Leveraging Quantum Machine Learning for Intrusion Detection in Software-Defined Networks

Joan Lo Anguera
José Antonio Lázaro
ORCID: 0000-0002-5592-8434
Javier Ruiz-Hidalgo
ORCID: 0000-0001-6774-685X
Àlex Solé
ORCID: 000-0002-2071-9317
Josep Ramon Casas
ORCID: 0000-0003-4639-6904
Signal Theory and Comm. Dept.
UPC - Univ. Politècnica de Catalunya
Barcelona, Spain
joan.anguera@upc.edu
jose.antonio.lazaro@upc.edu

Samael Sarmiento
LuxQuanta Technologies S.L.
Barcelona, Spain
ORCID: 0000-0001-6549-8426

Adolfo Lerín
CognitIAs, Spain
Madrid, Spain
ORCID: 0000-0003-2612-9956

***Abstract*—Quantum machine learning (QML) algorithms for intrusion detection systems in software-defined networks are investigated, and their effectiveness is compared with their classical machine learning methods. The University of Nevada - Reno intrusion detection dataset (UNR-IDD) is used to evaluate different QML models, including quantum k-nearest neighbors (QKNNs), quantum support vector machines (QSVMs), quantum neural networks (QNNs), and hybrid quantum neural networks (HQNNs). These models were tested with quantum simulators to evaluate their potential advantages in processing complex datasets. The results show that HQNN and QSVM have higher accuracy than their classical SVM and NN counterparts. This study shows the potential of leveraging QML to enhance precision. References to other works that dive into efficiency and complexity are included.**

Keywords— Quantum machine learning (QML), Intrusion Detection, Cybersecurity, Software-Defined Networks (SDN).

I. Introduction

Quantum computing [1] and machine learning [2] are changing modern computing. On the one hand, since qubits can exist in multiple states simultaneously, quantum computers can process information in a way that their classical counterparts cannot. A clear example of this difference is searching in a large database. While a classical computer would examine each entry individually, a quantum computer can use Grover's algorithm [3] to examine multiple entries simultaneously, making the searching task considerably faster. On the other hand, machine learning, which is usually used to make predictions and decisions based on vast amounts of data, is reaching the limits of classical computing. To address this challenge, quantum machine learning (QML) [4] (QML) has emerged as a promising approach that has the potential to benefit several industries, including the telecommunications sector.

As telecommunications networks operate as a shared medium, ensuring robust security is essential to protect personal data and maintain user trust. Network security can be significantly improved by integrating QML with modern network management models, such as software-defined networking (SDN) [5]. For example, using quantum-based intrusion detection systems (IDSs) along with SDN will ensure that QML-powered IDS can be dynamically reconfigured based on the location and intensity of an attack to ensure optimal coverage.

This article has been partially financed by the Smart Networks and Services Joint Undertaking (SNS JU) under the European Union's Horizon Europe research and innovation program Grant Agreement No. 101139182, and Spanish MICIU founded, TRAINER-B (PID2020-118011GB-C22).

In this paper, we investigate the application of QML to network intrusion detection systems (NIDSs) in SDN-based networks and compare the results with classical machine learning (ML) algorithms using a public database.

A. Software Defined Networking (SDN)

Software-defined networking (SDN) separates the network control (control plane) from communication and routing (data plane), marking a shift from monolithic and static network architectures [5]. This separation allows independent release cycles for the two planes but requires standardized interfaces between them.

SDN supports centralized control with full visibility of network resources, improving management, maintenance, and automation. Additionally, SDN enables flexible control architectures, including hierarchical and distributed models.

B. Network Intrusion Detection Systems (NIDSs)

Network intrusion detection systems (NIDSs) are essential for monitoring and analyzing network traffic to identify malicious activities or policy violations. These systems are classified as signature-based, which use predefined attack patterns, or anomaly-based, which apply machine learning to detect deviations from normal network behavior.

The University of Nevada - Reno intrusion detection dataset (UNR-IDD) addresses the limitations of existing NIDS datasets by focusing on network port statistics, offering finer analysis and faster intrusion detection. It supports both binary and multiclass classification of common cyber-attacks and has demonstrated accuracy comparable to larger datasets while significantly reducing training times [6].

C. Classical and Quantum ML Models for NIDSs

Machine learning (ML) algorithms are integral to data-driven research, enabling classification, regression, and pattern recognition. Common methods include random forests (RFs), k-nearest neighbors (KNNs), support vector machines (SVMs), and artificial neural networks (ANNs).

