Scholar J
Computational Science

Research Article

Investigation of Benign and Scan Samples of Cyber Security Network Traffic Data: A Hybrid Quantum-Classical Study

Jessy Akaabo, Samera Otor and Aamo Iorliam

Department of Mathematics and Computer Science,
Benue State University, Makurdi, Nigeria.

Received 18 December 2024; Acceptance 16 January 2025; Published 28 February 2025.

Abstract

Worldwide internet usage is expanding quickly, creating numerous opportunities in various industries, such as sports, education, entertainment, and finance. The growth of network technology has necessitated the need for effective management, maintenance, and monitoring of network infrastructures to maximize economic efficiency and maintain smooth operations. Like any coin, the internet has merits and demerits, and cyber attacks are major demerits that every internet user should be aware of. Amongst the most common tactic used in cyber attacks is social engineering, which attackers often leverage to get user credentials and access target networks and active assets. This research employs a hybrid quantum-classical model designed using a simple variational quantum circuit with a single qubit for the training, evaluation, and classification of network traffic data into attacked and non-malicious using a network traffic dataset comprising of benign and scan samples collected by Lawrence Berkeley National Laboratory. The hybrid quantum classical model revealed an accuracy of 99.89% with a training time of 2.36 seconds and evaluation times of 0.02 seconds. Based on these results, it was therefore recommended among others, that the Hybrid Quantum-Classical model holds significant promise for the future of real time anomaly detection in the cyber security space. Overall, the adoption of the Hybrid Quantum-Classical model by cyber security practitioners, could lead to substantial improvements in identifying benign and scan data, thereby enhancing the overall security posture.

Keywords: *Benign, Scan, Hybrid, Quantum, Classical.*

Correspondence to: Jessy Akaabo, e-mail: akaabojessy@yahoo.com

How to Cite: Akaabo et al. (2025). Investigation of Benign and Scan Samples of Cyber Security Network Traffic Data: A Hybrid Quantum- Classical Study. *Scholar J Computational Science,* 2**(2)**. DOI: 10.5281/zenodo.14947966

Introduction

The study of computers' processing capacity and other characteristics using quantum-mechanical concepts is known as quantum computing, this harnesses the collective features of quantum states, such as superposition, interference and entanglement to conduct computation [1]. The groundwork for the creation of quantum computing (QC), a new paradigm for computers, is set by quantum mechanics, a branch of physics that studies the behaviour of exceedingly small particles [2].

Quantum computing systems sometimes referred to as quantum computers, come in a variety of forms. These include quantum cellular automata, quantum Turing machines, adiabatic quantum computers, one-way quantum computers, and quantum circuit models [3-4]. The most often used model is the quantum circuit, which is based on the quantum bit or qubit whose development follows the Schrödinger equation $i \partial \partial t \mid \psi = H\hat{} \mid \psi$ and has a two-level quantum system with basis states usually written as $|0\rangle$, $|1\rangle$ or a linear combination of both states at the same time, a phenomenon known as superposition. When measured, a qubit can be in one of two quantum states: 1 or 0, or it can be in a superposition of both states [5]. However, the likelihood of either result relies on the quantum state of the qubit before it is measured.

The emergence of interconnected networks has impacted sectors like finance, real estate, healthcare, and education, leading to the emergence of digitalization [6]. Cyberspace has expanded astronomically due to information technology, which is also the cause of the digitization we are currently experiencing.

More concerning though, is the fact that the development of information technology has eventually led to the emergence of a brand-new criminal activity known as cybercrime. Cybercriminals employ computers, smartphones and other electronic devices to steal critical information from vulnerable networks or shut down victims' servers, which negatively impacts businesses. It is imperative to enhance cybersecurity measures in order to combat cybercrime, given the rise in the frequency of cyberattacks [7].

Cybersecurity is a collection of tools, procedures, and practices that guard against attacks, damage, and unauthorized access to networks, devices, software, and data. And it has become imperative that advanced, automated cybersecurity techniques be implemented in order to stay up with the quickly evolving cybercrime landscape [8].

According to [9], a network intrusion detection system (NIDS) is designed to continuously scan, identify, and analyze network systems for suspicious activity. They are set up to listen to packets travelling between hosts over a network and use either packet inspection or flow detection to find security anomalies. To defend against these invasions, a hybrid classical-quantum computing approach is considered. This model takes into account the challenges and problems that network administrators have when trying to use the best feature subset to identify and classify network data and traffic into the appropriate class to which they belong [9].

