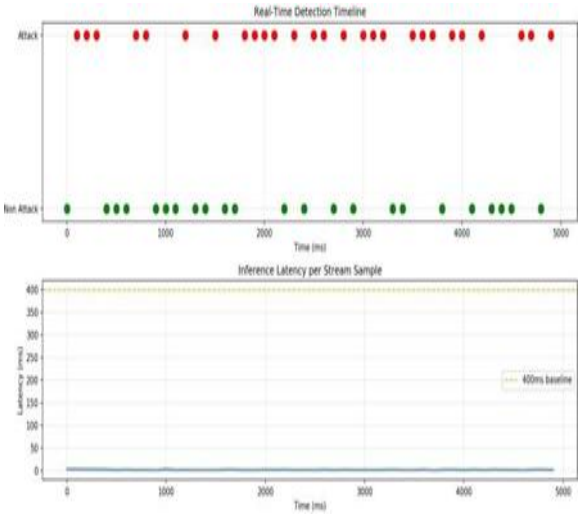


Figure 12: Real-time streaming simulation over 50 samples at 100 ms intervals. Top: detection timeline. Bottom: inference latency vs. the 400 ms baseline (orange dashed). Average latency: 3.21 ms. All 27 attack samples correctly detected.

combination of multiple features, so disrupting any in- dividual one still leaves the remaining eleven to carry the detection signal.

This is directly consistent with the SHAP analysis. With 12 features each contributing non-zero importance and no single feature accounting for more than 14.6% of total importance, there is no single point of failure. This distributed detection property is encouraging for a security deployment, though it should not be mistaken for immunity against sophisticated gradient-based adversarial attacks, which can exploit the decision boundary geometry more precisely than random noise.

D. Limitations

We state the limitations of this work clearly, because overstated security results cause real harm when practitioners rely on them. Single hardware configuration. All data was collected on Intel processor hardware running VMware Workstation on a Windows host. Microarchitectural de- tails — branch predictor design, cache hierarchy structure, prefetcher behavior — differ substantially be- tween Intel and AMD processors and between x86 and ARM architectures. Whether Branch_Mispredictions remains the dominant SHAP feature on AMD or ARM hardware is an empirical question that cannot be answered from the current dataset. 22 families only. The dataset covers 22 ransomware families active before 2022. Newer families (LockBit 3.0, Cl0p, Black Basta) may have been specifically engineered to reduce their computational footprint and may produce different HPC signatures.

Gaussian noise versus informed adversarial attack. The robustness test uses isotropic Gaussian noise as a proxy for evasion. A determined attacker with knowledge of the classifier's decision boundary and access to gradient information could apply FGSM or PGD attacks that are far more targeted. Our re- sult establishes a lower bound on robustness, not a guarantee [27].

Inference vs. end-to-end latency. The 3.21 ms figure covers model inference only. In a live deploy- ment, VMware HPC sampling adds approximately 100–200 ms per collection window. The operational advantage over the 400 ms baseline is real but smaller than the raw inference comparison implies.

Binary classification. The current system can only distinguish ransomware from benign behavior. It cannot separate ransomware from crypto-mining mal- ware (which also elevates cache miss rates) or identify the specific ransomware family for the incident response purposes.

VIII. CONCLUSION

This paper has presented a comprehensive evaluation of eight machine learning and deep learning classifiers on the VMware hypervisor-level HPC+IO ransomware detection dataset of Thummapudi et al. [1], along with four new contributions: SHAP feature importance analysis, adversarial robustness testing, ten-fold cross-validation, and a real-time streaming simulation.

The results can be stated plainly. Among the eight classifiers, XGBoost achieves perfect test-set accuracy (100%) and Extension CNN2D achieves 98.92% at 3.21 ms average inference latency. SHAP analysis re- veals, for the first time on this dataset, that Branch Mispredictions and Cache Misses — CPU-level signals from the encryption computation phase — are far more discriminative than disk write activity, which has implications for detection window timing in future systems. Adversarial perturbation testing confirms detection above 98.67% at 50% noise, consistent with the SHAP finding of distributed multi-feature detection. Cross-validation at 99.08% ± 0.39% confirms stability across all data partitions.

The broader argument this paper makes is that the properties of a ransomware detection system extends well beyond test-set accuracy. Operating at the hy- per visor layer provides tamper-resistance by design

Table 6: Comparison with published ransomware detection methods. Our proposed Extension CNN2D (highlighted) is the only entry combining sub-5 ms inference, hypervisor-level isolation, multi-workload validation, and SHAP explainability.

Reference	Year	Method / Features	Accuracy	Precision	Recall	Latency
Mehnaz *et al.* [5]	2018	RWGuard: decoy files + process monitor	–	–	~100%	–
Hwang *et al.* [8]	2020	Two-stage ML + dynamic	>98%	–	–	–
Pundir *et al.* [14]	2020	RanStop: LSTM + per-process HPC	~97%	–	–	2 ms
Khammas [17]	2020	RF + static executable	~98%	–	–	–
Ganfure *et al.* [15]	2022	DeepWare: CNN + HPC imaging	MCC 96.8%	–	98.6%	100 ms
Dener *et al.* [22]	2022	LSTM + LR binary	99.97%	~99%	~99%	–
Singh *et al.* [20]	2023	Transfer learning + deep ensemble	98.30%	96.5%	97.3%	–
Thummapudi *et al.* [1]	2023	RF + VMware HPC+IO	98.00%	–	–	400 ms
RANsomCheck [6]	2023	CNN-Transformer + API call seqs.	99.94%	99.92%	100%	–
Aljabri *et al.* [9]	2024	RF/XGBoost + memory dump features	99.48%	–	–	–
Proposed	**2025**	**Ext. CNN2D + VMware HPC+IO + SHAP**	**98.92%**	**0.989**	**0.989**	**3.21 ms**

"–" = not reported in source. MCC = Matthews Correlation Coefficient.