RFs are ensemble learning methods that build multiple decision trees and combine their predictions to avoid overfitting and improve overall performance [7]. In KNNs, a non-parametric algorithm makes predictions based on the k nearest training samples in the feature space, and uses proximity to infer the output [8]. SVMs are supervised learning models that find the hyperplane that best separates classes with maximum margin and use kernel functions to handle non-linear boundaries [9].

Finally, ANNs are computational models inspired by the architecture of biological neural networks [10]. They consist of interconnected neurons that learn by adjusting weights using backpropagation [11].

In recent years, it has become more than obvious that classical NIDSs will not be sufficient in the short and medium term to meet the challenges facing computer systems. Here, QML algorithms are promising. These algorithms use quantum mechanical principles to overcome the limitations of classical solutions, both in the identification of intrusion events and in their classification. In particular, QML algorithms have the potential to improve NIDSs due to their ability to process massive data in an efficient way, using superposition and quantum entanglement principles, with a clear advantage in terms of efficiency and scalability compared to classical solutions [12].

On the other hand, recent research [13] has shown that the use of quantum models to identify distributed denial of service (DDoS) attacks can achieve success rates above 96%. Other recent research [14], proposed other alternatives based on quantum support vector machines (QSVMs) to detect DDoS attacks in smart microgrids, and in all cases showed better performance than their classical SVM counterparts.

Just as neural networks are used for classical applications to identify behavioral patterns, the use of quantum convolutional neural networks (QCNNs), as well as the use of variational quantum circuits (VQCs) and hybrid quantum-classical models, are being considered to improve the performance of attack identification and detection using quantum principles [15]. This work is promising in the short term both for detection and multiclass classification of intrusion events or attacks.

While it is true that QML based NIDSs are still at an early stage of research, the studies conducted so far show high potential for improving cybersecurity in the quantum era. However, such solutions require more research and parallel development of quantum hardware [16] to efficiently exploit the principles on which they are based, especially considering that threats and attack techniques evolve in parallel [17].

II. Exploratory Data Analysis of the UNR-IDD

The UNR-IDD was developed using Mininet software, a network function virtualization (NFV) environment to create an SDN topology consisting of 10 hosts and 12 Open vSwitches. For feature extraction, a custom implementation of the open network operating system (ONOS) SDN controller was used to measure network traffic. IPerf software was used to generate 10 Mbps TCP and UDP data streams between randomly chosen source-destination pairs every 5 seconds.

The features were collected using *OFPPortStatsRequest* and *OFPPortStatsReply* messages between the SDN controller and Open vSwitches, for a total number of 34 features as found in [6]. In particular, the *Label* distinguishes between different network attacks: *TCP-SYN flood, Blackhole, Port Scan, Flow table overflow,* and *Diversion.* The *Binary Label* indicates whether the labels are *Normal* or *Attack* network traffic.

A. Selection of Features and Number of Samples

A crucial step in preprocessing is to identify possible blank spots among the data, or highly correlated samples to reduce the total number of features. Fortunately, this dataset contains no blank samples and has only a single duplicate that can be easily resolved. Class labels comprise about 90% of *Attack* and 10% *Normal* for both binary and multiclass scenarios, leading to a relevant class imbalance that needs to be considered when benchmarking metrics, especially in the binary case.

The total size of the dataset is 37412 samples with 34 features each. The amount of computation time can be very disadvantageous if the number of features and samples is high, especially for quantum simulators. A balance must therefore be found between the minimum number of features, the number of samples, and the accuracy of the models. For this purpose, a statistical analysis of the features is performed, which leads to the removal of: *Switch ID, Port Number, Packets Rx Dropped, Packets Tx Dropped, Packets Rx Errors, Packets Tx Errors, Delta Packets Rx Dropped, Delta Packets Tx Dropped, Delta Packets Rx Errors, Delta Packets Tx Errors, is_valid, Table ID,* and *Max Size*, as they have standard deviation and percentiles of zero. Next, the importance scores of the remaining 19 features were calculated using a RF-based algorithm to further reduce their dimensionality. All possible combinations of 4 elements without repetitions are performed for the 10 most important features. The importance of the features indicates that the remaining ones provide little additional information. Selecting four features is based on the dimensionality of quantum state vectors, which follow powers of 2. To identify the best combination of features, RFs, KNNs, SVMs, and NNs are used to calculate accuracies, with an 80-20% train-test split. The combination that achieves the highest accuracy across these models is chosen. Then, the dataset size is adjusted from 500 to 30000 samples (in steps of 500), and the accuracies are recalculated to determine the optimal sample size for the selected feature combination.