A hybrid quantum-classical algorithm needs significant amounts of both quantum and classical computational resources to operate and cannot be described in isolation from classical computation [3] These hybrid classical-quantum computing algorithms consist of parameterized quantum circuits, which run on real quantum computers or quantum computer simulators, and classical neural networks, which train

and update the parameters. This means that a classical control system reads out measurements, performs some classical conditional logic at the quantum processor level, and then executes the next series of quantum operations based on the results of these so-called mid-circuit measurements. Similarly, [7] stated that the majority of quantum computing algorithms need some basic classical pre-processing to transform problems into a format that can be read by a quantum computer and that can be identified when the data is returned from the quantum computer and transformed into the required solution. A pre-processing step that gets data ready for a quantum algorithm and a post-processing step that deals with data that comes from a quantum algorithm are both incorporated into the hybrid model.

Gaikwad et al in [10] implemented IDS using the Bagging Ensemble approach. Because of its simplicity, they chose Partial Decision Tree as their basis classifier. They then selected features using a genetic algorithm. Model building time, true positives, false positives, and classification accuracy are all used to assess the suggested intrusion detection system. The results showed that, while employing cross-validation, the introduced system achieved the greatest classification accuracy of 99.71%. On the test dataset, this model outperforms all other classifiers with the exception of the C4.5 classifier in terms of classification accuracy. However, the model's primary drawback is that it takes longer to construct, making online training of the IDS less desirable.

Bhatia et al in [11] implemented a multi-level P2P traffic classification technique, sub-divided into the packet-level/flow-level classification process. Using the UNIBS offline dataset, the evaluation model was run achieving a cumulative accuracy of 98.30% which shows that the hybrid model outperformed other alternative methods.

In a paper by [12] the cluster centre and nearest neighbour (CANN) approach was presented by the authors as a unique feature representation method. This approach calculated and added two distances, the first of which was based on the separation between the cluster centre and each data sample. The next distance measures the separation between the data and its closest neighbour inside the same cluster. Each data sample is addressed for intrusion detection by a K-NN classifier using this novel, one-dimensional distance. In terms of classification accuracy, detection rates, and false alarms, the results indicate that the CANN classifier outperformed k-NN and SVM, achieving the maximum classification accuracy of 99.46%. Its inability to detect U2L and R2L attacks effectively is one of its drawbacks. This means that the one-dimensional distance-based feature representation is unable to accurately depict the pattern of these two attack types, and its computational efficiency during classifier training and testing is 1570 minutes.

Recently, quantum computing techniques have shown to be incredibly efficient at handling cybersecurity-related jobs, therefore it is in the light of this that the investigation of Benign and Scan Samples of network traffic data for cyber security applications: a hybrid quantum-classical study is presented.

Research Method

Data Source: The dataset used in this study comprises benign and scan samples collected by Lawrence Berkeley National Laboratory [11]. It has packets spanned for more than 100 hours of activities from several

thousand internal hosts and is publicly available in [13]. Two classes (normal and abnormal) were used to label the dataset which we adopted for a binary classification problem.

Data Pre-processing: We created a single dataset for our classification task by concatenating the scan and benign datasets, which were originally distinct CSV files with the same number of columns but different numbers of rows, during the data pre-processing step. But in order to differentiate the samples from the two datasets, we added a new column named "label" and gave the benign samples a label of 0. To break any innate patterns that might be present in the datasets, we reshuffle the concatenated dataset to randomly arrange the samples. Subsequently, the dataset is divided into an 80:20 train-test ratio, which guarantees that the model has enough data to learn from during training and permits independent validation during testing. Additionally, the dataset's features were standardized to promote uniform scaling across various features and support model convergence. The process of standardization entails converting the data to have a mean of 0 and a standard deviation of 1. This process helps in stabilizing the training process, especially when features have different scales or units.

Hybrid Quantum-Classical Model: The classifier to be used is a hybrid quantum-classical neural network. This type of classifier combines classical neural network layers with a variational quantum circuit layer [14]. Below is the pseudo code for the hybrid quantum-classical model:

```
Start
Load Dataset
Preprocess Data (Split into training and testing sets)
Initialize HybridModel
        Initialize Classical Layers
            Input Layer (115 features)
            Fully Connected Layer 1 (64 units) with ReLU activation
            Fully Connected Layer 2 (2 units)
        Initialize Quantum Circuit Layer
            Single Qubit Initialization
            Hadamard Gate
            RZ Gate with Trainable Parameter
            Measurement
Train Model
      For each epoch:
            Forward Pass
                  Pass input through Fully Connected Layer 1
                  Apply ReLU activation
                  Pass through Fully Connected Layer 2
                  Pass through Quantum Circuit Layer
            Compute Loss
            Backward Pass
                  Compute Gradients
                  Update Weights
            Evaluate Model (Compute accuracy on test set)
Evaluate Model on Test Set
Output Results
      Final Accuracy
      Evaluation Metrics (Precision, Recall, F1 Score)
      Confusion Matrix
Plot Training Loss and Accuracy
End
```

Classical Neural Network Layer

Input Layer (fc1): The first layer (fc1) is a fully connected (dense) layer with 115 input features and 64 output units, followed by a ReLU activation function.

