# Quantum Machine Learning Techniques for Network Intrusion Detection in Software-Defined Networks

Anonymous submission

**Abstract**

The application of quantum machine learning (QML) algorithms for intrusion detection systems in software-defined networks is investigated, and their effectiveness is compared with classical machine learning methods. The research utilized the UNR-IDD dataset 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, indicating the potential of QML to enhance precision and efficiency in cybersecurity.

## Introduction

Quantum computing (Sood 2024, Lu 2023) and machine learning (Patil et al. 2024, Amin et al. 2024) 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 a large database. While a classical computer would examine each entry individually, a quantum computer can use Grover's algorithm (Szablowski 2021) 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) (Biamonte et al. 2017) 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. By integrating QML with modern network management models, such as software-defined networking (SDN) (Nisar et al. 2020, Xia et al. 2015), network security can be significantly improved in several ways: (a) faster and more accurate real-time threat detection by combining the pattern analysis capabilities of QML with granular network data provided by SDN, (b) improved anomaly detection mechanisms through QML analyzing SDN-generated telemetry to identify new attack vectors and adapt to evolving threats, (c) faster encryption and decryption based on QML techniques and flexible SDN policy enforcement secure data with minimal latency, (d) quantum-based intrusion detection systems (IDSs) where SDN ensures that QML-powered IDS can be dynamically reconfigured based on the location and intensity of an attack to ensure optimal coverage, and (e) improved resilience against advanced persistent threats.

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.

## Software Defined Networking (SDN)

SDN is an approach for network control and management that relies on the separation between the network control (control plane) and the communication and routing process (data plane). This differentiation has a non-negligible impact that offers several advantages where some points become more relevant than in traditional approaches (Xia et al. 2015). This is a significant leap from previous networking paradigms that relied on monolithic and static architectures.

This decoupling between data and control planes separates the release cycles of the different elements that conform both planes. While this is beneficial, it requires agents for the data plane that implement commonly agreed interfaces with the control plane.

SDN enables the implementation of a centralized control that has entire visibility of the different network resources, which facilitates the management and maintenance and improves the automation of network control. Other control architectures are also possible under the SDN paradigm, enabling hierarchical and distributed control options.

## Network Intrusion Detection Systems (NIDSs)

Network intrusion detection systems (NIDSs) are critical components of modern cybersecurity infrastructure, designed to monitor and analyze network traffic for signs of

malicious activity or security policy violations. NIDSs are strategically placed in a network to examine data packets flowing to and from all devices and compare them to known attack signatures or detect anomalous behavior.

There are two main types of NIDSs: signature-based and anomaly-based. Signature-based NIDSs compare network traffic against preconfigured patterns of known attacks, while anomaly-based NIDSs use ML techniques to establish a baseline of normal network behavior and detect deviations from this norm.

The University of Nevada - Reno intrusion detection dataset (UNR-IDD) is a novel dataset developed to overcome the limitations of existing NIDS datasets (Dans et al. 2023). UNR-IDD focuses on network port statistics, providing finer analysis and potentially faster intrusion detection compared to datasets that mainly use flow-level statistics. UNR-IDD supports both binary and multiclass classification tasks and covers common cyber-attacks on network devices and end hosts. In performance evaluations, UNR-IDD achieved comparable accuracy to larger datasets while significantly reducing overall training times.

## Classical and Quantum ML Models for NIDSs

ML algorithms have become an integral part of data-driven research and provide robust solutions for classification, regression, and pattern recognition in various applications. Random forests (RFs), k-nearest neighbors (KNNs), support vector machines (SVMs), and artificial neural networks (ANNs) are some of the most widely used and established approaches in ML.

RFs are ensemble learning methods that build multiple decision trees and combine their predictions to avoid overfitting and improve overall performance (Breiman 2001).

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 (Hand et al. 2001).

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 (Cortes et al. 1995).