III. Application of QML to the UNR-IDD NIDS

This section focus on the application of QML algorithms to the NIDS task using the UNR-IDD, and systematically explore the quantum implementations of: QKNN-V1, QKNN-V2, QKNN-V3, QSVM, QNN, and HQNN. For all QML algorithms except for QKNN-V3, the dataset contained 10000 samples of 4 features corresponding to the best feature combination obtained in the previously performed exploratory data analysis. For QKNN-V3, however, the number of samples was limited to 200 for reasons of computational cost. Qiskit library [18] was chosen to implement the QML models except for QSVM, which was programmed using PennyLane [19] due to its more convenient approach.

A. Quantum k-Nearest Neighbors (QKNNs)

Three QKNN model versions have been implemented: QKNN-V1, QKNN-V2 and QKNN-V3. QKNN-V1 serves as a foundational quantum approach for KNNs using controlled-SWAP gates to compute Euclidean quantum distances. Amplitude and angle encoding are used to transform classical data into quantum states. QKNN-V2 introduces an adaptation

of the circuit of [20] that includes additional quantum operations to improve performance and accuracy. This model uses angle encoding. Finally, QKNN-V3, an implementation of the QKNN algorithm presented by [21], incorporates more resource-intensive quantum circuit designs and encoding strategies. To calculate Euclidean quantum distances, the fidelity equation $F = |\langle\psi|\phi\rangle|^2$ is used, where ψ and ϕ are the train and test quantum states respectively. Both QKNN-V1 and QKNN-V2 use the `sklearn.neighbors` library to fit and train the model with `KNeighborsClassifier()`, using quantum inner products as the distance metric, while QKNN-V3 processes and indexes quantum states with a quantum circuit and an oracle.

1) QKNN-V1

To compute quantum inner products, the circuit shown in Fig. 2 of [20] was run for 2048 realizations. A measurement is performed on the first ancilla qubit when its probability is 0 in the form of $P(0) = \frac{1}{2} + \frac{1}{2}|\langle\psi|\phi\rangle|^2$. For amplitude encoding, the resulting distance is obtained using $D = 4Z(P(0) - 0.5)$, where $Z = |a|^2 + |b|^2$. The classic data vectors correspond to $a = (a_1, a_2, \cdots, a_n)$ and $b = (b_1, b_2, \cdots, b_n)$, which are not necessarily normalized, and are transformed into the states $|\psi\rangle = \frac{1}{\sqrt{2}}(|0\rangle|a\rangle + |1\rangle|b\rangle)$ and $|\phi\rangle = \frac{1}{\sqrt{2}}(|a||0\rangle - |b||1\rangle)$. For the case of angle encoding, 2D unitary single-qubit transformations are applied to the quantum states in the form of $|\psi\rangle = U(a_1', a_2')|0\rangle$ and $|\phi\rangle = U(b_1', b_2')|0\rangle$, where U is

$$U(\theta, \gamma) = \begin{pmatrix} cos\frac{\theta}{2} & -sin\frac{\theta}{2} \\ e^{i\gamma}sin\frac{\theta}{2} & e^{i\gamma}cos\frac{\theta}{2} \end{pmatrix} \quad (1)$$

with $a_1' = \frac{7\pi}{2}(a_1 + 1)$ and $a_2' = \frac{7\pi}{2}(a_2 + 1)$ (same for b). The factor $7\pi/2$ may vary depending on the underlying dataset characteristics. In this approach the two other features indexed as $i = 3, 4$ are not considered since in this case the number of features is $n = 4$. Other techniques such as considering the mean $a_1 = \bar{a}_{1,2}$ and $a_2 = \bar{a}_{3,4}$ for both a and b were tested but yielded slightly worse results. Finally, the metric distance is $D = \sqrt{Z \cdot P(1)}$, where the probability of the ancilla qubit to be 1 is $P(1) = \frac{1}{2} - \frac{1}{2}|\langle\psi|\phi\rangle|^2$.

