Quantum-based Multi-Model Machine Learning for Security Data Analysis

Mason Chester
Computer Science Department
Kennesaw State University
United States
mchest15@students.kennesaw.edu

Ethan Barton
Computer Science Department
Kennesaw State University
United States
ebarton6@students.kennesaw.edu

Andrew Liban
Computer Science Department
Kennesaw State University
United States
aliban@students.kennesaw.edu

Andrew Polisetty
Computer Science Department
Kennesaw State University
Marietta, United States
apoliset@students.kennesaw.edu

Yong Shi
Computer Science Department
Kennesaw State University
Marietta, United States
yshi5@kennesaw.edu

***Abstract*— In today's rapidly evolving technology era, cybersecurity threats have become sophisticated, challenging conventional detection and defense. Classical machine learning aids early threat detection but lacks real-time data processing and adaptive threat detection due to the reliance on large, clean datasets. New attack techniques emerge daily, and data scale and complexity limit classical computing. Quantum-based machine learning (QML) using quantum computing (QC) offers solutions. QML combines QC and machine learning to analyze big data effectively. This paper investigates multiple QML algorithms and compare their performance with their classical counterparts.**

Keywords*— Quantum Computing, Quantum-based Machine Learning, Email Phishing, Malware, DDoS

I. INTRODUCTION

As technology continues to reshape our world, it brings heightened cybersecurity risks that affect businesses, individuals, and governments. Identifying and differentiating these attacks is crucial. Classical machine learning (ML) techniques can perform these tasks but are quickly approaching their capacity. As the sophistication of attacks increases a new solution will be needed. Quantum-based Machine Learning (QML) based on Quantum Computing (QC) and ML poses a solution to this problem.

QML aims to harness the power of classical ML algorithms in a quantum state, using the power of quantum properties such as entanglement, interference, and superposition to increase the parallelizability of computation. While quantum advantage has been proven, qubit limitations and error correction affect the computation. The result is that quantum computations are simulated on classical hardware. Researchers claim that QML sometimes performs better in certain situations than classical algorithms.

This research focuses on three types of cybersecurity attacks: Email Phishing, Malware in Android devices, and DDoS attacks. We evaluate the performance of three quantum algorithms: Quantum Neural Networks (QNN), Quantum Logistic Regression (QLR), and Quantum Support Vector Machines (QSVM) with datasets that contain those cybersecurity attacks. We evaluate the performance of each algorithm against its classical contemporary using standard evaluation metrics such as accuracy, loss, precision, recall, etc. We then compare the results from quantum simulations to quantum NISQ devices. Our findings conclude that the future of this technology is bright.

II. RELATED WORK

Quantum computing is a computing paradigm that applies unique properties of quantum physics, such as superposition, entanglement, and interface, to process data and perform computational activities [1-3]. Quantum computing can solve complex issues that regular supercomputers can't handle efficiently. Benefits fields like industrial engineering, industrial engineering, and mathematics.

Machine learning trains a wide range of computational models on existing data to make decisions, predict events, and detect patterns [4-6]. Machine learning can be applied in many fields, such as industrial engineering, healthcare, and biology.

With the advancement of technology, more and more big data are generated daily, and traditional machine learning approaches cannot successfully extract useful information from the data because it requires enormous time and resources. Quantum computing is combined with machine learning to achieve quicker processing and more accurate data analysis [7-9], and QML has lately received a lot of interest from academia and industry.

III. METHODOLOGY

In this section, we will introduce the datasets involved, discuss the implementation of QML algorithms, and analyze their performance compared to their classical counterparts.

A. Datasets

All datasets used in this project are obtained from the open-source machine learning platform Kaggle [10], and they split into 70/30 training/testing sets.

B. Hybrid-Quantum Neural Networks

The hybrid-quantum neural network (HQNN) is based on a fundamental quantum algorithm called variational quantum circuits; these circuits are parametrized, allowing for the parameters to be learned during reptation. HQNN has the added advantage of both classical and quantum algorithms, such as the ability to encode classical data into a quantum state while being able to use classical optimization algorithms and architectural patterns that are easily implemented using popular libraries such as TensorFlow, Keras, Poarch, etc.

