

Hardware Robustness of Quantum Machine Learning Kernels for Malware Detection

Mary L. Cotrupi¹, Allison Aurilio², Sean Sanpietro³, Fabiana Torres⁴, Anwita Kasturi¹, Quinn Colognato¹, Brian Robert Callahan⁴

¹Dept. of Computer Science, Rensselaer Polytechnic Institute, 110 8th Street, Troy, NY 12180-3590 USA

²Dept. of Electrical and Computer Engineering, The College of New Jersey, 2000 Pennington Road, Ewing, NJ 08618-1100 USA

³Dept. of Computer Science, Misericordia University, 301 Lake Street, Dallas, PA 18612-7752 USA

⁴Dept. of Computer Science and Software Engineering, Monmouth University, 400 Cedar Avenue, West Long Branch, NJ 07764-1828 USA

cotrum@rpi.edu, aurilia2@tcnj.edu, ssanpiet@misericordia.edu, s1378769@monmouth.edu, kastua@rpi.edu, cologq@rpi.edu, bcallaha@monmouth.edu

Abstract

Quantum Machine Learning (QML) is beginning to show promise as an emerging technology for cybersecurity tasks. One such task is malware detection, which remains a difficult and critical problem for both individuals and today’s organizations. While there is a growing literature on applying QML to malware analysis, many studies remain confined to simulators, leaving open the question of how simulation results translate to the Noisy Intermediate-Scale Quantum (NISQ) machines that exist today. In this paper, we compare quantum-kernel classification across four execution environments—exact statevector evaluation, ideal shot-based Qiskit Aer, noisy Aer, and the IBM Quantum System One housed at Rensselaer Polytechnic Institute—and include matched classical baselines. Our contributions are twofold: first, we benchmark the full pipeline on real NISQ hardware; second, we provide a reproducible case study quantifying how hardware noise affects quantum-kernel geometry and downstream classification accuracy. Although we do not demonstrate quantum advantage, hardware-minus-exact balanced accuracy remained within two percentage points under the tested conditions despite substantial kernel distortion for the wider circuits. This is evidence of empirical stability, not formal statistical equivalence. Feature compression using principal component analysis (PCA) substantially reduced hardware-induced kernel distortion, although its effect on predictive accuracy was dataset dependent. Our results provide a grounded benchmark for cybersecurity practitioners evaluating NISQ-ready QML pipelines.

Code — <https://github.com/ibara/SRP2026>

Preprint: This is the peer-reviewed camera-ready version (*the Accepted Version*) of this paper, which authors may post (<https://aaai.org/aaai-publications/request-to-reproduce-copyrighted-materials>). This version will be updated with a link to the final published version, full citation for the final published version, and DOI of the final published version after publication.

Copyright © 2027, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

Introduction

Quantum computing is a new computing paradigm that is beginning to show promise in improving the way data is processed. It offers potential for representing high-dimensional data, which provides new opportunities for fields that traditionally use classical computing. Although quantum computing raises concerns about an increase in cyber threats, it also has the potential to help detect them.

Malicious binaries, more commonly called malware, are one such threat: malware continues to be a serious attack vector, one that is exceptionally difficult to address in theory (Cohen 1987) and in practice (Pachhala, Jothilakshmi, and Battula 2021). Recent work in quantum machine learning has shown modest potential for cybersecurity analysis tasks, for example in classifying malicious network traffic (Cotrupi and Callahan 2025). This suggests that further exploration into where quantum machine learning may benefit cybersecurity practitioners is warranted.

In this paper, we provide a four-environment comparison across several malware datasets, comparing quantum kernels evaluated exactly, on a simulator with and without noise, and on a real quantum computer. We also include matched classical models. Our contributions are twofold: first, we extend comparisons that are often limited to classical models and quantum simulation by running our pipeline on the quantum computer housed at Rensselaer Polytechnic Institute (RPI); second, we offer a practical case study of quantum-kernel behavior on this important cybersecurity task. Including both exact statevector evaluation and noiseless, shot-based Aer allows us to distinguish finite-shot effects from modeled and measured device noise.

The goal of this research is to evaluate the potential of quantum machine learning for detecting malware alongside traditional classical models. We sought to determine whether a quantum approach could be more accurate, faster, or more energy efficient when classifying unknown software as malware or safe. Because this experiment did not collect comparable end-to-end timing or system-level energy measurements, it directly evaluates predictive behavior and kernel distortion rather than speed or energy advantage.

