Quantum Machine Learning with Enhanced Autoencoders for Intrusion Detection

Milleny Teixeira de Souza, Matheus Alcântara Souza, Henrique Cota de Freitas

Computer Architecture and Parallel Processing Team (CArT)
Department of Computer Science – Pontifícia Universidade Católica de Minas Gerais
30.535-901 – Belo Horizonte – MG – Brazil

1284418@sga.pucminas.br,{matheusalcantara, cota}@pucminas.br

Abstract. *Intrusion detection remains a critical challenge in cybersecurity, particularly when dealing with high-dimensional data. In this work, we propose a hybrid quantum machine learning model that integrates a Quantum Autoencoder (QAE) for dimensionality reduction with a Quantum Neural Network (QNN) classifier. The QAE compresses input features into latent states without relying on classical methods such as Principal Component Analysis (PCA), preserving quantum-relevant information. These latent states are then classified using a Variational Quantum Circuit implemented with Qiskit. We systematically evaluated different ansatz architectures and hyperparameters on the NSL-KDD dataset. The best configuration, an EstimatorQNN with EfficientSU2 ansatz (4 repetitions), achieved an F1-score of 0.767, with balanced recall (0.70) and precision (0.84). These results highlight the potential of quantum-enhanced models for addressing the scalability challenges of intrusion detection, representing a step toward applying High-Performance Computing principles to cybersecurity.*

1. Introduction

Quantum computing is a broad field that leverages principles from physics to achieve superior computational performance (Johnston et al., 2019). It is closely related to High-Performance Computing (Britt et al., 2017), as both aim to utilize concurrent operations to tackle complex problems. A key advantage is that an increase in the number of *qubits* substantially expands the computational space, enabling simultaneous processing of vast amounts of information. This has led to promising applications across diverse domains, including biotechnology (Gabrich et al., 2024), cryptography (Vadisetty and Polamarasetti, 2024), face recognition (Zhu et al., 2024), and intrusion detection systems (Kukliansky et al., 2024).

As the landscape of cyber threats grows in complexity and volume, developing robust defense strategies has become a critical priority. Annual reports consistently highlight the rising sophistication of cyber threats and the lack of preparedness among companies (WEF, 2025). The increased use of Artificial Intelligence (AI) by adversaries is a particular concern, enabling more sophisticated and unpredictable attacks. In response, Intrusion Detection Systems (IDS) have increased efforts to improve detection

The authors would like to thank the National Council for Scientific and Technological Development of Brazil (CNPq - Codes 311697/2022-4 and 402837/2024-0) and PUC Minas FIP (Code 2025/32469).

quality using classical machine learning algorithms (Maciel et al., 2024, 2020). In parallel, the rapid advancement of Quantum Computing presents a dual challenge and opportunity: while it challenges current defense strategies (e.g., by threatening to break cryptographic algorithms), it also opens the door to radically new methods for data processing and cybersecurity defense.

With the exponential growth in network traffic, which leads to a corresponding increase in the volume and complexity of data that must be processed, performing Artificial Intelligence (AI)–based intrusion detection using *classical* machine-learning algorithms can be highly challenging, due to current hardware limitations. Modern classical algorithms are notoriously computationally intensive to train (Zaman et al., 2025), requiring massive investments in time, specialized hardware, and energy resources. Recent reports show that the computational demands for AI are already growing at *twice* the rate predicted by Moore's law (Moore, 2006; Crawford et al., 2025). Moreover, reports reveal that in the United States, data centers, particularly those used for AI training, can consume 4.4% of all energy produced in the US (O'Donnell, 2025). Therefore, classical machine learning algorithms often struggle to scale efficiently under these conditions. This bottleneck motivates the exploration of novel computational paradigms. For this reason, combining the power of quantum computing with machine learning techniques is a promising approach to better defend against and predict cyber attacks (Kukliansky et al., 2024; Hdaib et al., 2024).

In this study, we aim to investigate the use of Quantum Machine Learning in Network Anomaly Detection. First, we proposed the use of a Quantum Autoencoder (QAE) as a dimensionality reduction technique, unlike classical methods such as Principal Component Analysis (PCA) or feature selection, which risk discarding relevant information. Second, we fed the compressed latent states into a Quantum Neural Network (QNN), constructed with a Variational Quantum Circuit (VQC) and implemented using IBM Qiskit's Neural Network libraries, for binary classification between normal and anomalous traffic.