2) QKNN-V2

This adaptation extends the previous QKNN-V1 angle encoding implementation by applying unitary U 2D rotations (1) as $|\psi\rangle = \bigotimes_{i \in odd(n)} U(a_i', a_{i+1}')|0\rangle \otimes U(a_n', a_n')$ and $|\phi\rangle = \bigotimes_{i \in odd(n)} U(b_i', b_{i+1}')|0\rangle \otimes U(b_n', b_n')$. For this configuration, n must be an even number, matching the lengths of a and b. Here, $odd(n)$ means all odd indices from 1 to n. If feature vectors have odd dimensions, they can be zero-padded to make them even. Fig. 1 shows the functional quantum circuit constructed with Hadamard gates, single-qubit rotations, controlled-SWAP gates applied to the respective pairs of indices, and the measurement of the first ancilla qubit when its probability is 1. Again, the circuit was simulated for 2048 realizations with the final distance metric as $D = \sqrt{Z \cdot P(1)}$, where $Z = |a|^2 + |b|^2$. In addition, the angle factor for a_i' and b_i' was chosen to be $\pi/2$.

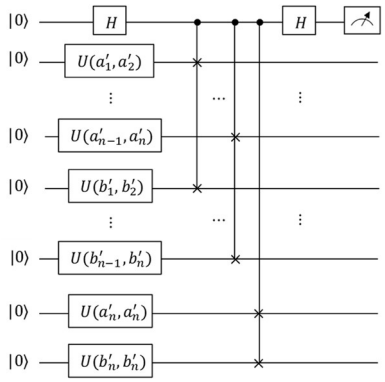

Fig. 1. QKNN-V2 quantum circuit.

3) QKNN-V3

Finally, QKNN-V3 adaptation was performed based on the algorithm by [21]. It aims to compute all distances with respect to their nearest neighbors simultaneously, using the previously defined fidelity distance F. It is implemented according to the scheme shown in Fig. 4 in [21]. The detailed quantum circuit requires an oracle $\mathcal{W}$ capable of generating these states in superposition. We have implemented an oracle in the form of $\mathcal{W}|0\rangle|i\rangle = |\phi_i\rangle|i\rangle$, where $|\phi_i\rangle$ are the indexed train states. The Grover's algorithm, which is not shown in cited Fig. 4, is included before the measurements to determine the nearest neighbors. Finally, the measurement of similarity, test and train registers is performed to compute the class label for the test state.

B. Quantum Support Vector Machines (QSVMs)

Next, we present a QSVM implementation that uses a quantum circuit to perform a kernel function. The PennyLane library was used to build the quantum circuit shown in Fig. 2, where a quantum feature map $x \rightarrow |\phi(x)\rangle$ is represented by a quantum circuit $|\phi(x)\rangle = U(x)|0^n\rangle$. Each kernel entry $K(x_i, x_j)$ is calculated by running the circuit $U^\dagger(x_j)U(x_i)$ on the input state $|0^n\rangle$ followed by estimating $|\langle 0^n|U^\dagger(x_j)U(x_i)|0^n\rangle|^2$ through the frequency of observing the outcome $|0^n\rangle$. The quantum circuit $U(x_i)$ is built by implementing a slightly modified `ZZFeatureMap` with X and RZ gates, as described in the Qiskit library [18]. This process is equivalent to calculating the inner products for each kernel entry, as represented by the Gram matrix in (2).

$$G = \begin{bmatrix} \phi(x_1)^T\phi(x_1) & \dots & \phi(x_1)^T\phi(x_M) \\ \vdots & \ddots & \vdots \\ \phi(x_M)^T\phi(x_1) & \cdots & \phi(x_M)^T\phi(x_M) \end{bmatrix} \quad (2)$$

Here, x_i and x_j represent the combinations of either training or test feature vectors. Initially, the training Gram matrix is computed, where both x_i and x_j are training samples. This is used to minimize the objective function $L(\alpha) = \sum_{j=1}^{M} y_j\alpha_j - \frac{1}{2}\sum_{j,k=1}^{M}\alpha_j K_{jk}\alpha_k$ [22] with the corresponding training labels. Subsequently, a test Gram matrix is constructed using the inner cross products of x_i (test samples) and x_j (training samples) to recover the hyperplane from the α coefficients and predict the test labels. Finally, a classical SVM is executed using these results to