Our implementation of a classical model is a Keras sequential model that consists of 3 Dense layers numbering 256, 128, and 64 nodes, respectively. All Dense layers use the Tanh activation; the last layer is a sigmoid-activated

2024 IEEE 4th International Conference on Advanced Learning Technologies on Education & Research (ICALTER) | 979-8-3315-1024-4/24/$31.00 ©2024 IEEE | DOI: 10.1109/ICALTER65499.2024.10819207

979-8-3315-1024-4/24/$31.00 ©2024 IEEE

Authorized licensed use limited to: Kennesaw State University. Downloaded on June 08,2025 at 15:02:48 UTC from IEEE Xplore. Restrictions apply.

classification layer. The classical model uses no dropout or batch normalization layers to make the comparison fair.

Our hybrid-quantum neural network (HQNN) model consists of a classical input and output layer with a hidden quantum layer. The classical input layer takes in 10 features and outputs 7 into the quantum layer using a ReLU activation function. The quantum layer consists of two separate Pennylane templates: the angle embedding layer (encoding our data into a quantum state) and the strongly entangled layer. The strongly entangled layer is trainable, allowing us to learn the parameters through successive repetitions. A strongly entangled layer performs a single qubit rotation followed by an entangler. This entanglement facilitates the efficient transfer of quantum information between qubits. Like our classical model, this model uses a 4-layer architecture that opts for 2 quantum layers. These quantum layers use 7 qubits, allowing us to encode 7 data features into a quantum state. The output layer inputs the quantum layers' resulting measurements and performs binary class prediction via a sigmoid activation. This is all bundled into a Kera sequential model using the same Adam optimizer as the classical model.

As mentioned, we use only a select number of features for both models. We do not use dimensionality-reducing techniques to keep comparisons focused on the algorithms. Instead, we use scalers to scale our data to aid performance. In our classical algorithm, we use the StandardScalar provided by the sklearn library. Because we are embedding our data as angles in a Hilbert space in our quantum algorithm, we use the MinMax scalar from the sklearn library to scale our data to be between 0 and 2π.

We then train our models on these data points after they have been scaled and split recording performance. We then plot our training, testing, and validation loss and accuracy. Confusion matrices are a standard metric used to evaluate model performance, and we use this method to compare our models. After calculating the matrices, we plot both loss and accuracy and the confusion matrix.

C. Quantum Logistic Regression

The logistic regression Model is an important machine learning tool used for classification. It has broad applications in text classification and image analysis. However, processing large amounts of data using the traditional logistic regression model is inefficient. Therefore, it is critical to design an effectual algorithm with the help of quantum computing. Our quantum logistic regression (QLR) model has a non-linear regression function f(x)=(1)/(1+exp(-x)) that has four levels. First, there is a basic entry level with seven spots using ReLU. This quantum part mixes angles and ties things using Pennylane. There are also two more levels with seven quantum dots each that are mixed. They all lead to the last classical level. The training uses TensorFlow Adam to figure out the loss. Quantum logistic regression is quite efficient, even with just seven features put into the quantum dots. This makes quantum-based machine learning very useful when computing and machine learning meet.

As mentioned in previous sections, quantum computing systems solve problems much faster than ordinary computers. The output of the classification provides an interpretation used for labeling a class. Quantum methodology evaluates solutions simultaneously, while standard computers must check things one at a time. Many researchers are devoted to creating algorithms for this quantum technology that work well and resist errors. At the same time, they are trying to make quantum machines more affordable and able to solve more complex problems. However, quantum logistic regression also has its challenges. Specifically, there is a constraint on the feature numbers encoded into Hilbert Space due to limitations on quantum simulations. Researchers have applied the logistic regression model to quantum processors to make predictions based on data features. They encoded only the seven most important features from each data set into quantum states. That led them to use the quantum processing power while working with the hardware restrictions.