Our main contribution is a hybrid architecture for network anomaly detection based on a QAE and a QNN. Our results achieved a promising F1-score, indicating the high-precision capability of our proposed architecture. This work represents a meaningful step toward advancing High-Performance Computing (HPC) in cybersecurity by leveraging quantum principles to handle the complex, high-dimensional data that challenges traditional systems. Thus, it demonstrates how theoretical quantum approaches can lead to a future where HPC is truly scalable and more effective for critical applications, such as intrusion detection.

This paper is organized as follows: Section 2 describes important concepts related to Quantum Machine Learning as background; Section 3 discusses related works; Section 4 presents our proposal; Section 5 describes the simulation method; Section 6 shows the result evaluation; and Section 7 concludes the paper.

2. Background

2.1. Quantum Computing Fundamentals

In classical computing systems, operations are based on classical bits. A classical bit is a binary unit that can be either 0 or 1. In quantum computing (Johnston et al., 2019),

we deal with quantum bits, or qubits, which can exist in a ***superposition*** of both 0 and 1 simultaneously. Unlike classical bits, a qubit's state is described by a vector where the entries correspond to the probability amplitudes of being in each state.

Classical computers use digital logic circuits to perform operations on bits. Similarly, quantum computation is performed using *quantum circuits* that operate on qubits. These circuits consist of *quantum gates*, which are unitary operations that modify the state of single or multiple qubits. Examples of fundamental quantum gates include the Pauli gates (X, Y, Z), the Hadamard gate (H), and the controlled-NOT (CNOT) gate.

2.2. Quantum Machine Learning

Quantum Machine Learning (QML) (Du et al., 2025; Mensa et al., 2025; Zeguendry et al., 2023) is a research field that leverages the capabilities of quantum computing to perform machine learning tasks more efficiently. By exploiting quantum phenomena such as superposition and entanglement, QML aims to achieve computational advantages over classical approaches.

QML can be categorized into four main paradigms, depending on the nature of the data and the type of processing algorithm (classical or quantum): ***1. CC (Classical–Classical)***: Traditional machine learning entirely on classical data and processors. ***2. QC (Quantum–Classical)***: Classical machine learning techniques used to analyze or optimize quantum systems. ***3. CQ (Classical–Quantum)***: Quantum computers are employed to process classical data — the primary focus of most current QML research (Zeguendry et al., 2023; Du et al., 2025). ***4. QQ (Quantum–Quantum)***: Quantum data generated by quantum systems is processed by quantum algorithms.

In this work, we address the CQ paradigm through a Hybrid Classical–Quantum approach based on a ***Variational Quantum Circuit (VQC)***. VQCs are quantum circuits comprising parameterized gates with trainable parameters, denoted by θ, and fixed gates such as CNOT and CZ, which are used to introduce entanglement between qubits.

Some examples of QML algorithms include Quantum Neural Networks (QNNs) and Quantum Support Vector Machines (QSVMs), among other QML algorithms. This study focuses specifically on the implementation and evaluation of QNNs.

It is important to note that in quantum computing literature, QNNs typically refer to models based on parameterized quantum circuits, which are often implemented as VQCs, and do not follow the layered architecture of classical neural networks. From a quantum perspective, QNNs are variational quantum algorithms that are trained by minimizing a cost function, utilizing classical optimizers within a hybrid quantum-classical loop. Consequently, the VQC serves as the core circuit architecture of a QNN.

2.3. Quantum Autoencoder

To understand Quantum Autoencoders (QAEs) (Bravo-Prieto, 2021; Romero et al., 2017), we step back to their classical version. A classical autoencoder (CAE) is a type of neural network architecture that is commonly used to compress and encode information from the input efficiently. Following compression, one can then uncompress the data through the use of a decoder.

Similarly, the Quantum Autoencoder (QAE) is a QNN designed to perform information compression. Its goal is to learn a low-dimensional representation of input quantum states by preserving relevant information in a subset of qubits, called *latent space*, while discarding redundant or less informative parts of the original data in a separate set of qubits called *trash space*.

The QAE circuit is, primarily, a VQC. The input circuit is structured into some spaces with distinct roles. The circuit structure includes an **Encoder**, composed of a set of parameterized unitary operations $U(\theta)$ with trainable parameters θ, and a **Decoder**, which corresponds to the inverse operation of the encoder.

The encoder is a parameterized variational quantum circuit (VQC) that receives the input state $|\psi_{\text{input}}\rangle$ and applies a series of unitary operations with trainable parameters $U(\theta)$. Additionally, two reference qubits and one auxiliary qubit initialized in the state $|0\rangle$ are appended to the system.

At the bottleneck, the system is partitioned into distinct subspaces: a **latent space** (A), composed of four qubits that represent the compressed features; a **trash space** (B), consisting of two qubits intended to absorb redundant or irrelevant information; and a **reference space** (B'), formed by the two qubits initialized in the reference state $|0\rangle$.