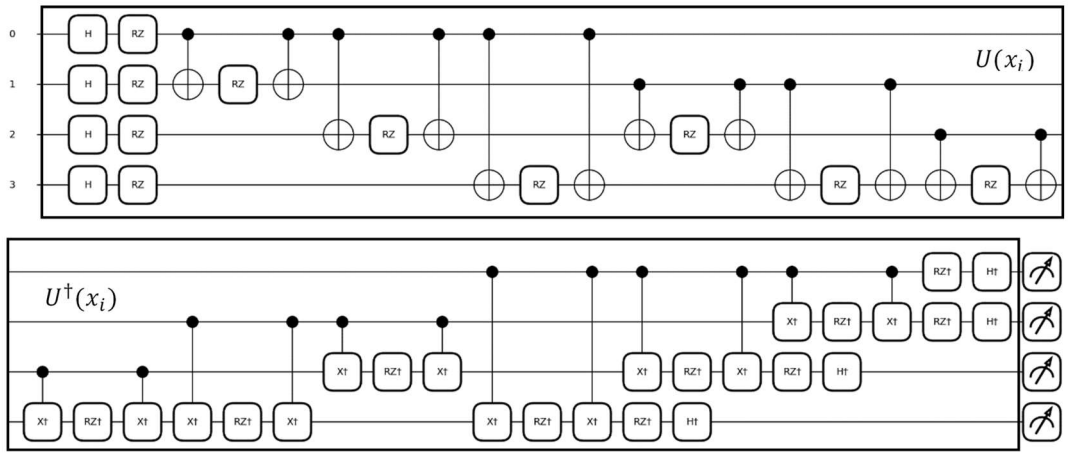

Fig. 2. Custom version of ZZFeatureMap to compute QSVM quantum kernel function.

utilize the precomputed quantum kernel. Since the number of shots in the execution of the quantum circuit was not specified, PennyLane performed an analytical execution.

C. Quantum Neural Networks (QNNs)

QNNs act as variational quantum algorithms, where classical data is encoded into quantum states using a feature map $\mathcal{U}_x$ in the form of $|\psi_x\rangle = \mathcal{U}_x|0\rangle^{\otimes S}$, within a S-qubit Hilbert space, to be later on processed by a quantum variational circuit $\mathcal{G}_\theta$ as $|g_\theta(x)\rangle = \mathcal{G}_\theta|\psi_x\rangle$, as Fig. 3 illustrates. The parameters of the variational circuit (or ansatz), are trained and updated by minimizing an objective function. During this minimization, the output of the quantum model $z = (z_1, \dots, z_S)$ is derived from a classical post-processing function applied to a measurement result $y = f(z)$.

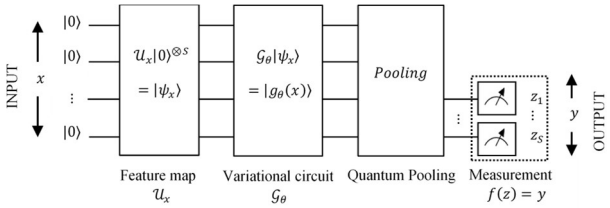

Fig. 3. Quantum neural network architecture.

To adjust the number of qubits to the output measurement, a quantum pooling layer is applied immediately after the ansatz. This quantum pooling circuit consists of individual two-qubit pooling circuits, as shown in Fig. 4.

Two combinations of feature maps and ansatzes were tested using Qiskit library [18]. The first combination paired a `ZZFeatureMap` with an `EfficientSU2` ansatz (Fig. 5), and the second used a `PauliFeatureMap` with a `RealAmplitudes` ansatz (Fig. 6), both modified with four repetitions. Quantum pooling reduced qubits to 1 and 3 for binary and multiclass scenarios, respectively. The `SamplerQNN` instance was used to train the model with the COBYLA optimizer for 600 iterations. The parameter-shift rule was used to compute quantum backpropagation, along with L2 and cross-entropy objective functions for binary and multiclass cases, respectively.

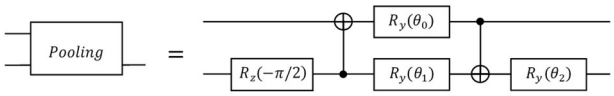

Fig. 4. Quantum pooling circuit.

D. Hybrid Quantum-Classical Neural Networks (HQNNs)

Hybrid Quantum Neural Networks (HQNNs) integrate feature maps, quantum variational circuits, and classical neuron layers. The structure employs two combinations of feature maps, ansatz, and quantum pooling layers. Outputs from the quantum components are measured and integrated into a classical feedforward neural network using Qiskit's `TorchConnector`, as shown in Fig. 7.

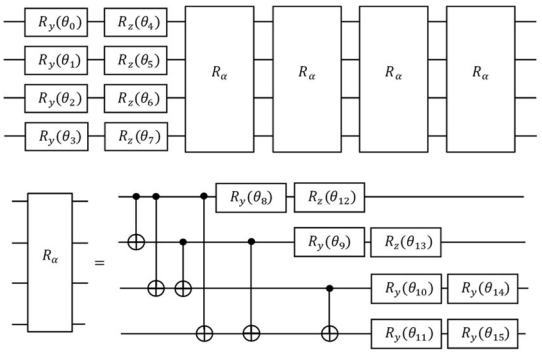

Fig. 5. EfficientSU2 variational quantum circuit with 4 repetitions.