We examined four malware datasets, studying how feature compression and NISQ hardware noise affect quantum-kernel geometry and malware-classification performance. The central result is that IBM hardware noticeably distorted the measured quantum kernels, especially for wider circuits, while the observed downstream classification changes remained small. Although we did not find evidence of quantum advantage, the experiment supports empirical hardware stability under the specific conditions tested.

It is our hope that this study inspires more work toward understanding real-world quantum systems for cybersecurity practitioners. As cybersecurity increases in importance for individuals, enterprises, and governments around the world, consistently evaluating the potential benefits and limitations of emerging technologies such as quantum computing is essential.

Background

The intersection of quantum computing and applied cybersecurity is developing into a rich avenue for practical research. For example, recent work has explored the potential for quantum computing to aid in Distributed Denial-of-Service (DDoS) traffic analysis (Cogburn, DeCusatis, and Spillane 2026).

Within malware detection specifically, quantum computing is emerging as a serious object of study. Previous work compared classical, hybrid, and purely quantum models for detecting Android malware, noting that small datasets yielded competitive results for purely quantum approaches, although these approaches did not scale to the largest datasets, where classical methods remained stronger (Mercaldo et al. 2022). Other work compared quantum models for malware classification, demonstrating accuracy–efficiency tradeoffs between a Quantum Multilayer Perceptron (QMLP) and a Quantum Convolutional Neural Network (QCNN): QMLP achieved higher accuracy, whereas QCNN offered greater training efficiency (Lopez et al. 2025).

A recurring theme across many of these malware studies is the lack of testing on real quantum hardware. Some work does not identify whether a real quantum computer or simulator was used (Joshi and Guha 2025), while other studies explicitly report simulator-only experiments (Ciaramella et al. 2022; Mercaldo et al. 2022; Lopez et al. 2025; Vyas, Lo, and Zhang 2025).

More recent work has begun to conduct experiments using real quantum hardware, for example in malicious network-traffic detection (Cotrupi and Callahan 2025), supply-chain malware detection (Passo, Kothavade, and Prevost 2025), and comparisons of Quantum Support Vector Machines (QSVMs) with Quantum Neural Networks (QNNs) (Barrué and Quertier 2025). These studies are particularly useful because quantum simulators, while valuable for validating an implementation, do not fully reproduce the noise of present-day NISQ hardware. Some real-hardware studies also incorporate malware datasets used in our study (Passo, Kothavade, and Prevost 2025); our work extends this direction across a corpus of malware datasets to better characterize the readiness of quantum kernels for malware detection.

Methodology

For our tests, we use a support vector machine (SVM) for binary classification, with malicious samples labeled 1 and benign samples labeled 0. Rather than a classical precomputed kernel, we provide the support vector classifier (SVC) with a quantum kernel. We then evaluate the behavior of the quantum kernel under simulated and real NISQ hardware conditions for the selected malware datasets.

Datasets We used four malware datasets for our comparison: CLaMP (Kumar 2020), EMBER2018 (Anderson and Roth 2018), CIC-YNU-IoTMal Dataset 2026 (Dadkhah et al. 2026), and a Malware and Benign dataset (MBR, as we refer to the Kaggle malware/benign collection published by Aashish.zzz (Aashish.zzz 2026)).

Table 1: Dataset configurations. Selected-row counts were obtained from deterministic source sampling and duplicate removal.

Dataset	Source rows	Selected rows	Config(s)
CLaMP	5,184	4,040	PCA-4, 10
MBR	1,156	895	PCA-4, 8
CIC-YNU-IoTMal	2,121,949	1,280	PCA-4
EMBER2018	799,912	1,200	PCA-4

Three datasets—CLaMP, MBR, and EMBER2018—contain static Windows Portable Executable (PE) features, which are widely used in machine-learning-based malware classification (Ucci, Aniello, and Baldoni 2019). CIC-YNU-IoTMal instead contains Linux system-activity reports collected from IoT binaries executed in ARM, MIPS, MIPSSEL, and x86 sandboxes as an additional benchmark.