A visual representation of this architecture is provided in Figure 1.

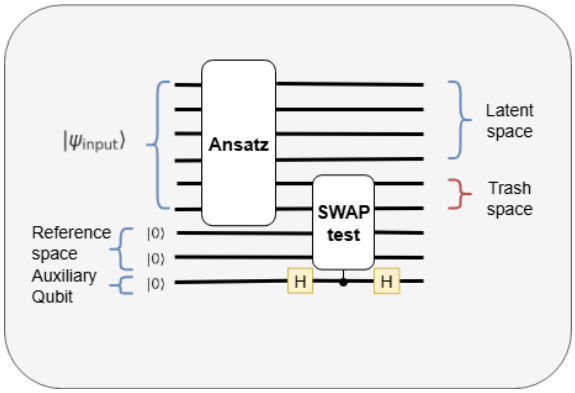

Figure 1. The encoder circuit

To learn data compression, we train our VQC to find the optimal parameter values. For each step of the training process, we need to evolve the input state $|\psi_{input}\rangle$ under the unitary operations with a set of parameters (to be optimized) $U(\theta)$ and perform a SWAP test between the trash space and the reference space (Romero et al., 2017).

The **SWAP test** is used to evaluate how close the state of the trash qubits is to the reference state $|0\rangle$. In the ideal case, after training, the trash qubits should be driven as close as possible to the state $|0\rangle^{\otimes|B|}$, meaning that all redundant information has been successfully discarded.

The SWAP test works by entangling the trash subsystem (B) with the reference subsystem (B') using an auxiliary qubit. The measurement of the Pauli-Z operator on this auxiliary qubit provides information about the overlap (fidelity) between the two states.

If the trash subsystem is exactly in the reference state $|0\rangle$, then the SWAP test

yields an expectation value $\langle Z \rangle \approx 1$. Deviations from this ideal value indicate that residual, unwanted information remains in the trash qubits.

Based on this, the cost function is defined as

$$\mathcal{C}(\theta) = 1 - \langle Z \rangle, \tag{1}$$

which penalizes deviations from the desired reference state. Minimizing this cost function using a classical optimizer (e.g., SPSA, COBYLA) encourages the encoder to compress information into the latent qubits while discarding irrelevant information into the trash qubits.

3. Related Work

Several works have focused on Quantum Machine Learning in Intrusion Detection Systems. Abreu et al. (2024a) explores three different models of QML: Variational Quantum Classifier, QSVM, and Quantum K-means (QKM) applied to a hybrid pipeline for intrusion detection, with a multi-class classification method. Later in 2024, the authors extended the research, incorporating the Quantum Convolutional Neural Network (QCNN) (Abreu et al., 2024b) and including pooling layers for dimensionality reduction.

Hdaib et al. (2024) explores more profoundly the use of Quantum Autoencoders as part of a QML intrusion detection model. The study proposes three frameworks, using the QAE as a dimensionality reduction method and applying it to the models, one-class SVM, Random Forest, and kNN. In this study, the authors used PCA to reduce the number of features, followed by the amplitude encoding technique. However, this work did not explore QNNs in the sequence of the QAE.

Other works have proposed the use of QAE as a classifier. Frehner and Stockinger (2024) explored QAEs for time series anomaly detection by investigating two classification techniques. The first method analyzes the reconstruction error, while the second uses latent representation analysis via a Swap-Test measurement. Similarly, Huot et al. (2024) proposed a Quantum Autoencoder-based Fraud Detection (QAE-FD) model that employs only the encoder component of the QAE as a feature extractor. In this approach, transactions are classified by applying a threshold to the measurement outcomes of the trash qubits, which separates anomalous inputs from normal ones. In the intrusion detection field, this approach is still underexplored.

In our work, we explore the power of the QAE to compress data, implementing a quantum reductionality method instead of relying on classical methods (e.g, PCA, Selectkbest). QAE has proved to be an efficient method for data compression. However, many QML algorithms designed for network intrusion detection still rely on classical methods, instead of exploring quantum methods. We also address the current lack of research on implementing QNN for intrusion detection, which typically involves algorithms such as QSVM or kNN. We also propose an adaptation of the EF-QAE (Bravo-Prieto, 2021) to allow encoding a higher number of features into the circuit, while keeping it with a reasonable depth.

4. Intrusion Detection Proposal

This section introduces our intrusion detection proposal based on Quantum Machine Learning with Enhanced Autoencoders.