Type: Linear (Fully Connected)
Input Features: 115
Output Features: 64
Bias: True

Mathematical Expression:

$$\mathrm{y1} = Wx + b \tag{1}$$

Description: This layer performs a linear transformation of the input features (115-dimensional vector) to produce a 64-dimensional output.

Hidden Layer (fc2): The second layer (fc2) is another fully connected layer with 64 input units and 2 output units, which represents the number of classes for the classification task.

Type: Linear (Fully Connected)

Input Features: 64

Output Features: 2

Bias: True

Mathematical Expression:

$$y2 = Wx + b \tag{2}$$

Description: This layer performs a linear transformation of the 64-dimensional input to produce a 2-dimensional output. It is typically followed by a softmax activation function for classification.

Quantum Circuit Layer

The quantum circuit use in this work is a simple variational quantum circuit with a single qubit. Initially, the qubit is put into a superposition state using the Hadamard gate, and then a phase rotation is applied using the parameterized Rz gate. Finally, a measurement is performed to obtain classical information from the quantum system.

Circuit Definition

The variational quantum circuit is defined in the build_circuit method of the Hybrid class as shown below:

```
def_build_circuit(self):
    qc = QuantumCircuit(1)
    qc.h(0)
    qc.rz(self._theta[0].item(), 0)
    qc.measure_all()
return qc
```

i. **Quantum Circuit Initialization**: qc = QuantumCircuit(1)

This initializes a quantum circuit with one qubit.

ii. **Hadamard Gate**: qc.h(0)

The Hadamard gate (H) is applied to qubit 0. The Hadamard gate creates a superposition state from the basis state |0⟩:

$$H|0\rangle \rightarrow \frac{|0\rangle+|1\rangle}{\sqrt{2}} \quad (3)$$

iii. **Parameter-Dependent RZ Gate**: $\mathrm{qc.rz(self.\_theta[\theta].item(),\ \theta)}$

The RZ gate is applied to qubit 0 with a rotation angle that is a trainable parameter. The RZ gate performs a rotation around the Z-axis of the Bloch sphere:

$$Rz(\Theta)=\begin{bmatrix} e^{-i\frac{\emptyset}{2}} & 0 \\ 0 & e^{i\frac{\emptyset}{2}} \end{bmatrix} \quad (4)$$

The self._theta is a PyTorch parameter, and self._theta[θ].item() extracts the numerical value of this parameter.

iv. **Measurement**: qc.measure_all()

The measure_all() method adds measurement operations to all qubits, collapsing their states and producing classical bit outcomes.

Quantum Circuit Components and Operations

Qubit Initialization*:* This represents the initial state of the qubit. In quantum computing, qubits are typically initialized in the ground state, denoted as |0⟩. This is a pure quantum state where the qubit is in the basis state |0⟩.

Hadamard Gate*:* The Hadamard gate (H) is applied to the qubit. The Hadamard gate transforms the basis states |0⟩ and|1⟩into superposition states:

$$H|0\rangle \rightarrow \frac{|0\rangle+|1\rangle}{\sqrt{2}}$$

$$H|1\rangle \rightarrow \frac{|0\rangle-|1\rangle}{\sqrt{2}} \quad (5)$$

After the Hadamard gate, the state of the qubit is:

$$|\psi\rangle = H|0\rangle = \frac{|0\rangle+|1\rangle}{\sqrt{2}} \quad (6)$$

Parameterized RZ Gate*:* The RZ gate is a rotation gate that rotates the qubit around the Z-axis of the Bloch sphere by an angle θ. The rotation angle θ for the RZ gate is a trainable parameter, initially set to a random value scaled by π/2. During training, this parameter is optimized to minimize the loss function, thereby allowing the quantum circuit to learn the most effective rotations for the given classification task. The matrix representation of the RZ gate is:

$$Rz(\Theta) = e^{-i\frac{1}{2}Z} = \begin{bmatrix} e^{-i\frac{\emptyset}{2}} & 0 \\ 0 & e^{i\frac{\emptyset}{2}} \end{bmatrix} \quad (7)$$

When applied to the qubit in state |ψ⟩, the new state becomes:

$$Rz(\theta)|\psi\rangle = Rz(\theta)\left(\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)\right) = \frac{1}{\sqrt{2}}\left(e^{-i\frac{\emptyset}{2}}|0\rangle + e^{i\frac{\emptyset}{2}}|1\rangle\right) \quad (8)$$

Measurement: The measurement operation collapses the quantum state to one of the basis states ($|0\rangle \; or \; |1\rangle$) with certain probabilities. The outcome of this measurement is a classical bit (cbit), which can be either 0 or 1. The probabilities of measuring the qubit in $|0\rangle \; or \; |1\rangle$ depend on the amplitudes of these states in the final quantum state after the RZ gate.