D. Hybrid-Quantum Support Vector Machine

The hybrid quantum support vector machine (HQ-SVM) represents an innovative fusion of quantum computing and classical machine learning techniques. Designed to exploit quantum computing benefits like efficient high-dimensional data processing, HQ-SVM retains the robustness of the classical SVM. At its core, the model enhances SVM using quantum state representations for data, particularly in kernel computations, which are critical for SVM performance. HQ-SVM incorporates variational quantum circuits that are parametrized to adapt and learn dynamically.

For the hybrid quantum model, we use data processed through a quantum circuit to train the classical SVM algorithm. Before we can create our quantum data, we must process the data to use it with the quantum aspect of the algorithm. To use our original datasets with the quantum aspect of the algorithm, we use angle embedding, similar to the process in our HQNN model. As discussed in previous sections, quantum embedding encodes the classical data points (image, text, audio) as quantum states within a Hilbert space using a quantum feature map. Angle embedding is a type of quantum embedding that assigns each feature of the dataset a rotation gate with the angle of rotation set by the value of the feature. The type of quantum gate we used is the Pauli-Z gate. This matrix operates on a single qubit. It leaves the basis state $|0\rangle$ unchanged and maps $|1\rangle$ to $-|1\rangle$. On the Bloch sphere, a graphical representation of qubit states, the Pauli-Z gate corresponds to a rotation of 180° about the Z-axis. It effectively flips the qubit from the northern hemisphere to the southern hemisphere or vice versa. Angle embedding is a relatively simple yet versatile quantum encoding method that allows us to use the three datasets with our quantum algorithms. The quantum data are then fed into the classical SVM model for evaluation.

The performance of HQ-SVM is meticulously evaluated against traditional SVM models, focusing on accuracy, loss, and detailed insights from confusion matrices. The 70/30 training/test split allows for a rigorous comparison, and the restriction to 7 features ensures a fair assessment of each model's capabilities. This approach highlights the enhancements brought by quantum computing to the SVM algorithm, demonstrating notable improvements in efficiency and accuracy. The HQ-SVM showcases the potential of quantum computing in enhancing classical machine learning models and sets the stage for future research that could expand the model to handle larger datasets and explore additional quantum algorithms for further advancements.

Authorized licensed use limited to: Kennesaw State University. Downloaded on June 08,2025 at 15:02:48 UTC from IEEE Xplore. Restrictions apply.

E. *Quantum Hardware Resarch*

This section discusses our findings on the current state and plausibility of implementing these algorithms on current quantum hardware. Quantum devices are in an era typically referred to as the noisy-intermediate scale quantum state, or NISQ for short. This acronym refers to the current iteration of quantum computers that cannot properly perform computation without noise interference. This is due to the lack of error-correcting or detection and the high sensitivity to environmental noise such as electromagnetic radiation, et cetera. Companies like Microsoft, Google, and Amazon have invested heavily in developing these technologies.

While all these quantum resource providers are leading technology innovations for this research effort, we could not use them for our research. This is mainly due to the free tier of their service offering little to no resources per month. For example, IBM Qiskit offers 1 hour of computing resources per month. This was unreasonable for our application, as the QNN algorithm alone took 20 hours to train.

IV. Experimental Results

In this section, we demonstrate and discuss the experimental results, comparing the performance of quantum-based machining learning modules with their counterparts on various data sets introduced in section III. In the tables below, data set #1 is the Email Phishing data set, data set #2 is the Malware in Android Devices data set, and data set #3 is the DDoS Attacks data set.

In our experiments, we use Google Colaboratory [11]. This tool allows the execution of Python code. It uses Google Drive to design, implement, and test our classical and quantum machine learning modules on those data sets.

A. *Neural Networks*

All training and validation metrics are measured as an average over 5 epochs. We use accuracy, precision, recall, and F1-score as our preferred comparison metrics. These metrics give us a good general overview of model performance.

In our experiment, we first download all required libraries, mount the Google Drive, create a directory to hold Kaggle datasets and the Kaggle API key, and then import all packages needed. Next, we perform the dataset preprocessing step, where we download the dataset, load it into a data frame, and sample it. We find the top K features from the data frame for both the Quantum and Classical models, where we set K=10. We then scale the data to each model. We use a standard scalar for the classical model, and for the quantum model, we use a min-max scalar. We then split our data into the 70/30 sets.