4.1. Dataset and data preprocessing

As network traffic data is sensitive, performing AI training in real-world scenarios is a challenging task. For this reason, publicly available datasets have been developed to support advancements in research. In our case, we chose the NSL-KDD dataset (Zaib, 2019) for model training. The NSL-KDD is an academic intrusion detection dataset and remains one of the most widely used datasets in the field of intrusion detection.

Preparing classical data for use in quantum machine learning (QML) algorithms is a crucial step for their performance, as it involves encoding classical data into quantum states. To accomplish this, we applied classical preprocessing techniques.

- Nominal features were transformed using one-hot encoding. For nominal features with a high number of distinct values, label encoding was applied instead to avoid augmenting the number of columns.
- The min-max normalization was applied to scale the data to the range of $[-\pi, \pi]$
- As a dimensionality reduction technique, we applied the QAE to compress data, which is explained in the next section.

4.2. The Quantum Autoencoder for dimensionality reduction

Dealing with high-dimensional data is one of the challenges faced when training a QML model. For intrusion detection datasets such as NSL-KDD, reducing the number of features while retaining meaningful information is a crucial preprocessing step. Classical dimensionality reduction techniques, such as PCA or SelectKBest, are widely used in the literature, including in QML. However, these classical methods may discard important quantum-relevant patterns during transformation, potentially limiting the performance of quantum models. Quantum versions of classical methods, such as QSPSA and QUBO problems, are a research field that is advancing but is still in its initial stages.

For that reason, in this work, the QAE is employed as a quantum dimensionality reduction method. QAEs are effective for quantum data compression (Romero et al., 2017), making them a promising alternative to classical methods. We aim to explore the QAE to compress the data, without relying on classical methods.

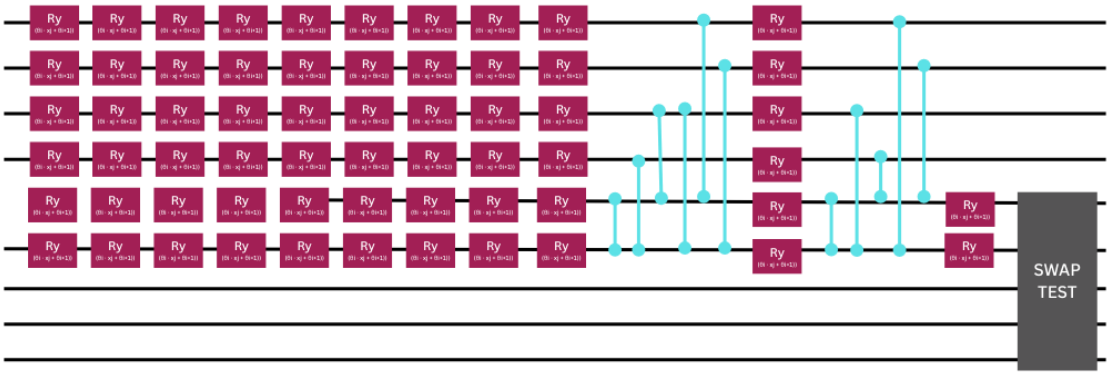

Figure 2. The adapted QAE circuit representation

4.2.1. The QAE's architecture

Our QAE's architecture was built using six qubits for the input states, two reference qubits, and an auxiliary qubit for the swap test, divided as previously shown in Figure 1. Thus, the dataset's features must be encoded to fit this limit. As discussed, applying classical dimensionality reduction prior to the QAE was intentionally avoided to preserve quantum-relevant information. Another possibility is amplitude encoding, which theoretically allows representing N features using $\lceil \log_2 N \rceil$ qubits. While efficient in theory, this method was tested and is extremely computationally expensive in practice.

To achieve that, we adopt an architecture that allows us to encode all the features of the dataset by adapting the strategy proposed in the EF-QAE model by Bravo-Prieto (2021). Unlike the standard QAE, the EF-QAE incorporates a classical feature vector directly into the parameterized rotations of the circuit. The features are encoded into each of the single *Ry* qubit rotations according to the equation

$$R_y(\theta_i \cdot x_j + \theta_{i+1}), \tag{2}$$

where x_j is the j-th component of the feature vector and θ are trainable parameters optimized through a quantum-classical feedback loop.

In our study, we adapt this encoding strategy to handle the high-dimensional classical data from the dataset. Specifically, we extend the feature vector $\mathbf{x}$ to represent the full set of 54 features, embedding them directly into the circuit rotations. This adaptation takes advantage of the hybrid parameterization scheme EF-QAE to efficiently incorporate all features in a circuit by directly encoding the dataset into the variational rotations, removing the need for classical dimensionality reduction methods by allowing the autoencoder to learn compression in a quantum-native manner. Furthermore, it maintains the circuit's reasonable depth, thereby avoiding computational increases and the occurrence of barren plateaus.