The probability of measuring $|0\rangle$ is: $P(0) = \left|\frac{e^{-i\frac{\emptyset}{2}}}{\sqrt{2}}\right|^2 = \frac{1}{2}$

The probability of measuring | 1⟩ is: $P(1) = \left|\frac{e^{i\frac{\emptyset}{2}}}{\sqrt{2}}\right|^2 = \frac{1}{2}$ (9)

The actual measurement probabilities would depend on the specific values of θ (the trainable parameters) and the interference effects from superposition and rotation.

Quantum Circuit Execution

The quantum circuit is executed using the QiskitAer simulator backend with a specified number of shots to obtain measurement outcomes.

QiskitAer Simulator Backend: Qiskit provides various simulators for executing quantum circuits, including the Aer simulator backend. The Aer simulator allows users to simulate the behavior of quantum circuits on classical computers, providing insights into the quantum states and measurement outcomes.

Execution Process

The execution of the quantum circuit using the QiskitAer simulator backend involves transpiling the circuit, assembling it into a Qiskit quantum object Qobj, submitting it for execution, and obtaining measurement outcomes. By specifying the number of shots, users can adjust the level of accuracy in the simulation results based on their computational resources and requirements. The following are elaboration of the execution process:

Transpilation: Before execution, the quantum circuit is transpiled to map it onto the target quantum hardware or simulator backend efficiently. Transpilation involves decomposing complex gates into elementary gates supported by the target backend and optimizing the circuit for better performance.

Assembly: After transpilation, the transpiled quantum circuit is assembled into a Qobj suitable for execution. The Qobj encapsulates the quantum circuit along with additional execution parameters such as the number of shots (measurement repetitions) and backend options.

Execution: The QiskitAer simulator backend is the chosen backend for which the assembled Qobj is submitted for execution. Using quantum gates and modeling measurements in accordance with the principles of quantum physics, the simulator replicates the evolution of the quantum circuit while it is in operation.

Measurement Outcomes: After the simulation completes, the simulator provides measurement outcomes corresponding to the specified number of shots. Each measurement outcome represents a possible result obtained by measuring the quantum state of the system at the end of the circuit execution.

Specified Number of Shots: The parameter shots specify the number of times the circuit is executed or measured. A higher number of shots provide a more accurate estimation of the quantum state probabilities

and measurement outcomes, reducing statistical fluctuations. By specifying the number of shots, users can control the trade-off between computational resources and the accuracy of the simulation results.

Model Training

The Adam optimizer is used iteratively to update the model parameters in order to minimize the cross-entropy loss function during the model training phase. By performing forward and backward passes over multiple epochs, the model learns to classify the input data effectively.

Adam Optimizer: To train the model, the Adam optimizer is utilized. For neural network training, it is a well-liked optimization method. Adam optimizer combines the benefits of two other extensions of stochastic gradient descent (SGD) – AdaGrad and RMSProp – to provide efficient optimization with adaptive learning rates.

The Adam optimizer updates the parameters of the model according to the following mathematical expressions:

Exponential Moving Average of Gradient

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1)\, g_t$$

Where:

m_t is the exponential moving average of the gradient at time step *t*.

g_t is the gradient at time step *t*.

β_1 is the exponential decay rate for the first moment estimates (typically set to 0.9).

Exponential Moving Average of Squared Gradient

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2)\, g^2 t$$

Where:

v_t is the exponential moving average of the squared gradient at time step *t*.

$g^2 t$ is the element-wise square of the gradient at time step *t*.

β_2 is the exponential decay rate for the second moment estimates (typically set to 0.999).

Bias Correction: Since m_t and v_t are initialized to zero, they are biased towards zero, especially during the initial time steps. To counteract this bias, Adam applies bias correction:

$$\hat{m}_t = \frac{m_t}{1 - \beta^t{}_1}$$
$$\hat{v}_t = \frac{v_t}{1 - \beta^t{}_2} \qquad (10)$$

Where:

t is the time step.

***Parameter Update*:** Finally, the parameters θ of the model are updated using the bias-corrected moving averages and the learning rate.

$$\theta_{t+1} = \theta_t - \frac{\alpha}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t \quad (11)$$

where:

α is the learning rate.

ϵ is a small constant (typically set to10^{-8}) to prevent division by zero.

***Learning Rate*:** The learning rate for the Adam optimizer is set to 0.001. The learning rate determines the step size taken during optimization and affects the convergence speed and stability of the training process. The learning rate is mathematically expressed as:

$$\theta_{t+1} = \theta_t - \frac{\alpha}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t \quad (12)$$

Where:

θ_t represents the parameters of the model at time step *t*.

α denotes the learning rate, which is set to 0.001 in this case.

$\hat{m}_t$ and $\hat{v}_t$ are the bias-corrected moving averages of the gradient and the squared gradient, respectively.

ϵ is a small constant (typically set to 10^{-8})) to prevent division by zero.