In the classical neural network implementation, we define network architecture, compile and fit the model to the data, and plot all performance measurements, such as accuracy, loss for training and validation, and confusion matrix.

Table 1. Classical Neural Network training and validation results

Dataset	Training Accuracy	Training Loss	Validation Accuracy	Validation Loss
#1	90.6%	0.222	93.52%	0.186
#2	81.93%	0.3563	81.54%	0.353
#3	99.86%	0.0049	100%	3.7e-4

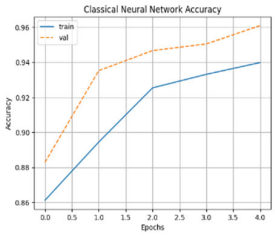

Fig. 1. Classical neural network accuracy changes over Epochs.

Table 2. Classical Neural Network confusion matrix results

Dataset	Accuracy	Precision	Recall	F1-Score
#1	95%	95.31%	94.40%	94.84%
#2	81.75%	78%	98.8%	87.17%
#3	99%	99%	99%	99%

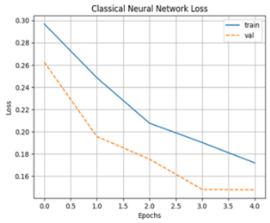

Fig. 2. Classical neural network loss changes over Epochs.

In the quantum neural network implementation, we define the quantum variation circuit, the base circuit for a quantum neural network. This network encodes 7 features into a quantum state and has a two-layer, strongly entangling layer that acts as the network's hidden layer. We then define the entire network using Keres layers with a 7-node input layer and a 1-node output layer without the previous quantum layer in the middle. We compile and fit out the model using the same metrics and optimizer as the classical model. Next, we plot all relevant performance metrics.

Table 3. Quantum Neural Network training and validation results

Dataset	Training Accuracy	Training Loss	Validation Accuracy	Validation Loss
#1	82.62%	.451	85.54%	.428
#2	78.88%	.4329	81.4 %	.3948
#3	98.75%	.054	99.83%	.015

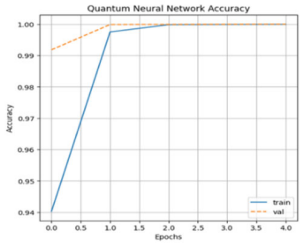

Fig. 3. Quantum neural network accuracy changes over Epochs.

Table 4. Quantum Neural Network confusion matrix results

Dataset	Accuracy	Recall	Precision	F1-Score
#1	90.43%	90.5%	89.81 %	90.15%
#2	81.65%	78.39%	97.77%	87.01%
#3	99.98%	99.99%	99.97%	99.99%

Authorized licensed use limited to: Kennesaw State University. Downloaded on June 08,2025 at 15:02:48 UTC from IEEE Xplore. Restrictions apply.

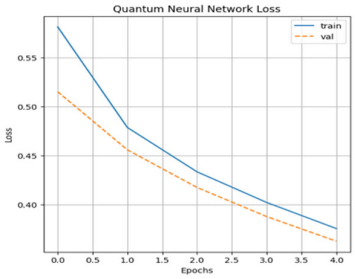

Fig. 4. Quantum neural network loss changes over Epochs.

Given the context of our quantum model, only being able to see 7 of the features, the performance is very promising. While the metrics are like the classical model, training time is exponentially different; for instance, the last dataset took around an hour per epoch to train. In comparison, the classical model took around 4ms per epoch. This is most likely due to quantum simulation. With actual hardware implementations, we should see noticeable improvements. Overall, in the current implementations, quantum neural networks do not perform as well as classical networks, even when large datasets are processed.

As this technology continues to develop, though, it has a bright future in outperforming classical algorithms in terms of performance. Also, it will be a large area of research to determine how to increase training times.

B. Support vector machines

We also design the classical and quantum SVM models and compare their performance. We create the quantum wires and circuit using angle embedding and Pauli-z in the quantum SVM implementation. We load the dataset and clean it up using the K-best feature selection, then apply the MinMaxScaler to place the data points' values in the correct range. We then generate the quantum data using the circuit created earlier in the code, split the data into training and testing datasets, and test the trained model.