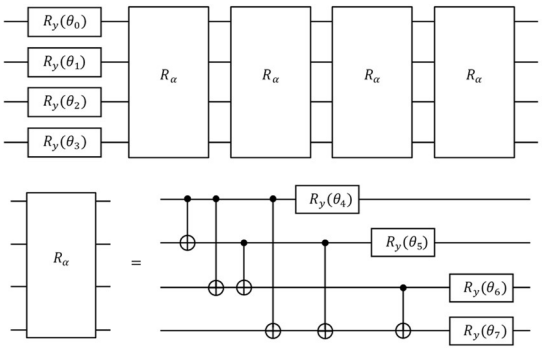

Fig. 6. RealAmplitudes variational quantum circuit with 4 repetitions.

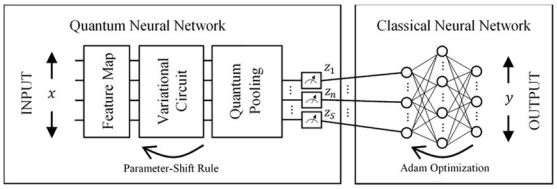

Fig. 7. Hybrid quantum-classical neural network scheme.

The classical network consists of two hidden layers with dropout, batch normalization, and Leaky ReLU activations. Training involves 50 epochs using Adam optimization for classical layers and the parameter-shift rule for quantum layers. For binary and multiclass classification tasks, the quantum neural network (QNN) uses L2 and cross-entropy objective functions, while HQNN employs binary and standard cross-entropy, respectively.

IV. Classical Model Results

Fig. 8 illustrates the accuracy results for classical models based on four-element feature combinations using the entire dataset. The accuracy decreases as the combinations progress to the right, reflecting the descending order of feature importance. The optimal feature combination was derived from the multiclass scenario, which involved higher complexity. The selected features are: *Port Alive Duration (S), Packets Matched, Packets Looked Up,* and *Active Flow Entries*. These four features, were then used to train the QML algorithms on a subset of 10000 samples with the previous proportions of 80% and 20% for train and test respectively. Fig. 9 shows the respective accuracies of the features.

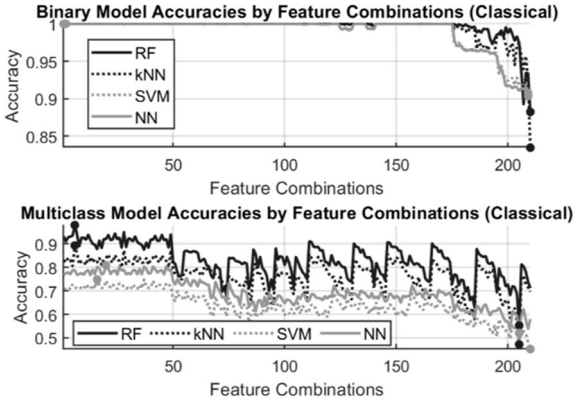

Fig. 8. Binary and multiclass classical accuracies as a function of four-element feature combinations.

Table I. Classic Model Comparison for 10000 Samples (80% Train and 20% Test).

Classical Models	Classification Type	Accuracy	$\overline{F1}$ score
RF	***Binary***	100%	100%
	Multiclass	87.35%	87.71%
KNN	***Binary***	100%	100%
	Multiclass	81.45%	89.35%
SVM	***Binary***	100%	100%
	Multiclass	69.65%	67.73%
NN	***Binary***	99.50%	98.64%
	Multiclass	77.10%	73.25%

Mean $F1$ score ($\overline{F1}$) is included to account for the significant class imbalance. For this, the dataset was shuffled between runs to avoid training bias. The similarity between accuracies and $\overline{F1}$ suggests that the models achieve a good balance between precision and recall. No validation set was needed because models generalized well, saving additional hyperparameter tuning computation. Notably, the multiclass scenario presented greater classification challenges.