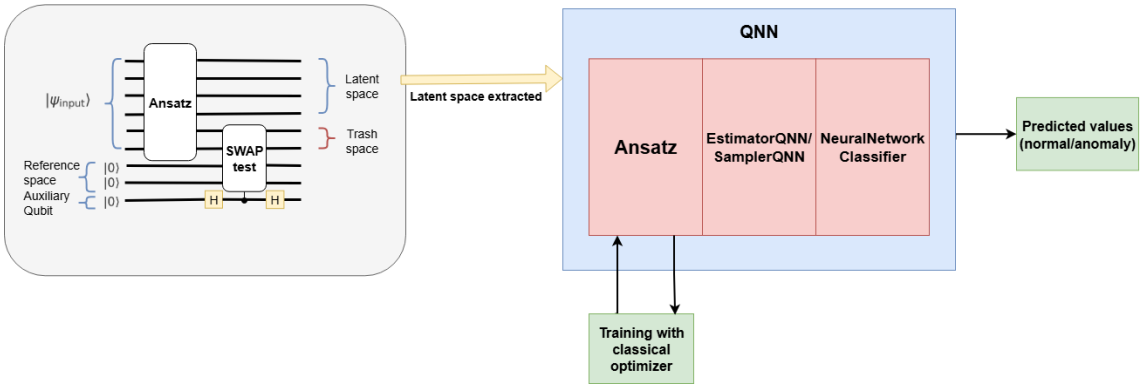

Figure 3. Complete pipeline. The QAE encoder circuit is used to extract the compressed data on the latent space, and it is used to feed the QNN

We then need to train the QAE to perform compression by minimizing the cost function (see Section 2.3). We tested the classical optimizers COBYLA, SPSA, and L_BFGS_B, ultimately choosing COBYLA as it showed better convergence on practical tests. We then use the information encoded in the latent qubits (the compressed data) as

input to our QNN, effectively acting as a feature-extraction layer in the final classification pipeline (Figure 3).

4.3. Quantum Neural Network (QNN)

Quantum Neural Networks (QNNs) are composed of basically an input circuit, an ansatz (a parameterized circuit), and an output layer where the prediction values are.

For the input layer, we utilize the QAE encoder with fixed, optimized parameters from previous training to extract the latent space and apply an ansatz comprising rotations and entangling gates. The circuit, consisting of the encoder (selecting only the latent qubits) plus the ansatz with trainable parameters, is then used as the input circuit for the EstimatorQNN or SamplerQNN class in Qiskit, which are neural network implementations from the Qiskit Machine Learning library. The EstimatorQNN is designed to output expectation values from defined observables on the circuit. On the other hand, SamplerQNN is built to produce probability distributions from quantum circuits (Mensa et al., 2025). The last one is especially well-suited for classification tasks, although both can be applied for this purpose.

In sequence, we pass the instance created to qiskit's NeuralNetworkClassifier module. Once the QNN is defined, the NeuralNetworkClassifier wraps it into a supervised learning model that can be trained using classical optimizers. In our work, this module enables integration of the whole pipeline and performs binary classification between normal and anomalous traffic samples in the NSL-KDD dataset.

For the EstimatorQNN-based pipeline, to read out predictions, we define Pauli-Z observables and measure their expectation values $\langle Z \rangle$. For the SamplerQNN, we define an interpret function to read out the probabilities. In both cases, we employ the cross-entropy loss function. Training is carried out using the classical COBYLA optimizer, as in our tests, it showed reduced computational cost compared to SPSA, while still providing reliable convergence.

5. Simulation Method

The experiments were conducted on an Amazon Web Services (AWS) virtual machine c7i.8xlarge and c6i.8xlarge. The simulations were executed using the Qiskit Aer simulator, without access to a physical quantum backend. The computational environment consisted of Ubuntu 22.04 with Python 3.12. The following library versions were used in our experiments: *qiskit 1.2.4*, *scikit-learn 1.7.1*, *jupyterlab 4.4.6*, *qiskit-aer 0.17.1*, *qiskit-algorithms 0.3.1*, *qiskit-machine-learning 0.8.3*, and *numpy 2.3.2*.

The model's training is divided into two phases: (1) The QAE is trained for efficient data compression, using COBYLA to optimize the parameters, with a cost function based on the expectation value on the auxiliary qubit (see Sections 2.3 and 4.2.1). (2) We use the compressed data as input for the QNN using the classes described in the last section.