Table 5. Classical SVM confusion matrix results

Dataset	Accuracy	Precision	Recall	F1-Score
#1	92.73%	93%	92%	93%
#2	99.94%	100%	100%	100%
#3	98.38%	99%	98%	98%

Table 6. Hybrid Quantum SVM confusion matrix results

Dataset	Accuracy	Precision	Recall	F1-Score
#1	71.33%	76%	71%	70%
#2	99.92%	100%	100%	100%
#3	96.52%	96%	97%	96%

Tables 5 and 6 show the running results of classical SVM and quantum SVM on various data sets. From these two tables, we can see that the quantum SVM does not outperform its counterparts. There can be multiple reasons, such as the data we choose, the quantum gate we select, the data preprocessing step we conduct, etc.

C. Logistic regression

We also design the classical and quantum logistic regression models and compare their performance. In the quantum logistic regression implementation, we choose how many qubits to use and split the data by qubit count. We then select a quantum device and analyze how to gauge errors using mean squared error through our quantum loss function. In our quantum model with layers, each had its own weight and parameters tally. We adjust these figures until they cut down losses as needed.

Table 7. Classical vs. quantum Logistic Regression results

Dataset	Accuracy	Dataset	Accuracy
#2	43.40%	#2	56.26%
#3	97.40%	#3	50.60%

Table 7 shows part of the running results of classical logistic regression vs. quantum logistic regression algorithm on various data sets. As we can see, quantum logistic regression outperforms its counterpart for some data sets.

V. Conclusion and Future Work

Quantum machine learning still has a long way to go before it catches up to classical algorithms' performance, specifically in training time when classical algorithms usually outperform quantum algorithms. This can be equated to the performance of quantum simulations calculated on classical hardware. As the development of quantum hardware devices continues to evolve, we will see dramatic performance increases. As more data features can be encoded and qubit gates are optimized, quantum machine learning will be the next step in the evolution of machine learning.

For future work, we will continue exploring quantum-based machine learning, implement and evaluate more quantum machine learning models, and compare their performance with their traditional counterparts. We will also study what types of data sets suit each quantum machine learning model.

Acknowledgment

The work is partially supported by the U.S. National Science Foundation under award # 2413540. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation

References

[1]. Nielsen, M.A. and Chuang, I.L., 2010. Quantum computation and quantum information. Cambridge University Press.
[2]. Steane, A., 1998. Quantum computing. Reports on Progress in Physics, 61(2), p.117.
[3]. Farhi, E., Goldstone, J., Gutmann, S., Lapan, J., Lundgren, A., Preda, D. A quantum adiabatic evolution algorithm applied to random instances of an NP-complete problem. Science. 2001 Apr 20;292(5516):472-5. doi: 10.1126/science.1057726. PMID: 11313487.
[4]. Murphy, K.P., 2012. Machine learning: a probabilistic perspective. MIT Press.
[5]. Alpaydin, E., 2020. Introduction to machine learning. MIT Press.
[6]. Jordan, M.I. and Mitchell, T.M., 2015. Machine learning: Trends, perspectives, and prospects. Science, 349(6245), pp.255-260.
[7]. Biamonte, J., Wittek, P., Pancotti, N., Rebentrost, P., Wiebe, N. and Lloyd, S., 2017. Quantum machine learning. Nature, 549(7671), pp.195-202.
[8]. Schuld, M., Sinayskiy, I. and Petruccione, F., 2015. An introduction to quantum machine learning. Contemporary Physics, 56(2), pp.172-185.
[9]. Schuld, M. and Killoran, N., 2019. Quantum machine learning in feature Hilbert spaces. Physical review letters, 122(4), p.040504.
[10]. https://www.kaggle.com/datasets
[11]. Google Colab. https://colab.research.google.com/

Authorized licensed use limited to: Kennesaw State University. Downloaded on June 08,2025 at 15:02:48 UTC from IEEE Xplore. Restrictions apply.