Finally, ANNs are computational models inspired by the architecture of biological neural networks (Rosenblatt 1958). They consist of interconnected neurons that learn by adjusting weights using backpropagation (Rumelhart et al. 1986). Deep neural networks, with many hidden layers (Fiesler et al. 1996), have achieved top results in image recognition (Krizhevsky et al. 2012), speech recognition (Hinton et al. 2012), and natural language processing (Vaswani et al. 2017).

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 (Payares et al. 2021).

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 (Abreu et al. 2024, Kim et al. 2024).

On the other hand, recent research (Payares-Martínez et al. 2021) 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, (Said et al. 2023), 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 (Abreu et al. 2024, Kim et al. 2024). 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 (Cerezo et al. 2022) to efficiently exploit the principles on which they are based, especially considering that threats and attack techniques evolve in parallel (Faker et al. 2023).

## 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 (Das et al. 2022). 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.

### 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 (Figure 1), leading to a relevant class imbalance that needs to be considered when benchmarking metrics, especially in the binary case.

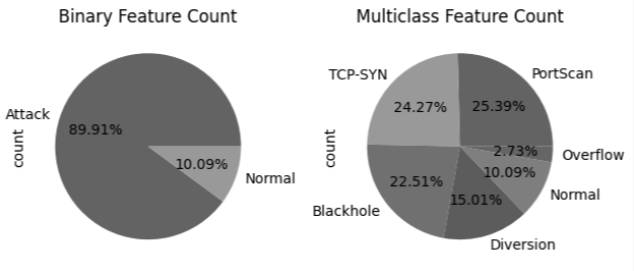


Figure 1: Binary and Multiclass Label Percentages.

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 reason is that the importance of the features indicates that the remaining features are not informative enough, and the choice of 4 elements is based on a reasonable value for the dimensionality of quantum state vectors, which are proportional to powers of 2. Finally, the accuracies of all feature combinations are calculated using RFs, KNNs, SVMs and NNs, with a proportion of train and test samples of 80% and 20% respectively. The highest overall accuracy of the four ML models indicates the best feature combination. Once this combination is found, the number of samples is chosen appropriately after recalculating the accuracies for the best combination using dataset sizes ranging from 500 to 30000 samples in steps of 500.

## Application of QML to the UNR-IDD NIDS

In this section, we focus on the application of QML algorithms to the NIDS task using the UNR-IDD. This section is organized to systematically explore the quantum implementations of six algorithms: 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 (Qiskit 2024) was chosen to implement the QML models except for QSVM, which was programmed using PennyLane (PennyLane 2024) due to its more convenient approach.

### Quantum k-Nearest Neighbors (QKNNs)

Three QKNN model versions have been designed with progressively increasing complexity: QKNN-V1, QKNN-V2 and QKNN-V3. QKNN-V1 serves as the simplest implementation, providing 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 goes a step further by introducing an adaptation of the circuit of (DiAdamo et al. 2022) 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 (Afham et. al. 2021), represents the most sophisticated version incorporating advanced quantum circuit designs and encoding strategies, making it the most complex and resource-intensive of the three.

To calculate Euclidean quantum distances, the fidelity equation $F = |\langle\psi|\phi\rangle|^2$ is used, where $\psi$ and $\phi$ 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.

**QKNN-V1**

To compute quantum inner products, the circuit shown in Figure 2 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 $n = 4$.

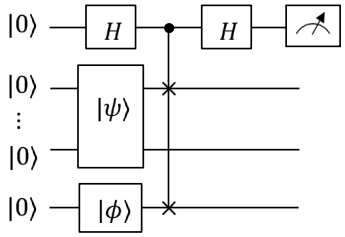


Figure 2: QKNN-V1 Quantum Circuit.

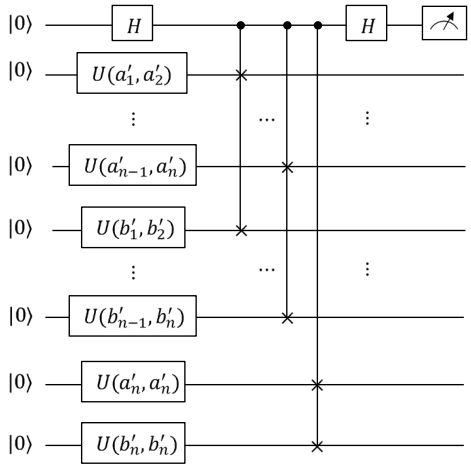


Figure 3: QKNN-V2 Quantum Circuit.