Preprocessing & Data Splitting CLaMP and MBR were evaluated using both selected native features and four-component PCA representations; CIC-YNU-IoTMal and EMBER2018 were evaluated using PCA-4 only (Table 1). For each of the six configurations, we repeated the experiment on three held-out splits. Each split contained 80 training and 80 test samples, balanced between benign and malicious classes. Test samples were reserved before training so that no sample appeared in both sets within a split, although training samples could overlap across splits. The experiments are therefore repeated holdouts rather than independent replicates.

Duplicate feature vectors were removed before splitting. All preprocessing operations were fitted using the training partition of each split and then applied to its test partition. Invalid values were treated as missing and median-imputed. Native features were min–max scaled. For PCA-4, features were standardized, projected onto four principal components, and min–max scaled; test values outside the training range were clipped. This split-specific fitting prevented information from the test data from entering imputation, scaling, or PCA.

Quantum Kernel Implementation

An n -qubit system spans a 2^n -dimensional Hilbert space. We map n classical input features to quantum states in this Hilbert space, using one qubit per input feature. Thus, for PCA-4, the

resulting quantum state lies in a $2^4 = 16$ -dimensional Hilbert space. We use Qiskit’s `zz_feature_map` (Javadi-Abhari et al. 2024) with two repetitions and linear entanglement for this encoding. Defining the encoded quantum state as

$$|\phi(\mathbf{x})\rangle = U_\phi(\mathbf{x})|0^d\rangle,$$

the fidelity kernel is given by

$$K(\mathbf{x}, \mathbf{z}) = \left| \langle 0^d | U_\phi^\dagger(\mathbf{z}) U_\phi(\mathbf{x}) | 0^d \rangle \right|^2, \quad (1)$$

where $d \in \{4, 8, 10\}$ is both the number of selected input features and the number of qubits used in the feature map.

The circuit first applies $U_\phi(\mathbf{x})$, followed by $U_\phi^\dagger(\mathbf{z})$. Here, $U_\phi(\mathbf{x})$ is the circuit constructed by `zz_feature_map` with its parameters bound to the input feature vector $\mathbf{x}$. The amplitude of the $|0^d\rangle$ state after these operations corresponds to the overlap between the quantum states encoding the samples $\mathbf{x}$ and $\mathbf{z}$. The kernel value $K(\mathbf{x}, \mathbf{z})$ is the squared magnitude of this overlap and can therefore be estimated as the probability of measuring the $|0^d\rangle$ state.

To maintain a valid training kernel in the presence of finite-shot sampling and device noise, we first applied $K \leftarrow (K + K^T)/2$ and set the diagonal to one. We then eigendecomposed the kernel, clipped negative eigenvalues to zero, reconstructed the matrix, and renormalized its diagonal to one (Schölkopf and Smola 2002; Javadi-Abhari et al. 2024). The rectangular test kernel was not projected. For measured kernel K_m and exact kernel K_e , relative Frobenius error was

$$E_F = \frac{\|K_m - K_e\|_F}{\|K_e\|_F}. \quad (2)$$

We also report the Pearson correlation between corresponding test-kernel entries (Golub and Van Loan 2013; Pearson 1895).

We tested four environments: (1) exact statevector evaluation, (2) ideal shot-based Qiskit Aer simulation, (3) noisy Aer simulation, and (4) `ibm_rensselaer` hardware execution. The three sampling environments used 512 shots. Exact statevector evaluation provides the mathematical reference, while ideal Aer isolates finite-shot and transpilation effects before simulated or measured hardware noise is introduced. The same transpiled circuit was used in the sampling environments. Noisy Aer used a backend-derived noise model taken from `ibm_rensselaer` calibration data, including its gate and readout errors. Transpilation used the backend coupling map and its native `rz`, `sx`, `x`, and `cz` basis gates. Dynamical decoupling, twirling, and measurement-error mitigation were not applied.

Classification and Statistical Analysis Quantum kernels were supplied to a precomputed-kernel SVC with $C = 1$. The matched linear and radial-basis-function (RBF) SVC baselines also used $C = 1$ without hyperparameter tuning. Balanced accuracy was the primary predictive metric, and ROC AUC was secondary.

Reported means combine the three repeated splits. The 95% confidence intervals were obtained from 10,000 class-stratified bootstrap replicates, with resampling performed within each class and split. Hardware-versus-exact intervals

Table 2: Transpiled circuit resources for `ibm_rensselaer`. Circuits were transpiled at optimization level 3 with transpiler seed 42. Duration is the estimated single-circuit duration based on backend instruction timings.