***Cross-Entropy Loss Function*:** The cross-entropy loss function is utilized as the objective (or cost) functions during training. Cross-entropy loss is commonly used in classification tasks, penalizing models based on the difference between predicted probabilities and actual labels. In the context of binary classification, the cross-entropy loss function $L(y,\hat{y})$ measures the dissimilarity between the true labels *y* and the predicted probability $\hat{y}$.

For a single example, it can be expressed as:

$$L(y,\hat{y}) = -\left(y \log(\hat{y}) + (1-y) \log(1-\hat{y})\right) \quad (13)$$

Where:

y is the true label (either 0 or 1).

$\hat{y}$ is the predicted probability of the positive class (between 0 and 1).

This expression computes the cross-entropy loss for a single binary classification example. The term *ylog(y)* penalizes incorrect predictions when the true label is 1, and the term $(1-y)\log(1-\hat{y})$ penalizes incorrect predictions when the true label is 0.

Training Loop

Forward Pass: *During* each epoch of training, the model performs a forward pass to compute the predicted outputs for the input data. The forward pass involves propagating the input data through the layers of the model, resulting in predicted class probabilities for each sample.
Given input data X, the model computes the predicted outputs using its current parameters θ.
Mathematically, this can be represented as: $\hat{y}$ = Model (X; θ)

Loss Calculation: After obtaining the predicted outputs, the loss function is calculated to quantify the discrepancy between the predicted probabilities and the ground truth labels. In this case, the cross-entropy loss function is computed between the predicted outputs and the true labels.

After obtaining the predicted output $\hat{y}$, the cross-entropy loss function
$L(y,\hat{y})$ is computed to measure the discrepancy between the predicted probabilities and the true labels y.
The loss function for the entire dataset D can be expressed as:

$$(\theta) = \frac{1}{|D|}\sum(X, y) \in D\ L\ (y, \hat{y}) \tag{14}$$

***Backward Pass (Gradient Descent)*:** Once the loss is computed, the backward pass is performed to compute the gradients of the loss with respect to the model parameters. These gradients indicate the direction and magnitude of parameter updates required to minimize the loss.
The gradients of the loss function with respect to the model parameters θ are computed using back propagation.
This can be represented as:

$$\nabla_\theta \mathrm{L} = \frac{1}{|D|}\sum(X, y) \in D\ \ \nabla_\theta \mathrm{L}(y, \hat{y}) \tag{15}$$

***Parameter Update (Optimization)*:** Using the gradients computed during the backward pass, the Adam optimizer updates the model parameters (weights and biases) to minimize the loss. The optimizer adjusts the parameters in the direction that reduces the loss, scaled by the learning rate.

The Adam optimizer updates the model parameters θ based on the computed gradients and the learning rate η.
The update rule for the parameters is given by: $\theta \leftarrow \theta - \eta$. AdamUpdate ($\nabla_\theta \mathrm{L}$)

Epoch Iteration: The training procedure runs through a number of epochs, each of which is a full run of the training dataset. By repeating the forward and backward passes for each epoch, the model gradually learns to better classify the input data.

Training Evaluation: During training, metrics such as loss values may be logged and monitored to track the progress of the training process. Additionally, the model's performance on a separate validation dataset may be evaluated periodically to prevent over fitting and ensure generalization.

Evaluation

Performance Metrics: The performance of the hybrid quantum-classical model is assessed using various metrics to gauge its effectiveness in classifying benign and scan samples. The primary performance metric utilized is accuracy, which measures the proportion of correctly classified samples out of the total number of samples in the test set.

Testing Process: After the completion of model training, the trained hybrid quantum-classical model is evaluated on a separate test dataset that was not used during training. The testing process involves feeding the test dataset through the trained model and obtaining predicted outputs for each sample. Subsequently, the predicted outputs are compared to the ground truth labels to determine the model's performance.
The primary metric used for evaluation is accuracy, calculated as the ratio of correctly predicted samples to the total number of samples in the test set. Additionally, other performance metrics such as precision, recall, and F1 score may be computed to provide a more comprehensive understanding of the model's performance. Confusion matrices may also be generated to visualize the model's ability to classify benign and scan samples and identify any potential misclassifications or biases. In addition to traditional performance metrics, computational efficiency is evaluated by measuring the time taken for training and evaluation of the model. This evaluation provides insights into the model's scalability and practical feasibility in real-world cybersecurity applications.

Results and Discussion

The hybrid quantum-classical approach seamlessly integrates classical neural network layers with a quantum-classical layer, utilizing the advantages of quantum and classical computing paradigms. Here are the detailed results obtained from this approach:

Accuracy (ACC): The accuracy of the hybrid model is 99.89% on the test dataset, underscoring its capability to correctly classify samples.

Precision (PREC): With a precision of 99.74%, the hybrid model demonstrates its proficiency in minimizing false positives, crucial for ensuring the reliability of cybersecurity systems.