As explained before, an important part of a QNN is the ansatz/VQC, as we will submit to optimization of the parameterized circuit. In order to test different ansatz architectures and evaluate their impact on the model's performance, we varied the following hyperparameters:

- Ansatz: The parameterized circuit (VQC), we tested the Real Amplitudes, EfficientSU2, and TwoLocal architectures.
- Reps: Number of the ansatz repetitions, we varied from 1 to 4
- Qiskit's Neural Network Classes: EstimatorQNN and SamplerQNN were used.

6. Result Evaluation

Our evaluation method includes some metrics: accuracy, F1-score, precision, and recall. The results for each choice of hyperparameter are shown in Tables 1 and 2. As we are dealing with an unbalanced dataset, accuracy is not our main metric of evaluation. This is because accuracy can be misleading in imbalanced scenarios, as it reflects high values simply due to the prevalence of the majority class, without capturing how well the model identifies the minority class. Instead, we prioritize the F1-score, which is the harmonic mean of precision and recall, providing a balanced measure that accounts for both false positives and false negatives (Abdelhamid and Desai, 2024).

Table 1. Performance metrics for EstimatorQNN with ansatz types and repetitions

NeuralNetwork Class	Ansatz Type	Reps	Accuracy	F1 Score	Precision	Recall
EstimatorQNN	TwoLocal	1	0.6868	0.6277	0.9710	0.4637
EstimatorQNN	TwoLocal	2	0.6191	0.5077	0.9605	0.3450
EstimatorQNN	TwoLocal	3	0.5839	0.4291	0.9800	0.2747
EstimatorQNN	TwoLocal	4	0.7588	0.77076	0.8399	0.7121
EstimatorQNN	Real Amplitudes	1	0.6871	0.6377	0.9356	0.4837
EstimatorQNN	Real Amplitudes	2	0.7612	**0.7597**	0.8890	0.6633
EstimatorQNN	Real Amplitudes	3	0.7599	**0.7528**	0.9092	0.6423
EstimatorQNN	Real Amplitudes	4	0.7398	0.7296	0.8933	0.61661
EstimatorQNN	EfficientSU2	1	0.6833	0.6205	0.9760	0.4548
EstimatorQNN	EfficientSU2	2	0.7401	**0.7068**	0.9878	0.5502
EstimatorQNN	EfficientSU2	3	0.7541	**0.7544**	0.8741	0.6635
EstimatorQNN	EfficientSU2	4	0.7576	**0.7670**	0.8470	0.7008

Overall, the precision remained high for all configurations, with values higher than 0.8. However, we observe a different behavior for the Recall, especially when increasing the number of repetitions.

TwoLocal Ansatz achieved high precision values across all configurations (above 0.83), the Recall dropped significantly with increased repetitions until 3, falling from 0.46 (1 rep) to 0.27 (3 reps). Consequently, its F1-score dropped from 0.62 to 0.42. However, with a higher number (4), the values augmented.

In contrast, the Real Amplitudes Ansatz demonstrated better balance between precision and recall for configurations with 2 or 3 repetitions. The best performance was achieved with two repetitions, with an accuracy of 0.76 and an F1-score of 0.759. However, performance dropped drastically at four repetitions, where the F1-score dropped to 0.287 and recall to just 0.16, likely due to overfitting or high circuit depth.

The EfficientSU2 Ansatz consistently improved with increasing repetitions. At four repetitions, it achieved the highest F1-score overall (0.767), with a recall of 0.70 and precision of 0.84. This balance suggests EfficientSU2 provides better generalization and robustness in detecting both normal and anomalous traffic. Unlike the other ansatz types,

EfficientSU2 did not show a decrease in performance with higher repetitions, making it a strong candidate for more complex quantum models.

Table 2. Performance metrics for SamplerQNN with ansatz types and repetitions.

NeuralNetwork Class	Ansatz Type	Reps	Acc	F1 Score	Precision	Recall
SamplerQNN	TwoLocal	1	0.4385	0.0367	0.7850	0.0188
SamplerQNN	TwoLocal	2	0.5704	**0.7228**	0.5712	0.9841
SamplerQNN	TwoLocal	3	0.4554	0.0936	0.8904	0.0494
SamplerQNN	TwoLocal	4	0.4742	0.1547	0.9110	0.0845
SamplerQNN	Real Amplitudes	1	0.5757	**0.7195**	0.5768	0.9561
SamplerQNN	Real Amplitudes	2	0.4478	0.1039	0.6811	0.0563
SamplerQNN	Real Amplitudes	3	0.4532	0.0797	0.9519	0.0416
SamplerQNN	Real Amplitudes	4	0.5256	0.2874	0.9908	0.1681
SamplerQNN	EfficientSU2	1	0.5659	**0.7186**	0.5694	0.9738
SamplerQNN	EfficientSU2	2	0.4536	0.0817	0.9416	0.0427
SamplerQNN	EfficientSU2	3	0.4492	0.0936	0.7402	0.0499
SamplerQNN	EfficientSU2	4	0.4416	0.0524	0.7699	0.0271