Qubits	Depth	Circuit size	CZ gates	Duration (μ s)
4	98	174	22	5.308
8	170	400	54	7.524
10	206	518	70	8.660

used the same test rows. Exact McNemar tests compared these paired predictions, and Holm adjustment was applied with $\alpha = 0.05$ (McNemar 1947; Holm 1979). We interpret nonsignificant differences as observed stability rather than proof of equivalence and must limit population-level inference.

Results

Table 3 and Figure 1 summarize predictive performance. Across the six configurations, hardware-minus-exact balanced accuracy remained within two percentage points. Each paired bootstrap interval included zero, and none of the six paired hardware-versus-exact comparisons was significant after Holm adjustment (all adjusted $p = 1.0$). These results show small observed differences under the tested conditions but do not establish statistical equivalence.

Kernel distortion increased with circuit width (Figure 2). Mean hardware relative error was approximately 0.11 for the four-qubit configurations and 0.44–0.48 for the wider circuits, while correlations with the exact kernels remained between 0.924 and 0.997. Across the primary hardware runs, CZ count was strongly associated with relative kernel error ($r = 0.857$, Spearman rank correlation $\rho = 0.812$) but not with hardware-minus-exact balanced accuracy ($r = -0.032$, $\rho = -0.129$). This is a descriptive association: CZ count had only three levels, splits were clustered within configurations, and width, depth, circuit size, and CZ count increased together.

PCA reduced circuit width, depth, and CZ count but produced a dataset-dependent predictive tradeoff. For CLaMP, hardware balanced accuracy decreased by 10.0 points relative to the native representation ($p = 0.0179$ after Holm adjustment). The corresponding MBR decrease was 2.9 points ($p = 0.458$). The matched classical comparison in Figure 3 showed no consistent quantum advantage: RBF SVC matched or exceeded hardware-quantum performance on CLaMP, CIC, and EMBER2018, whereas the quantum kernels were descriptively higher on MBR.

The hardware experiments required approximately 103 million shots and 28,000 seconds of IBM-reported aggregate QPU usage. This aggregate usage is distinct from the estimated duration of one transpiled circuit in Table 2.

Discussion

The principal result was the stability of hardware classification despite measurable changes in the quantum kernels from

Table 3: Primary exact and IBM hardware results. BA is balanced accuracy, intervals are 95% class-stratified bootstrap intervals, Δ is hardware minus exact in percentage points, AUC is reported as exact/hardware, and hardware kernel error and correlation are relative to the exact test kernel.

Configuration	Exact BA [95% CI]	Hardware BA [95% CI]	Δ	AUC E/H	HW error	HW corr.
CLaMP native-10	.842 [.796, .883]	.846 [.800, .888]	+0.4	.926/.915	.479	.970
CLaMP PCA-4	.750 [.696, .804]	.746 [.692, .800]	−0.4	.839/.817	.112	.996
MBR native-8	.975 [.954, .992]	.958 [.933, .979]	−1.7	.982/.980	.441	.924
MBR PCA-4	.938 [.904, .967]	.929 [.896, .958]	−0.8	.963/.961	.114	.997
CIC PCA-4	.692 [.633, .746]	.704 [.646, .758]	+1.3	.780/.777	.111	.997
EMBER2018 PCA-4	.617 [.558, .671]	.613 [.558, .667]	−0.4	.741/.723	.111	.997