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 $P(1) = \frac{1}{2} - \frac{1}{2}|\langle\psi|\phi\rangle|^2$.

**QKNN-V2**

This adaptation improves on the previous QKNN-V1 angle encoding implementation by applying unitary $U$ 2D rotations Eq. (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. Figure 3 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 $\pi/2$.

**QKNN-V3**

Finally, the most complex and resource-intensive implementation of the QKNN was performed based on the algorithm by (Afham et al. 2021). 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 Figure 4 in (Afham et al. 2021). 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 Figure 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.

## 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 Figure 4, 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 (Qiskit 2024). This process is equivalent to calculating the inner products for each kernel entry, as represented by the Gram matrix in Eq. (2).

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

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 $L(\alpha) = \sum_{j=1}^{M} y_j\alpha_j - \frac{1}{2}\sum_{j,k=1}^{M}\alpha_j K_{jk}\alpha_k$ (Rebentrost et al. 2014) with the

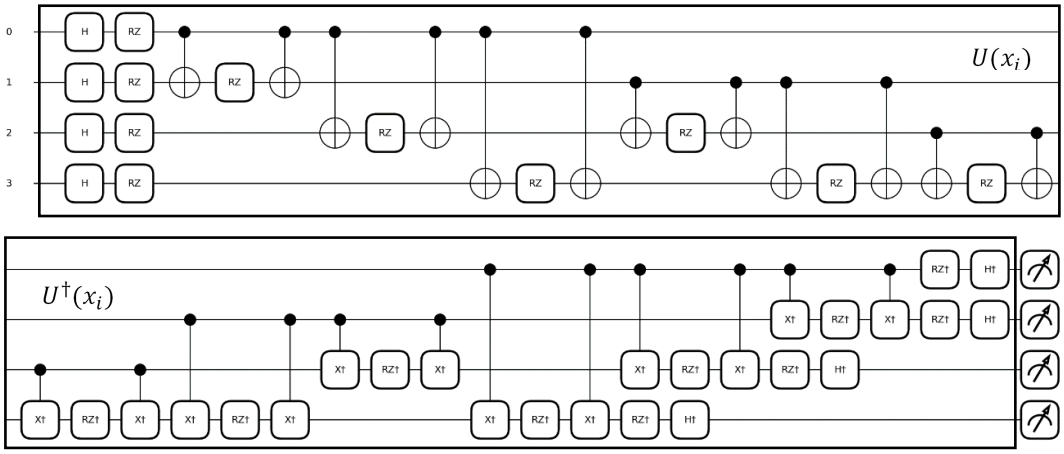


Figure 4: Custom Version of ZZFeatureMap to Compute QSVM Quantum Kernel Function.

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 $\alpha$ coefficients and predict the test labels. Finally, a classical SVM is executed using these results to 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.

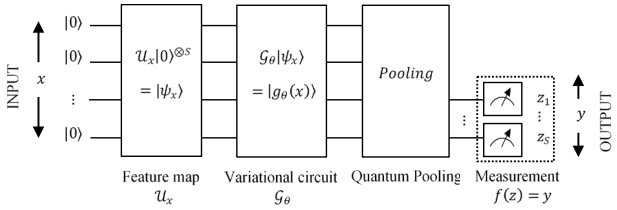


Figure 5: Quantum Neural Network Architecture.

## 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 Figure 5 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 $z = (z_1, \ldots, z_S)$ is derived from a classical post-processing function applied to a measurement result $y = f(z)$ (Abbas et. al. 2020).

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 Figure 6.