Overall, the results indicate that SamplerQNN performance is highly sensitive to the ansatz type and the number of repetitions. In most configurations, the best results are achieved at lower circuit depths, particularly with 1 or 2 repetitions. As the depth increases, a consistent pattern of degradation is observed, especially in the recall metric, suggesting training instabilities or overfitting in deeper variational circuits.

The best-performing configuration for the TwoLocal ansatz was observed at two repetitions, achieving an F1-score of 0.7228 and a very high recall of 0.9841, indicating strong sensitivity to detecting anomalies. However, configurations with 1, 3, and 4 repetitions suffered from recall degradation (below 0.09), despite keeping high precision.

Similarly, the Real Amplitudes ansatz exhibited its strongest performance at one repetition, reaching an F1-score of 0.7195 and a recall of 0.9561. As repetitions increased, recall dropped drastically, even though precision remained high. EfficientSU2 followed the same overall trend, with best performance at one repetition (F1-score: 0.7186, recall: 0.9738). All deeper configurations caused much lower recall, with the worst case at four repetitions (recall: 0.0271)

These results indicate that shallow circuits of SamplerQNNs are more suitable for classification tasks in the context of the NSL-KDD dataset. However, it also indicates that a more careful tuning of hyperparameters is necessary to further enhance the model's performance. In both cases, it is possible to notice the importance of building an expressive ansatz architecture, while carefully controlling circuit depth.

7. Conclusions

In this study, we propose a Quantum Neural Network (QNN) model to address the growing concern with cyber threats and the limitations of current computational resources by leveraging the power of quantum computing. We addressed a research gap in QNN applications by implementing an enhanced Quantum Autoencoder (QAE), enabling data compression without relying on classical dimensionality reduction methods.

Our results show promising performance. While still not surpassing classical models, the proposed architecture represents a meaningful step toward the practical use of QAE as a building block for QNN-based solutions in cybersecurity. The integration of the QAE enabled efficient data compression, allowing for the use of fewer qubits without significant loss of information.

For future work, we recommend optimizing the QNN architecture, particularly the choice of ansatz and training strategy, to enhance its classification capabilities. Evaluating the model on other publicly available intrusion detection datasets would enable a broader assessment of its generalization capabilities. Besides, there is a need better to evaluate the model's performance in real quantum hardware, in addition to simulations.

We also highlight the potential of using QAE not only as a preprocessor but directly as a classifier. While research in this direction is still in its early stages, to the best of our knowledge, no application has yet been proposed in the context of intrusion detection, making it a promising area for future exploration.

References

M. Abdelhamid and A. Desai. Balancing the scales: A comprehensive study on tackling class imbalance in binary classification, 2024. URL https://arxiv.org/abs/2409.19751.

D. Abreu, C. Rothenberg, and A. Abelém. qids: Sistema de detecção de ataques baseado em aprendizado de máquina quântico híbrido. In *XLII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, pages 295–308, 2024a. doi: 10.5753/sbrc.2024.1353.

D. Abreu, C. E. Rothenberg, and A. Abelém. Qml-ids: Quantum machine learning intrusion detection system. In *Proceedings of the IEEE Latin-American Conference on Communications (LATINCOM 2024)*. IEEE, 2024b.

C. Bravo-Prieto. Quantum autoencoders with enhanced data encoding. *Machine Learning: Science and Technology*, 2(3):035028, jul 2021. doi: 10.1088/2632-2153/ac0616. URL https://dx.doi.org/10.1088/2632-2153/ac0616.

K. A. Britt, F. A. Mohiyaddin, and T. S. Humble. Quantum accelerators for high-performance computing systems. In *2017 IEEE International Conference on Rebooting Computing (ICRC)*, pages 1–7, 2017. doi: 10.1109/ICRC.2017.8123664.

D. Crawford, A. Hoecker, and D. Aulanier. Technology report 2025, 2025. URL https://www.bain.com/globalassets/noindex/2025/bain_report_technology_report_2025.pdf. Accessed: Sept. 27, 2025.

Y. Du, X. Wang, N. Guo, Z. Yu, Y. Qian, K. Zhang, M.-H. Hsieh, P. Rebentrost, and D. Tao. Quantum machine learning: A hands-on tutorial for machine learning practitioners and researchers, 2025. URL https://arxiv.org/abs/2502.01146.