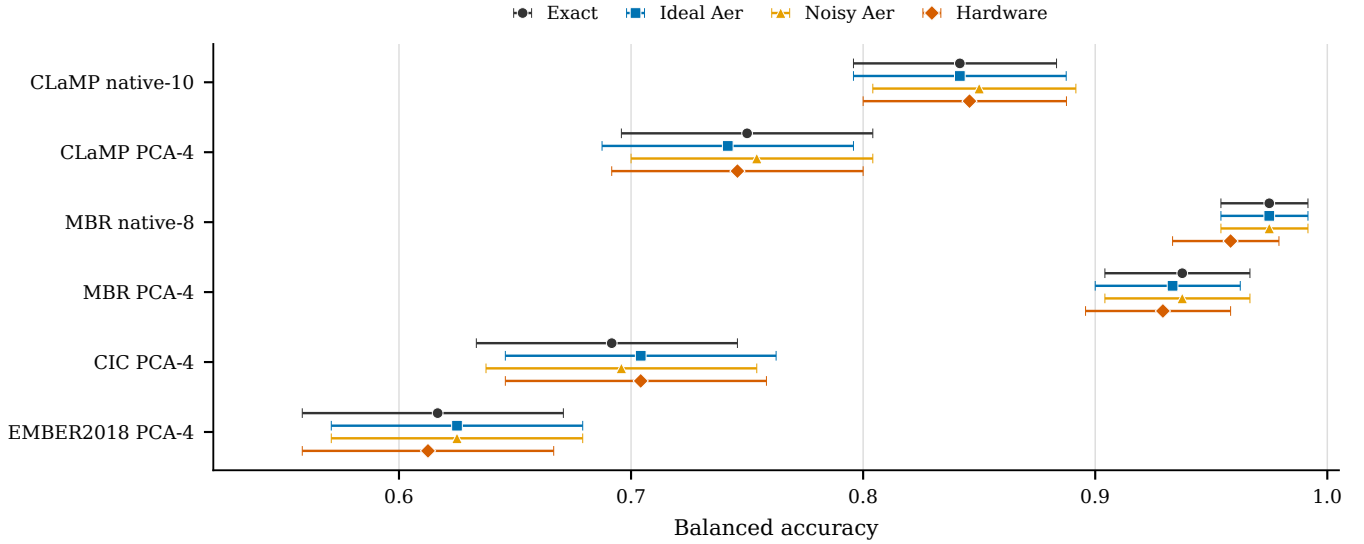


Figure 1: Balanced accuracy across exact, ideal-Aer, noisy-Aer, and IBM hardware execution. Points are means across three held-out splits; bars are 95% class-stratified bootstrap intervals based on 10,000 replicates and 240 held-out predictions per configuration and environment.

noise and device error. In particular, despite the substantial kernel error, especially for the wider circuits, balanced accuracy changed by less than two percentage points. The paired tests found no significant hardware-versus-exact prediction differences after multiplicity adjustment. These findings support empirical stability for the tested samples, feature map, shot count, backend, and calibration period, but they do not establish formal equivalence between exact and hardware execution.

We did not find kernel fidelity and classification performance to be interchangeable measures in this experiment. For instance, CZ count was strongly associated with kernel distortion but was not associated with a decrease in accuracy. However, because CZ count co-varied with circuit width, depth, and size, this analysis cannot identify CZ gates as the cause of the distortion. The hardware kernels were still able to retain enough relative structure compared with the exact kernels for the SVC to produce similar results. This observation, however, is specific to our tests; greater distortion could have different effects on other datasets, feature maps, regularization values, or sets of support vectors.

Further, applying four-component PCA reduced qubit

count, circuit depth, CZ count, and hardware-induced kernel distortion. This resource reduction corresponded to a loss of predictive information in some configurations, with a substantial loss for CLaMP and a smaller loss for MBR. PCA should therefore not be treated solely as a noise-reduction step; its resource benefit must be evaluated against representation loss for each dataset.

The classical comparisons did not demonstrate a consistent accuracy advantage for the quantum kernel. RBF SVC was competitive with or better than the hardware-quantum model on CLaMP, CIC, and EMBER2018, while the quantum models were descriptively higher on MBR. These comparisons used fixed $C = 1$ models without hyperparameter tuning to provide matched comparisons rather than optimized classical benchmarks. Because we did not collect comparable wall-clock time or system-level energy measurements, the results do not demonstrate computational or energy-based quantum advantage.