Two combinations of feature map and ansatz were tested for both scenarios using the Qiskit library (Qiskit 2024). The first combination consisted of a `ZZFeatureMap` and a modified `EfficientSU2` ansatz with four repetitions as shown in Figure 7. The second combination consisted of a `PauliFeatureMap` paired with a `RealAmplitudes` ansatz, also with four repetitions, as seen in Figure 8. To perform the measurement, the quantum pooling layer was used to keep 1 and 3 remaining qubits for the binary and multiclass scenarios, respectively. To build and train the model, the `SamplerQNN` instance was used utilizing 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.

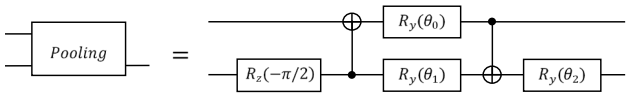


Figure 6: Quantum Pooling Circuit.

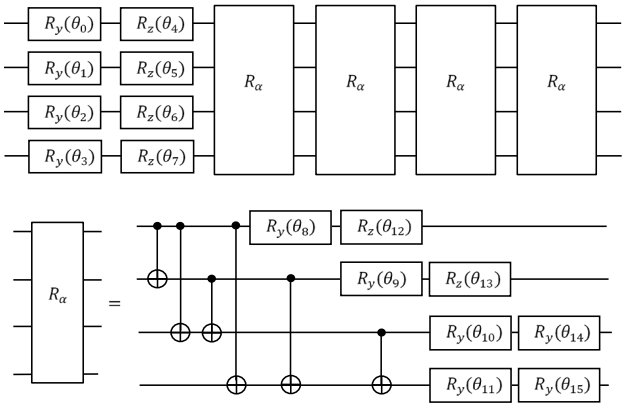


Figure 7: EfficientSU2 Variational Quantum Circuit with 4 Repetitions.

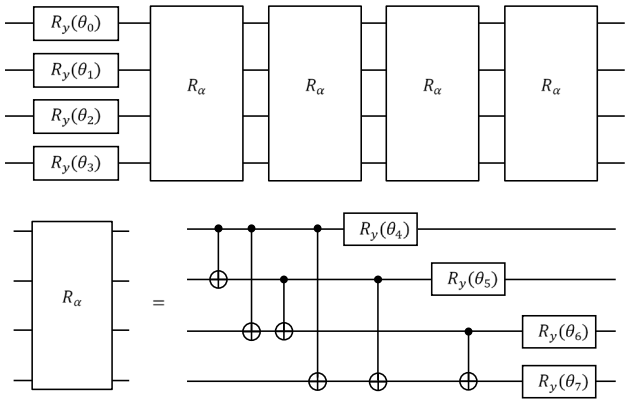


Figure 8: RealAmplitudes Variational Quantum Circuit.

## Hybrid Quantum-Classical Neural Networks (HQNNs)

HQNNs integrate feature maps, quantum variational circuits, and classical neuron layers. The structure of an HQNN can be designed in numerous ways. In this case, we use the two previously defined combinations of feature map, ansatz, and quantum pooling layers, to harness entanglement and superposition. The outputs of these quantum components are measured and integrated into a classical feedforward neural network using Qiskit's `TorchConnector`, as seen in Figure 9. The classical NN consists of two hidden layers interspersed with dropout and batch normalization layers, along with Leaky ReLU activation functions. The HQNN is trained over 50 epochs using Adam optimization for classical neurons, while the parameter-shift rule is used in quantum layers. For binary and multiclass classification tasks, the QNN used L2 and cross-entropy objective functions, respectively, while the HQNN used binary and standard cross-entropy objective functions for these tasks.

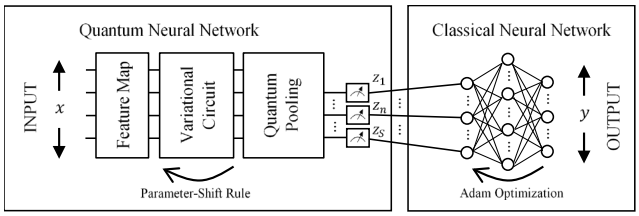


Figure 9: Hybrid Quantum-Classical Neural Network Scheme.

## Classical Model Results

Figure 10 illustrates the accuracy results for classical models based on four-element feature combinations using the entire dataset. As shown in the graphs, 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. Figure 11 shows the respective accuracies of the features as a function of the number of samples.