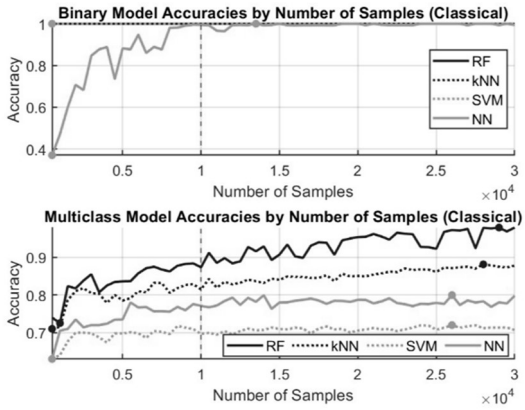

Fig. 9. Binary and multiclass classical accuracies for the best feature combination over the number of samples.

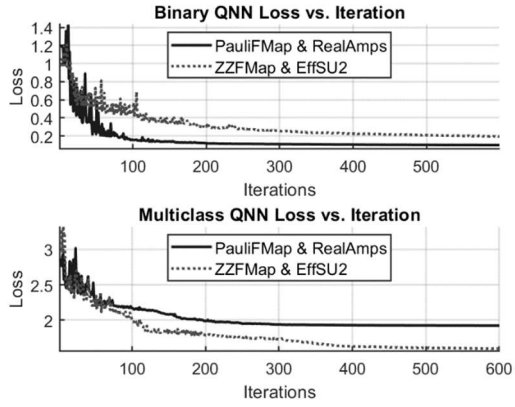

Fig. 10. Binary and multiclass quantum neural network training loss graphs.

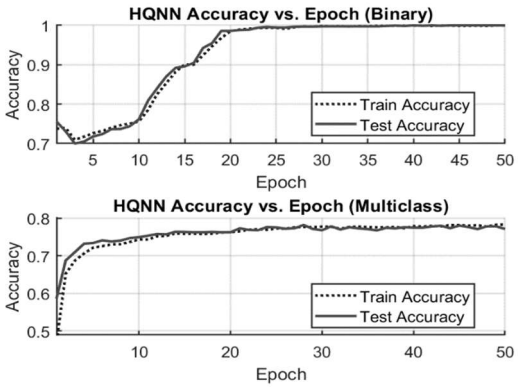

Fig. 11. Binary and multiclass hybrid quantum-classical neural network training graphs (PauliFeatureMap and RealAmplitudes combinations).

V. Quantum Model Results

Fig. 10 shows the evolution of QNN loss over training iterations. The combination of ZZFeatureMap and EfficientSU2 shows greater difficulty in reducing the loss compared to the PauliFeatureMap and RealAmplitudes combination for the binary case. Conversely, the opposite trend is observed for the multiclass task. Fig. 11 depicts the HQNN train and test accuracies. The first binary classification graph corresponds to the circuit combination of PauliFeatureMap and RealAmplitudes, whereas the multiclass graph contains the ZZFeatureMap and EfficientSU2 combination. The results align with the loss trends in Fig. 10.

An overview of all QML results can be found in Table II. In binary classification, several models achieve 100% accuracy: QKNN-V1, QKNN-V2, QSVM and HQNN. For the multiclass scenario, the best accuracy was achieved by HQNN with 78.24%, followed by QSVM with 72.40%, both overtaking classical ML results. It's noticeable that other models such as QKNN-V1 have limitations with only 46.80% and 19.80% for angle and amplitude encoding respectively. A further study focusing on computation time would require access to quantum computers. In this work, we have been using quantum simulators from IBM and PennyLane running on classical computers, which makes a comparison of computation time not fair and not representative. This paper doesn't include further research on the optimization of the QML algorithms, although other authors have shown significant efficiency improvements [23].

Table II. Quantum Model Comparison for 10000 Samples (80% Train and 20% Test), except 200 Samples for QKNN-V3.

Classical Models	Classification Type	Accuracy
QKNN-V1 (Ang.)	***Binary***	100%
	Multiclass	46.80%
QKNN-V1 (Amp.)	***Binary***	90.05%
	Multiclass	19.80%
QKNN-V2	***Binary***	100%
	Multiclass	56.90%
QKNN-V3[a]	***Binary***	85%
	Multiclass	55%
QSVM	***Binary***	100%
	Multiclass	72.40%
QNN	***Binary***	98.90%
	Multiclass	63.55%
HQNN	***Binary***	100%
	Multiclass	78.24%

VI. Conclusions

This work shows that QML algorithms can improve the results of classical ML. For example, the accuracy achieved with HQNN reaches 78.24%, surpassing the 77.10% achieved with classical NN. The accuracy of QSVM reaches 72.40% ahead of the 69.65% achieved by classical SVM, showing higher accuracies of HQNN and QSVM than their classical SVM and NN counterparts, suggesting the potential of QML to enhance precision in cybersecurity.