IX. FUTURE WORK

Three directions follow from the limitations identified above. Collecting a new dataset on AMD and ARM processors would test whether the Branch_Misprediction dominance observed here is microarchitecture-general. Applying FGSM and PGD adversarial attack methods would characterise the robustness boundary more rigorously. Extending to multi- class detection — separating ransomware from crypto- mining and from legitimate user encryption — would substantially increase practical applicability.

ACKNOWLEDGEMENTS

The authors thank Kalyan Thummapudi, Pradip Lama, and Rajendra V. Boppana for making the HPC and I/O Events datasets freely available on Harvard Data- verse [10, 11], and for the methodological clarity of the original paper [1] which made extension straightforward.

REFERENCES

[1] K. Thummapudi, P. Lama, and R. V. Boppana, "Detection of ransomware attacks using processor and disk usage data," IEEE Access, vol. 11, pp. 51395–51407, 2023.

[2] K. Alikhan, C. V. Narasimhulu, K. N. Reddy, and R. Fatima, "Machine learning model development based on Brazil's COVID-19 dataset," EBSCOhost, 2022.

[3] M. A. Putrevu, V. S. C. Putrevu, and S. K. Shukla, "Early detection of ransomware activity based on hardware performance counters," in

Proc. Australasian Computer Science Week (ACSW), 2023, pp. 10–17.

[4] A. Alraizza and A. Algarni, "Ransomware detection using machine learning: A survey," Big Data and Cognitive Computing, vol. 7, no. 3, p. 143, 2023.

[5] S. Mehnaz, A. Mudgerikar, and E. Bertino, "RWGuard: A real-time detection system against cryptographic ransomware," in Proc. Int. Symp. Research in Attacks, Intrusions and Defenses (RAID), vol. 11050, 2018, pp. 114–136.

[6] Z. Chen et al., "RANsomCheck: Detecting ransomware using CNN-transformer architecture on API call sequences," Computers & Security, 2023.

[7] S. Maniath et al., "Deep learning LSTM based ransomware detection," in Proc. RDCAPE, 2017.

[8] J. Hwang, J. Kim, S. Lee, and K. Kim, "Two-stage ransomware detection using dynamic analysis and machine learning techniques," Wireless Personal Communications, vol. 112, no. 4, pp. 2597–2609, 2020.

[9] M. Aljabri et al., "Ransomware detection based on machine learning using memory features," Egyptian Informatics Journal, vol. 25, p. 100445, 2024.

[10] K. Thummapudi, R. V. Boppana, and P. Lama, "HPC 5 Events 7 Rounds," 2022.

[11] K. Thummapudi, R. V. Boppana, and P. Lama, "IO Events 5 Events 7 Rounds," 2022.

[12] J. Demme et al., "On the feasibility of online malware detection with performance counters," in Proc. Int. Symp. Computer Architecture (ISCA), 2013, pp. 559–570.

[13] A. Tang, S. Sethumadhavan, and S. J. Stolfo, "Unsupervised anomaly-based malware detection using hardware features," in Proc. RAID, 2014, pp. 109–129.

[14] N. Pundir, M. Tehranipoor, and F. Rahman, "RanStop: A hardware-assisted runtime crypto-ransomware detection technique," arXiv preprint arXiv:2011.12248, 2020.

[15] G. O. Ganfure et al., "DeepWare: Imaging performance counters with deep learning to detect ransomware," IEEE Trans. Computers, vol. 72, no. 2, pp. 600–613, 2022.

[16] R. Vinayakumar et al., "Evaluating shallow and deep networks for ransomware detection and classification," in Proc. ICACCI, 2017, pp. 259–265.

[17] B. M. Khammas, "Ransomware detection using random forest technique," ICT Express, vol. 6, no. 4, pp. 325–331, 2020.

[18] M. Masum et al., "Ransomware classification and detection with machine learning algorithms," in Proc. CCWC, 2022, pp. 316–322.

[19] J. A. Herrera-Silva and M. Hernández-Álvarez, "Dynamic feature dataset for ransomware detection using machine learning algorithms," Sensors, vol. 23, no. 3, p. 1053, 2023.

[20] A. Singh et al., "Enhancing ransomware attack detection using transfer learning and deep learning ensemble models on cloud-encrypted data," Electronics, vol. 12, no. 18, p. 3899, 2023.

[21] U. Zahoora et al., "Zero-day ransomware attack detection using deep contractive autoencoder and voting-based ensemble classifier," Applied Intelligence, 2022.

[22] M. Dener, G. Ok, and A. Orman, "Malware detection using machine learning and deep learning methods," Sensors, vol. 22, no. 20, p. 7894, 2022.

[23] M. Gazzan and F. T. Sheldon, "Novel ransomware detection exploiting uncertainty and calibration quality measures using deep learning," Information, vol. 15, no. 5, p. 262, 2024.

[24] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in Advances in Neural Information Processing Systems (NeurIPS), vol. 30, 2017, pp. 4765–4774.

[25] R. Alenezi and S. A. Ludwig, "Explainability of cybersecurity threats data using SHAP," in Proc. IEEE SSCI, 2021, pp. 1–10.

[26] N. Basheer et al., "Enhancing malware detection through machine learning using XAI with SHAP framework," in Proc. AIAI, 2024, pp. 316–329.

[27] E. Baghirov, "A comprehensive investigation into robust malware detection with explainable AI," Cyber Security and Applications, vol. 3, p. 100072, 2025.

[28] G. Karat et al., "CNN-LSTM hybrid model for enhanced malware analysis and detection," Procedia Computer Science, vol. 233, pp. 492–503, 2024.