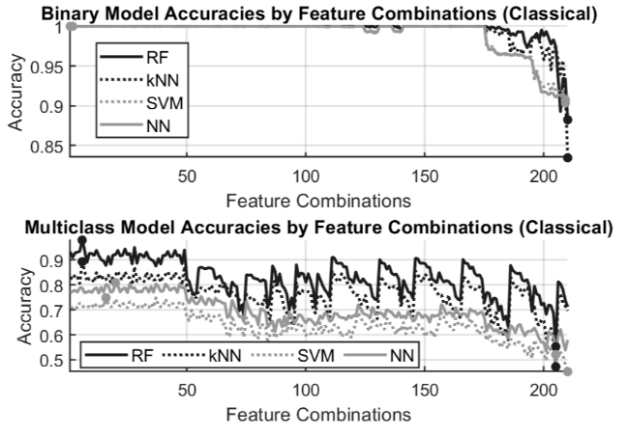


Figure 10: Binary and Multiclass Classical Accuracies as a Function of Four-Element Feature Combinations.

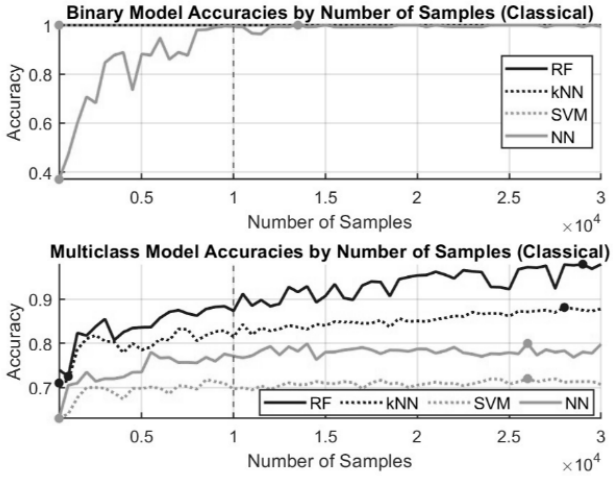


Figure 11: Binary and Multiclass Classical Accuracies for the Best Feature Combination Over the Number of Samples.

In Table 1 the mean $F1$ score ($\overline{F1}$) is included to account for the significant class imbalance, as previously shown in Figure 1. For this reason, the dataset was shuffled between runs to avoid training bias. The similarity between accuracies and $\overline{F1}$ scores suggests that the models achieve a good balance between precision and recall. Notably, the multiclass scenario presented greater classification challenges.

Table 1: 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% |

## Quantum Model Results

Figure 12 shows the evolution of QNN loss over training iterations. Notably, 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.

Figure 13 depicts the HQNN train and test accuracies in function of the number of epochs. The first binary classification graph corresponds to the circuit combination of `PauliFeatureMap` and `RealAmplitudes`, whereas the multiclass graph contains the `ZZFeatureMap` and `EfficientSU2` combination. These results align with the loss trends observed for QNN in Figure 12.

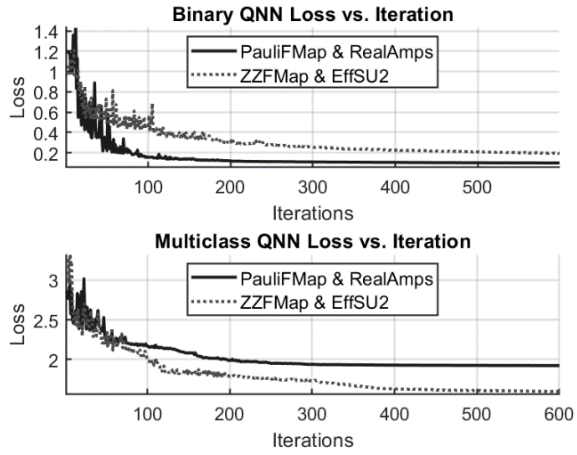


Figure 12: Binary and Multiclass Quantum Neural Network Training Loss Graphs.