These findings are limited to balanced samples of 80 training and 80 test observations across three repeated holdouts, one feature-map architecture, one shot count, one backend and calibration period, and only two native-versus-PCA com-

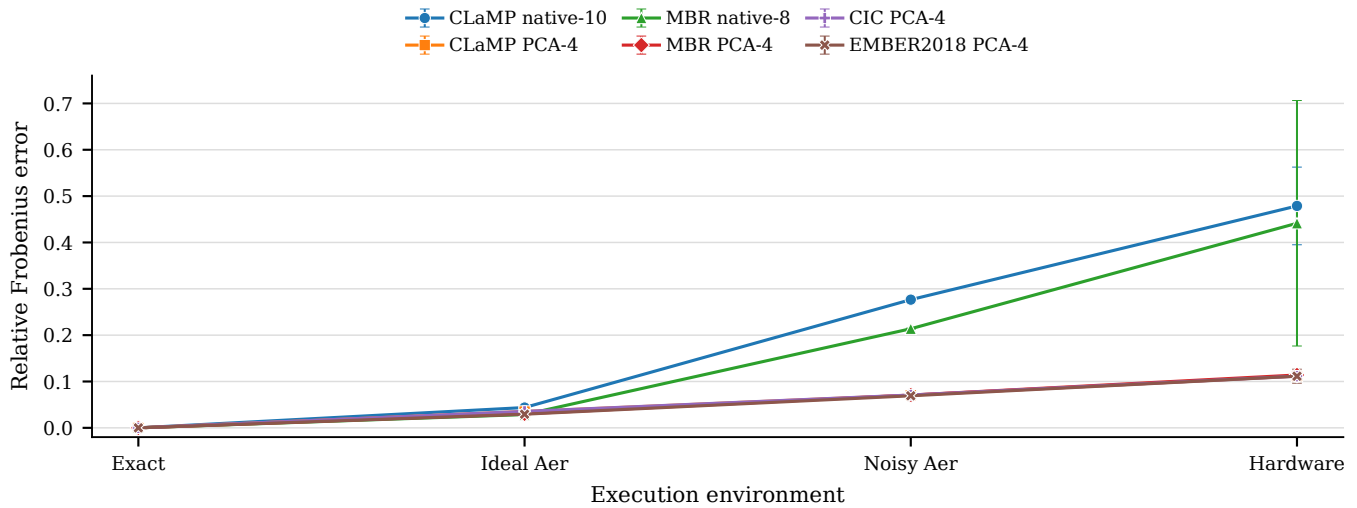


Figure 2: Test-kernel relative Frobenius error against the exact statevector kernel. Markers are three-split means and error bars are split standard deviations. Exact kernels define the zero-error reference.

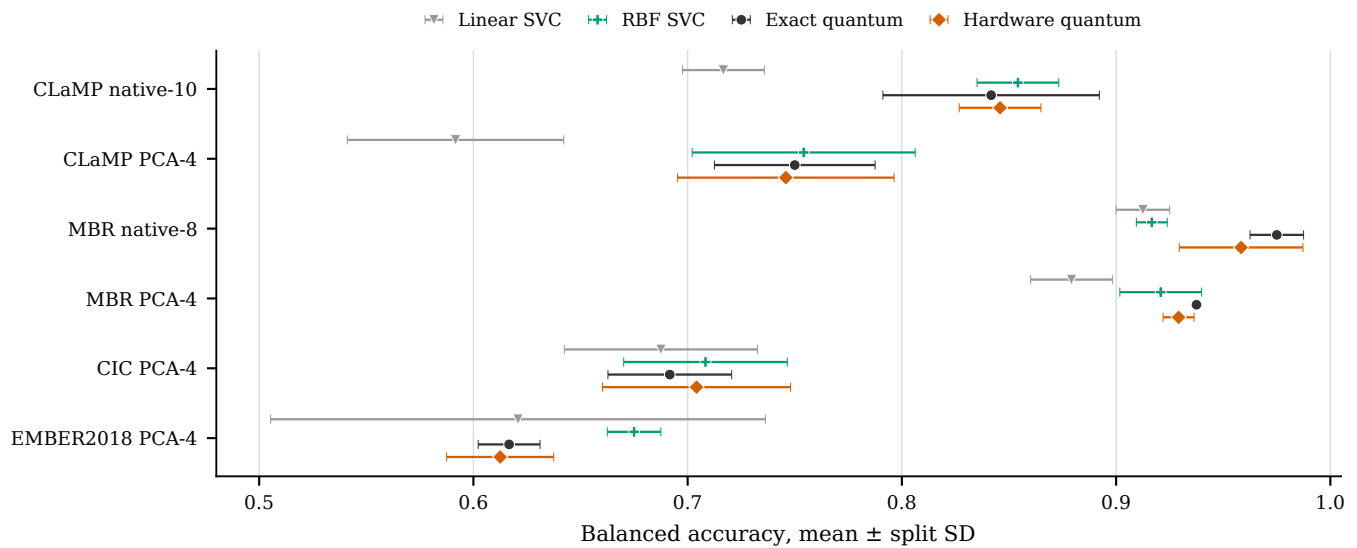


Figure 3: Matched linear-SVC, RBF-SVC, exact-quantum, and hardware-quantum balanced accuracy. Points are three-split means and bars are split standard deviations. Every classifier used $C = 1$ without hyperparameter tuning.

parisons. The classical models used fixed $C = 1$ without tuning, and the study did not measure comparable wall-clock time or system-level energy use.