Recall (REC): The hybrid model exhibits a recall of 99.87%, indicating its effectiveness in capturing true positives, a critical aspect in detecting potential threats.

F1 Score: The hybrid model achieves an F1 score of 99.80%, striking a balance between precision and recall, thereby offering robust performance across different evaluation aspects.

Confusion Matrix: A thorough analysis of the model's predictions is given by the confusion matrix with 9928 true positives and 21437 true negatives, shedding light on its performance across benign and scan samples. This visualization enables the identification of any misclassifications or biases present in the model's predictions.

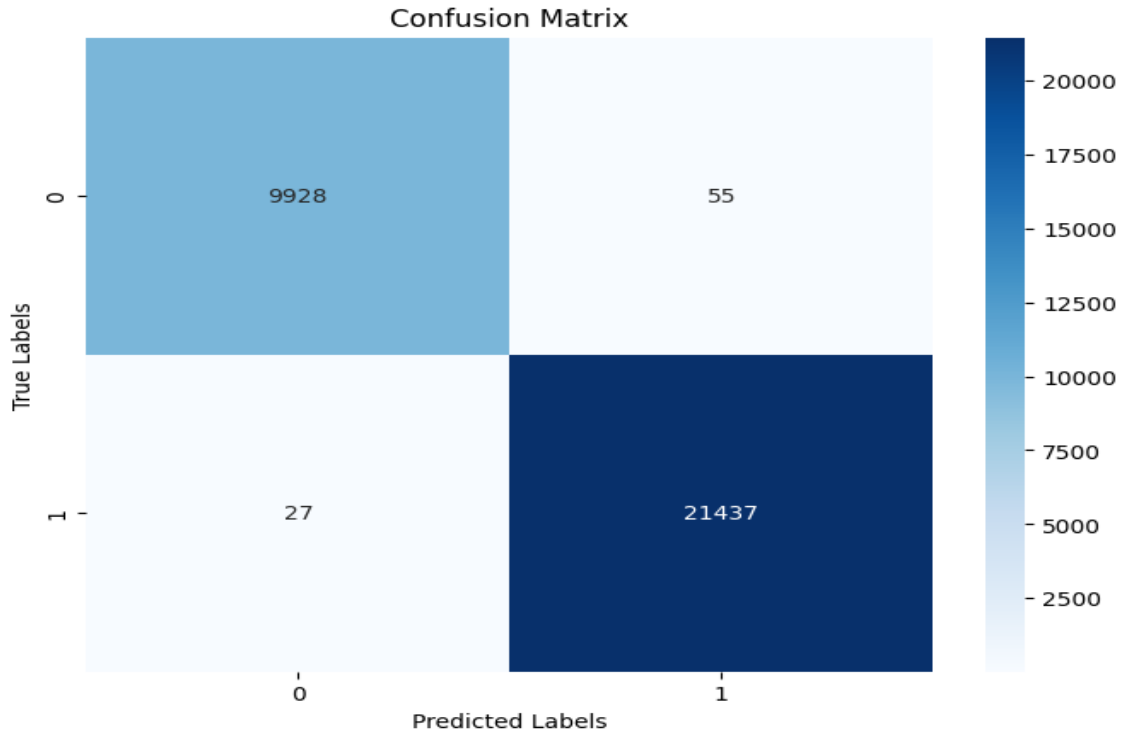

Figure 1. Confusion Matrix for Hybrid-Classical Quantum Model

Training Dynamics: The model's convergence during training is demonstrated in Figure 2, where the epoch vs. loss plot shows how the loss drops over time. On the other hand, figure 3's epoch vs. accuracy plot illustrates how accuracy changes over training epochs, emphasising the model's learning dynamics and improvement patterns.