References

[1] S. K. Sood and Pooja, "Quantum Computing Review: A Decade of Research," in IEEE Transactions on Engineering Management, vol. 71, pp. 6662-6676, 2024, doi: 10.1109/TEM.2023.3284689.

[2] D. Patil, et al. "Machine learning and deep learning: Methods, techniques, applications, challenges, and future research opportunities," Trustworthy Artificial Intelligence in Industry and Society, pp 28-81, 2024

[3] P. J. Szablowski, "Understanding mathematics of Grover's algorithm," Quantum Information Processing, 20(5), 191, 2021.

[4] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, "Quantum machine learning," Nature 549, 195–202, 2017, https://doi.org/10.1038/nature23474.

[5] W. Xia, Y. Wen, C. H. Foh, D. Niyato and H. Xie, "A Survey on Software-Defined Networking," in IEEE Communications Surveys & Tutorials, vol. 17, no. 1, pp. 27-51, Firstquarter 2015, doi: 10.1109/COMST.2014.2330903.

[6] Das, T., Hamdan, O. A., Shukla, R. M., Sengupta, S. and Arslan, E., "UNR-IDD: Intrusion Detection Dataset using Network Port Statistics," 2023. IEEE 20th Consumer Communications & Networking Conference (CCNC), Las Vegas, NV, USA, 2023, pp. 497-500, doi: 10.1109/CCNC51644.2023.10059640.

[7] Breiman, L. (2001). Random forests. Machine Learning, 45(1), 5-32.

[8] Hand, D., Mannila, H., & Smyth, P. (2001). Principles of Data Mining. MIT Press.

[9] Cortes, C., & Vapnik, V. (1995). Support-vector networks. Machine Learning, 20(3), 273–297.

[10] Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. Psychological Review, 65(6), 386–408.

[11] Rumelhart, D. E., Hinton, G. E., Williams, R. J. (1986). Learning representations by back-prop. errors. Nature, 323(6088), 533–536.

[12] Abreu, D., Rothenberg, C. E., & Abelém, A. (2024, June). QML-IDS: Quantum Machine Learning Intrusion Detection System. In 2024 IEEE Symposium on Computers and Communications (ISCC) (pp. 1-6).

[13] Payares, E. D., and Martínez-Santos, J. C., 2021. Quantum machine learning for intrusion detection of distributed denial of service attacks: a comparative overview. Quantum Computing, Communication, and Simulation, 11699, 35-43.

[14] Said, D. (2023). Quantum Computing and Machine Learning for Cybersecurity: Distributed Denial of Service (DDoS) Attack Detection on Smart Micro-Grid. Energies, 16(8), 3572. https://doi.org/10.3390/en16083572

[15] Kim, T. H., and Madhavi, S. 2024. Quantum intrusion detection system using outlier analysis. Scientific Reports, 14(1), 27114. doi

[16] Cerezo, M., Verdon, G., Huang H.Y., Cincio, L., Coles, P.J., Challenges and opportunities in quantum machine learning. Nature Computation Science, 2, 567-576 (2022).

[17] Faker, O., and Cagiltay, N. E. 2023. Quantum Machine Learning in Intrusion Detection Systems: A Systematic Mapping Study. In International conference on WorldS4 (pp. 99-113). Singapore: Springer Nature Singapore.

[18] Qiskit. 2024. IBM Quantum Documentation. https://docs.quantum.ibm.com/. Accessed: 2024-22-11.

[19] PennyLane. 2024. Quantum Programming Software. https://pennylane.ai/. Accessed: 2024-22-11.

[20] Diadamo, S., O'Meara, C., Cortiana, G., & Bernab'e-Moreno, J. (2021). Practical Quantum K-Means Clustering: Performance Analysis and Applications in Energy Grid Classification. IEEE Transactions on Quantum Engineering, 3, 1-16.

[21] Basheer, A., Afham, A., & Goyal, S. K. (2020). Quantum k-nearest neighbor algorithm. arXiv:2003.09187.

[22] Rebentrost, P., Mohseni, M., & Lloyd, S. (2014). Quantum support vector machine for big data classification. Physical review letters, 113(13), 130503.

[23] T. T. An, S. L. Cotton, J. Zhang, Y. Ding, and T. Q. Duong, "LoRa Radio Frequency Fingerprinting Using a Hybrid Quantum-Classical Neural Network," in Proc. of IEEE Vehicular Technology Conference (VTC2024-Fall), Washington DC, Oct. 2024.