Conclusion

Ultimately, this study provides evidence for the feasibility of quantum kernels, under NISQ conditions, for use in malware detection. The core principle we evaluated was whether, after applying PCA and performing feature reduction, quantum kernels could preserve learnability. The PCA-reduced representations retained predictive signal across the evaluated datasets, although predictive performance was dataset dependent. The deviation caused by device noise was small in downstream balanced accuracy compared with the numerical distortion of the kernels under the tested conditions.

In malware detection, PCA can substantially reduce the number of features without resulting in a complete loss of learning. However, PCA was not cost-free: it reduced circuit requirements and hardware kernel distortion but caused a substantial predictive loss for CLaMP and a smaller loss for MBR.

This study compared classical SVCs with exact, ideal Qiskit Aer, noisy Qiskit Aer, and IBM-hardware quantum kernels across four malware datasets. IBM hardware increasingly distorted the measured kernels as circuit resource requirements increased, but downstream balanced accuracy remained close to exact-kernel performance in all six configurations. PCA reduced circuit cost and hardware kernel distortion while causing dataset-dependent losses in predictive performance. Classical RBF models remained competitive, and the experiment did not demonstrate quantum advantage in accuracy, speed, or energy use. The results instead provide reproducible small-sample evidence that quantum-kernel malware classification can remain stable on current NISQ hardware despite substantial kernel distortion under the tested conditions.

Future Work Future evaluations should use larger samples and further vary the tested configurations. Both the classical and quantum models may be improved through model tuning. Another useful comparison would be between this quantum support vector machine approach and a trainable quantum neural network to test whether a learned representation improves predictive performance and exhibits the same sensitivity to hardware error.

Another idea that emerged from this study is to use quantum-high-performance computing (quantum-HPC) methods to evaluate quantum-kernel scalability. For example, quantum-kernel evaluation could be distributed across multiple quantum resources by parallelizing the independent pairwise circuit evaluations required to construct the kernel matrix. Such methods remained outside the scope of this study.

Acknowledgments

We would like to thank the Monmouth University School of Science Summer Research Program for providing funding for this project. We would also like to thank RPI; this research

was conducted using their on-campus IBM Quantum System One.