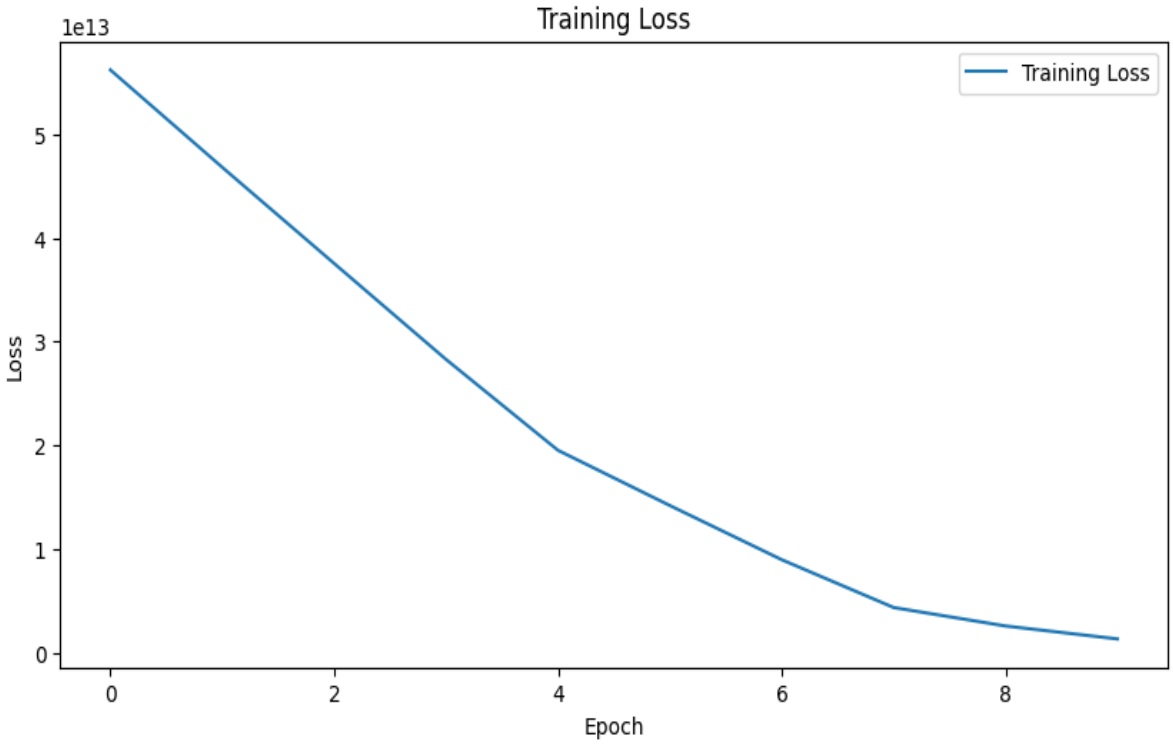

Figure 2. Hybrid Classical Quantum Loss vs Epoch Plot

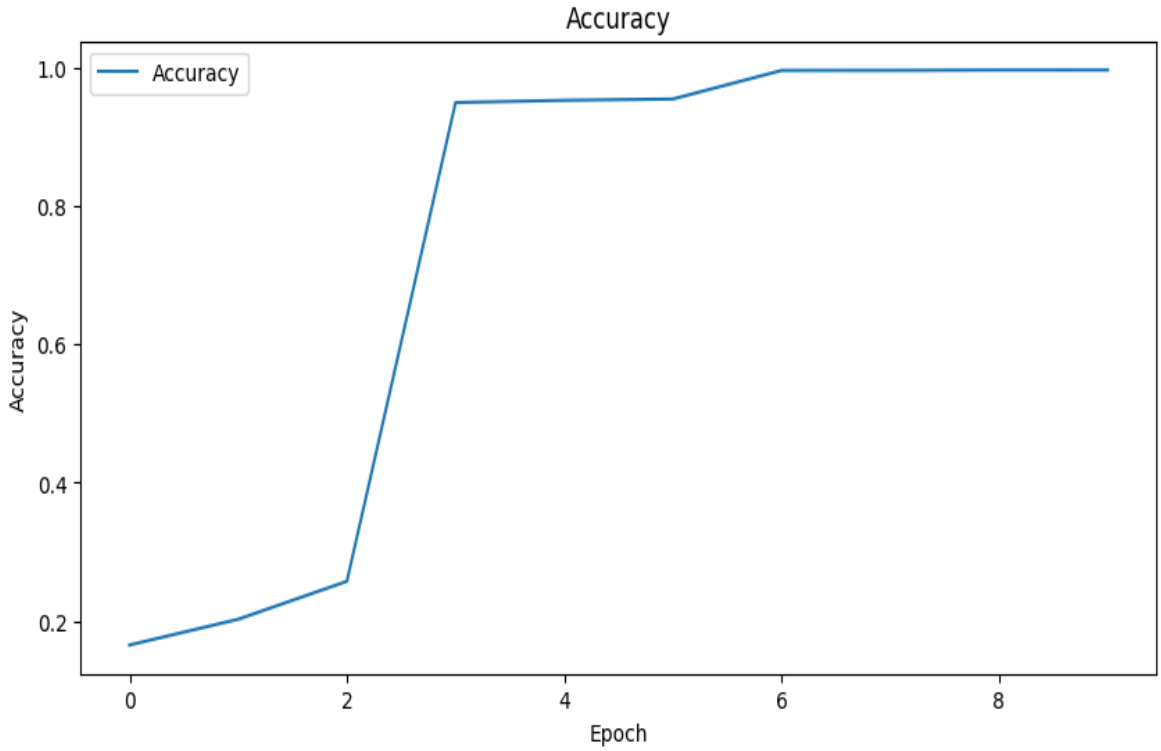

Figure 3. Hybrid Classical Quantum Accuracy vs Epoch Plot

Training Time Analysis for Hybrid Quantum Classical Model

The following findings from the hybrid quantum-classical model's training runtime demonstrate the model's computational effectiveness and performance:

Training Time: 2.3579 seconds

Evaluation Time: 0.0164 seconds

Overall, the hybrid quantum-classical approach seamlessly integrated classical neural network layers with a quantum-classical layer, leveraging both classical and quantum computing paradigms. This approach demonstrated excellent results that outperformed the ones of the traditional models; additionally, it received F1 rating of 99% with outstanding accuracy, precision, and recall.