R. Frehner and K. Stockinger. Applying quantum autoencoders for time series anomaly detection, 2024. URL https://arxiv.org/abs/2410.04154.

M. Gabrich, H. Freitas, and M. Souza. Análise de redes neurais para crispr: Uma abordagem com computação quântica. In *XXV Simpósio em Sistemas Computacionais de Alto Desempenho*, pages 13–24, 2024. doi: 10.5753/sscad.2024.244778.

M. Hdaib, S. Rajasegarar, and L. Pan. Quantum deep learning-based anomaly detection

for enhanced network security. *Quantum Machine Intelligence*, 6:26, 2024. doi: 10.1007/s42484-024-00163-2.

C. Huot, S. Heng, T.-K. Kim, and Y. Han. Quantum autoencoder for enhanced fraud detection in imbalanced credit card dataset. *IEEE Access*, 12:169671–169682, 2024. doi: 10.1109/ACCESS.2024.3496901.

E. Johnston, N. Harrigan, and M. Gimeno-Segovia. *Programming Quantum Computers: Essential Algorithms and Code Samples*. O'Reilly Media, 2019.

A. Kukliansky, M. Orescanin, C. Bollmann, and T. Huffmire. Network anomaly detection using quantum neural networks on noisy quantum computers. *IEEE Transactions on Quantum Engineering*, 5:1–11, 2024. doi: 10.1109/TQE.2024.3359574.

L. A. Maciel, M. A. Souza, and H. C. Freitas. Reconfigurable fpga-based k-means/k-modes architecture for network intrusion detection. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 67(8):1459–1463, 2020. doi: 10.1109/TCSII.2019.2939826.

L. A. Maciel, M. A. Souza, and H. C. Freitas. Energy-efficient cpu+fpga-based cnn architecture for intrusion detection systems. *IEEE Consumer Electronics Magazine*, 13 (4):65–72, 2024. doi: 10.1109/MCE.2023.3283730.

S. Mensa, M. E. Sahin, E. Altamura, O. Wallis, S. P. Wood, A. Dekusar, D. A. Millar, T. Imamichi, and A. Matsuo. Qiskit machine learning: An open-source library for quantum machine learning tasks at scale on quantum hardware and classical simulators, 2025. URL https://arxiv.org/abs/2505.17756.

G. E. Moore. Cramming more components onto integrated circuits, reprinted from electronics, volume 38, number 8, april 19, 1965, pp.114 ff. *IEEE Solid-State Circuits Society Newsletter*, 11(3):33–35, 2006. doi: 10.1109/N-SSC.2006.4785860.

J. O'Donnell. A história que você ainda não conhece sobre a pegada energética da ia. *MIT Technology Review*, 2025. URL https://mittechreview.com.br/impacto-energetico-ia-inteligencia-artificial-data-centers/. Accessed: Sept. 27, 2025.

J. Romero, J. P. Olson, and A. Aspuru-Guzik. Quantum autoencoders for efficient compression of quantum data. *Quantum Science and Technology*, 2:4, 2017. doi: 10.1088/2058-9565/aa8072.

R. Vadisetty and A. Polamarasetti. Quantum computing for cryptographic security with artificial intelligence. In *2024 12th International Conference on Control, Mechatronics and Automation*, pages 252–260, 2024. doi: 10.1109/ICCMA63715.2024.10843897.

WEF. Global cybersecurity outlook 2025. Technical report, World Economic Forum, 2025. URL https://www.weforum.org/reports/global-cybersecurity-outlook-2025. Accessed: Sept. 27, 2025.

M. H. Zaib. Nsl-kdd dataset, 2019. URL https://www.kaggle.com/datasets/hassan06/nslkdd. Accessed: Sept. 27, 2025.

K. Zaman, A. Marchisio, M. A. Hanif, and M. Shafique. A survey on quantum machine learning: Current trends, challenges, opportunities, and the road ahead, 2025. URL https://arxiv.org/abs/2310.10315.

A. Zeguendry, Z. Jarir, and M. Quafafou. Quantum machine learning: A review and case studies. *Entropy*, 25(2):287, 2023. doi: 10.3390/e25020287.

Y. Zhu, A. Bouridane, M. E. Celebi, D. Konar, P. Angelov, Q. Ni, and R. Jiang. Quantum face recognition with multigate quantum convolutional neural network. *IEEE Transactions on Artificial Intelligence*, 5(12):6330–6341, 2024. doi: 10.1109/TAI.2024.3419077.