References

- Aashish.zzz. 2026. Malware Detection (dataset with new malware info). <https://www.kaggle.com/datasets/aashishzzz/malware-detection-dataset-with-new-malware-info>.
- Anderson, H. S.; and Roth, P. 2018. EMBER: An Open Dataset for Training Static PE Malware Machine Learning Models. arXiv:1804.04637.
- Barrué, G.; and Quertier, T. 2025. Quantum Machine Learning for Malware Classification. In Meo, R.; and Silvestri, F., eds., *Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, 245–260. Cham: Springer Nature Switzerland. ISBN 978-3-031-74643-7. https://doi.org/10.1007/978-3-031-74643-7_19.
- Ciamarella, G.; Iadarola, G.; Mercaldo, F.; Storto, M.; Santone, A.; and Martinelli, F. 2022. Introducing Quantum Computing in Mobile Malware Detection. In *Proceedings of the 17th International Conference on Availability, Reliability and Security*, ARES '22. New York, NY, USA: Association for Computing Machinery. ISBN 9781450396707. <https://doi.org/10.1145/3538969.3543816>.
- Cogburn, C. V.; DeCusatis, C.; and Spillane, E. 2026. Hardware-Aware QAOA for Honeypot Traffic Partitioning on 100+ Qubit IBM Quantum Processors. arXiv:2606.09469.
- Cohen, F. 1987. Computer viruses: theory and experiments. *Computers & Security*, 6(1): 22–35. [https://doi.org/10.1016/0167-4048\(87\)90122-2](https://doi.org/10.1016/0167-4048(87)90122-2).
- Cotrupi, M. L.; and Callahan, B. R. 2025. Towards Practical Quantum Kernels for Network Intrusion Detection. *Proceedings of the AAAI Symposium Series*, 7(1): 339–342. <https://doi.org/10.1609/aaais.v7i1.36903>.
- Dadkhah, S.; Okey, O. D.; Maret, S. A.; Lo, Y.-W.; Firouzi, A.; Kuki, R.; Sasaki, T.; Yoshioka, K.; Ban, T.; Ozawa, S.; and Ghorbani, A. A. 2026. CIC-YNU-IoTMal: A comprehensive multilayer dataset for static and dynamic analysis of IoT malware behavior. *Information Systems*, 140: 102722. <https://doi.org/10.1016/j.is.2026.102722>.
- Golub, G. H.; and Van Loan, C. F. 2013. *Matrix Computations*. Baltimore, MD: Johns Hopkins University Press, 4 edition.
- Holm, S. 1979. A Simple Sequentially Rejective Multiple Test Procedure. *Scandinavian Journal of Statistics*, 6(2): 65–70.
- Javadi-Abhari, A.; Treinish, M.; Krsulich, K.; Wood, C. J.; Lishman, J.; Gacon, J.; Martiel, S.; Nation, P. D.; Bishop, L. S.; Cross, A. W.; Johnson, B. R.; and Gambetta, J. M. 2024. Quantum computing with Qiskit. arXiv:2405.08810.
- Joshi, T.; and Guha, K. 2025. Quantum AI Algorithm Development for Enhanced Cybersecurity: A Hybrid Approach to Malware Detection. arXiv:2509.05370.
- Kumar, A. 2020. CLaMP (Classification of Malware with PE Headers). <https://doi.org/10.17632/xvyv59vwvz.1>.

- Lopez, J.; Nowmi, S. R.; Cadena, V.; and Rahman, M. S. 2025. Towards Quantum Machine Learning for Malicious Code Analysis. In *2025 IEEE International Conference on Quantum Computing and Engineering (QCE)*, volume 02, 211–216. <https://doi.org/10.1109/QCE65121.2025.10321>.
- McNemar, Q. 1947. Note on the Sampling Error of the Difference Between Correlated Proportions or Percentages. *Psychometrika*, 12(2): 153–157. <https://doi.org/10.1007/BF02295996>.
- Mercaldo, F.; Ciaramella, G.; Iadarola, G.; Storto, M.; Martinelli, F.; and Santone, A. 2022. Towards explainable quantum machine learning for mobile malware detection and classification. *Applied Sciences*, 12(23): 12025. <https://doi.org/10.3390/app122312025>.
- Pachhala, N.; Jothilakshmi, S.; and Battula, B. P. 2021. A Comprehensive Survey on Identification of Malware Types and Malware Classification Using Machine Learning Techniques. In *2021 2nd International Conference on Smart Electronics and Communication (ICOSEC)*, 1207–1214. <https://doi.org/10.1109/ICOSEC51865.2021.9591763>.
- Passo, S. J. G.; Kothavade, V. H.; and Prevost, J. J. 2025. Supply Chain Malware Detection via Classical and Quantum Kernel Methods in Embedded Systems. In *2025 55th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*, 208–215. <https://doi.org/10.1109/DSN-W65791.2025.00063>.
- Pearson, K. 1895. VII. Note on Regression and Inheritance in the Case of Two Parents. *Proceedings of the Royal Society of London*, 58(347–352): 240–242.
- Schölkopf, B.; and Smola, A. J. 2002. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Cambridge, MA: MIT Press.
- Ucci, D.; Aniello, L.; and Baldoni, R. 2019. Survey of machine learning techniques for malware analysis. *Computers & Security*, 81: 123–147. <https://doi.org/10.1016/j.cose.2018.11.001>.
- Vyas, A.; Lo, D.; and Zhang, X. 2025. A Comparison of Classical and Quantum Neural Network Algorithms for Malware Detection. In *Proceedings of the 2025 ACM Southeast Conference, ACMSE 2025*, 134–144. New York, NY, USA: Association for Computing Machinery. ISBN 9798400712777. <https://doi.org/10.1145/3696673.3723063>.