Conclusion

The study concluded that the Hybrid Quantum-Classical model demonstrates competitive performance, achieving accuracy of approximately 99%. Results reveal that the hybrid approach exhibited superior training and evaluation times. Furthermore, findings show that the Hybrid Quantum-Classical model's efficiency in computational time for both training and evaluation highlights its potential for real-world applications, especially in scenarios where time is critical. The model excels in classification accuracy, precision, and reliability, making it robust and effective options for deployment in operational settings, particularly in cyber security applications where high accuracy and reliability are essential. Based on these results, it was therefore recommended among others, that the Hybrid Quantum-Classical model holds significant promise for the future of anomaly detection in the cyber security space. This particular model is well-suited for real-time applications where prompt threat identification is crucial due to its exceptional efficiency in training and evaluation timeframe. Additionally, for cyber security practitioners, the adoption of the Hybrid Quantum-Classical model could lead to substantial improvements in identifying benign and scan data, thereby enhancing the overall security posture. Implementing this model can provide a more dynamic and responsive approach to anomaly detection, ensuring that threats are identified and mitigated promptly. This capability is essential in today's rapidly evolving cyber threat landscape, where the speed of detection and response can significantly influence the impact of a security breach.

References

1. Shao, C. & School of Mathematics, University of Bristol, UK. (2021, August). Introduction to Quantum computing. *Lecture Notes for the Isogeny-based Cryptography School 2021*. Retrieved from *https://isogenyschool2020.co.uk/schedule/Introduction-to-quantum-computing-lecture-notes.pdf*

2. Callison, A., & Chancellor, N. (2022). Hybrid quantum-classical algorithms in the noisy intermediate-scale quantum era and beyond. *Physical Review. A/Physical Review, A*, *106*(1). https://doi.org/10.1103/physreva.106.010101

3. Calude, C. S., Calude, E., & Dinneen, M. J. (2015). Guest column. *ACM SIGACT News*, *46*(1), 40–61. https://doi.org/10.1145/2744447.2744459

4. Ronald, D. W. (2019, July 19). Quantum Computing: lecture notes. Retrieved from https://arxiv.org/abs/1907.09415

5. McClean, J. R., Romero, J., Babbush, R., & Aspuru-Guzik, A. (2016). The theory of variational hybrid quantum-classical algorithms. *New Journal of Physics*, *18*(2), 023023. https://doi.org/10.1088/1367-2630/18/2/023023

6. Toy, M. (2021). Future networks, services and management. *Springer eBooks*. https://doi.org/10.1007/978-3-030-81961-3

7. Draper-Gil, G., Lashkari, A. H., Mamun, M. S. I., & Ghorbani, A. A. (2016, February). Characterization of encrypted and vpn traffic using time-related. In Proceedings of the 2nd international conference on information systems security and privacy (ICISSP) (pp. 407-414).

8. Bhardwaj, A., Alshehri, M. D., Kaushik, K., Alyamani, H. J., & Kumar, M. (2022). (Retracted) Secure framework against cyber attacks on cyber-physical robotic systems. *Journal of Electronic Imaging*, *31*(06). *https://doi.org/10.1117/1.jei.31.6.061802*

9. Benedict, C. O. (2023). Detecting security anomalies using machine learning for smart homes - ProQuest. Retrieved from https://www.proquest.com/openview/aff4a0d1029f56b2e5e637e5f133dd10/1?pq-origsite=gscholar&cbl=18750&diss=y

10. Gaikwad, D. P., & Thool, R. C. (2015). Intrusion detection system using bagging with partial decision treebase classifier. Procedia Computer Science, 49, 92-98.

11. Bhatia, M., Sharma, V., Singh, P., & Masud, M. (2020). Multi-Level P2P Traffic Classification using Heuristic and Statistical-Based Techniques: A Hybrid approach. *Symmetry*, *12*(12), 2117. *https://doi.org/10.3390/sym12122117*

12. Lin, W., Ke, S., & Tsai, C. (2015). CANN: An intrusion detection system based on combining cluster centers and nearest neighbors. *Knowledge-Based Systems*, *78*, 13–21. https://doi.org/10.1016/j.knosys.2015.01.009

13. https://www.icir.org/enterprise-tracing/download.html.

14. Callison, A., & Chancellor, N. (2022b). Hybrid quantum-classical algorithms in the noisy intermediate-scale quantum era and beyond. *Physical Review. A/Physical Review, A*, *106*(1). *https://doi.org/10.1103/physreva.106.010101*