A general overview of all QML results can be found in Table 2. In binary classification, several models achieve 100% accuracy: QKNN-V1 (angle encoding), QKNN-V2, QSVM and HQNN. For the multiclass scenario, the best accuracy was achieved by HQNN with a value of 78.24%, outperforming classical NN, followed by QSVM with 72.40%, also overtaking classical SVM results. On the other hand, it must be mentioned that other algorithms such as QKNN-V1 have severe limitations with only 46.80% and 19.80% for angle and amplitude encoding respectively.

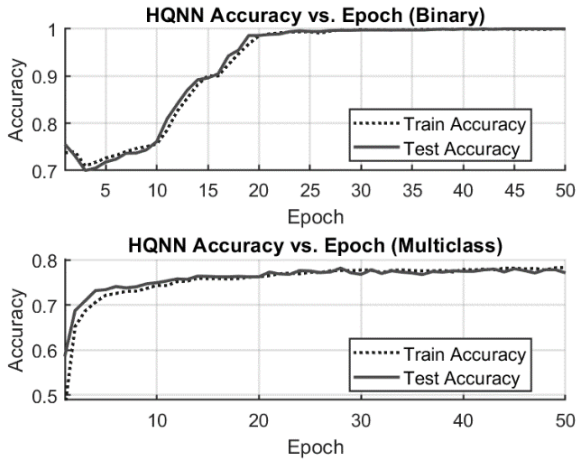


Figure 13: Binary and Multiclass Hybrid Quantum-Classical Neural Network Training Graphs (PauliFeatureMap and RealAmplitudes Combination).

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, as mentioned earlier, which makes a comparison of computation time not fair and not representative.

Table 2: Quantum Model Comparison for 10000 Samples (80% train and 20% test).

| Quantum 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*** | Binary | 85% |
| | Multiclass | 55% |
| **QSVM** | Binary | 100% |
| | Multiclass | 72.40% |
| **QNN** | Binary | 98.90% |
| | Multiclass | 63.55% |
| **HQNN** | Binary | 100% |
| | Multiclass | 78.24% |

*200 samples

## Conclusions

This work shows that quantum computing algorithms can improve the results of classical machine learning, even

using a smaller number of samples. 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 and efficiency in cybersecurity.

## Acknowledgments

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 under Grant Agreement No. 101139182, and Spanish MICIU founded, TRAINER-B (PID2020-118011GB-C22).

## References

A. A. Amin, M. S. Iqbal, M. H. Shahbaz, "Development of Intelligent Fault-Tolerant Control Systems with Machine Learning, Deep Learning, and Transfer Learning Algorithms: A Review," Expert Systems with Applications, vol 238(B) 2024, https://doi.org/10.1016/j.eswa.2023.121956.

Abbas, A., Sutter, D., Zoufal, C., Lucchi, A., Figalli, A., & Woerner, S. (2021). The power of quantum neural networks. Nature Computational Science, 1(6), 403-409.

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). IEEE.

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

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

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). Doi

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

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.

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.

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.

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.

Fiesler, E., & Beale, R. (Eds.). (1996). Handbook of Neural Computation (1st ed.). CRC Press. https://doi.org/10.1201/9780429142772

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

Hinton, G., Deng, L., Yu, D., et al. (2012). Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. IEEE Signal Processing Magazine, 29(6), 82–97. https://doi.org/10.1016/j.iot.2020.100289.

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.

K. Nisar, et al., "A survey on the architecture, application, and security of software defined networking: Challenges and open issues," Internet of Things, vol 12, 100289, 2020,

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

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. Advances in Neural Information Processing Systems, 25, 1097–1105.

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

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. Doi

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

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

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

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

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

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.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., … & Polosukhin, I. (2017). Attention is all you need. Advances in Neural Information Processing Systems, 30, 5998–6008.

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.

Y. Lu, A. Sigov, L. Ratkin, L. A. Ivanov, M. Zuo, "Quantum computing and industrial information integration: A review," Journal of Industrial Information Integration, vol 35, 100511, 2023, https://doi.org/10.1016/j.jii.2023.100511.