Quantum machine learning for intrusion detection of distributed denial of service attacks: A comparative overview

Conference Paper · March 2021

DOI: 10.1117/12.2593297

2 authors:

Esteban Payares
Universidad Tecnológica de Bolívar
4 PUBLICATIONS **1** CITATION
SEE PROFILE

Juan Carlos Martinez Santos
Universidad Tecnológica de Bolívar
55 PUBLICATIONS **168** CITATIONS
SEE PROFILE

Quantum machine learning for intrusion detection of distributed denial of service attacks: a comparative overview

Payares, E., Martinez-Santos, J.

E. D. Payares, J. C. Martinez-Santos, "Quantum machine learning for intrusion detection of distributed denial of service attacks: a comparative overview," Proc. SPIE 11699, Quantum Computing, Communication, and Simulation, 116990B (5 March 2021); doi: 10.1117/12.2593297

SPIE.

Event: SPIE OPTO, 2021, Online Only

Quantum Machine Learning for Intrusion Detection of Distributed Denial of Service Attacks: A Comparative Overview

E.D. Payares and J.C. Martinez-Santos

Universidad Tecnológica de Bolívar, Campus Tecnologico, Cartagena de Indias, Colombia

ABSTRACT

In recent years, we have seen an increase in computer attacks through our communication networks worldwide, whether due to cybersecurity systems' vulnerability or their absence. This paper presents three quantum models to detect distributed denial of service attacks. We compare Quantum Support Vector Machines, hybrid Quantum-Classical Neural Networks, and a two-circuit ensemble model running parallel on two quantum processing units. Our work demonstrates quantum models' effectiveness in supporting current and future cybersecurity systems by obtaining performances close to 100%, being 96% the worst-case scenario. It compares our models' performance in terms of accuracy and consumption of computational resources.

Keywords: Quantum computing, Quantum machine learning, Quantum Processing Units, Cybersecurity, DDoS attacks, Smart Intrusion Detection Systems

1. INTRODUCTION

Quantum computing is an area of computing touted with achieving incredible results for factoring and unordered search problems.[1] However, after many years, the scientific development achieved so far in quantum technologies is beginning to pay-off, starting what could be called a quantum revolution in many other applications. During those years of research, many powerful algorithms and applications appear for quantum hardware. In particular, the potential of quantum computers to enhance machine learning is genuinely exciting[2] because actually, these two technologies have the potential to alter the way computing addresses previously unsustainable problems.[3]

Driven by increasing computing power and algorithmic advances, machine learning techniques have become powerful tools for finding data patterns. Quantum systems produce outlier patterns that classical methods cannot make efficiently. It is then reasonable to postulate that quantum computers can outperform classical computers in machine learning tasks.[4] Therefore, Quantum machine learning (QML) can enhance conventional machine learning (ML) applications.

These technologies in the noisy intermediate-scale quantum (NISQ) era explore the potential for developing systems that conclude with the search for advanced applications by quantum technologies. Modern ML has provided us with generative modeling techniques that are perfectly suited for the emerging landscape of NISQ hardware.[5] An excellent example of this is the development of security systems against computer threats. Because quantum technologies allow us to solve problems and take us to a point where the information and communication technologies will be affected by new applications, new threats may arise. However, Quantum computing holds great promise in many areas, such as medical research, artificial intelligence, weather forecasting, etc. It also poses a significant cybersecurity threat, requiring a change in how we encrypt our data[6] because two decades ago, we learned that the quantum paradigm implies that practically all the deployed public-key cryptography will entirely break by a quantum computer.[7] For this reason, it is crucial to find out how different types of quantum security systems behave in the face of this type of problem in an industry that is slowly transitioning to making use of quantum technologies.

Further author information: (Send correspondence to E.D.P.)
E.D.P.: E-mail: epayares@utb.edu.co.
J.C. M-S.: E-mail: jcmartinezs@utb.edu.co.

Quantum Computing, Communication, and Simulation, edited by Philip R. Hemmer, Alan L. Migdall, Proc. of SPIE Vol. 11699, 116990B · © 2021 SPIE
CCC code: 0277-786X/21/$21 · doi: 10.1117/12.2593297

2. METHODS

We used three approaches for the classification model with the chosen dataset for the development of this work. For the first one, the QSVM model provided by QiskitAqua, which is the package for building algorithms and applications by Qiskit (IBM).[8] For the two other models, the specialized QML framework PennyLane (Xanadu Quantum Technologies Inc.)[9] The general steps applied in this methodology are described below and summarized in Figure 1.

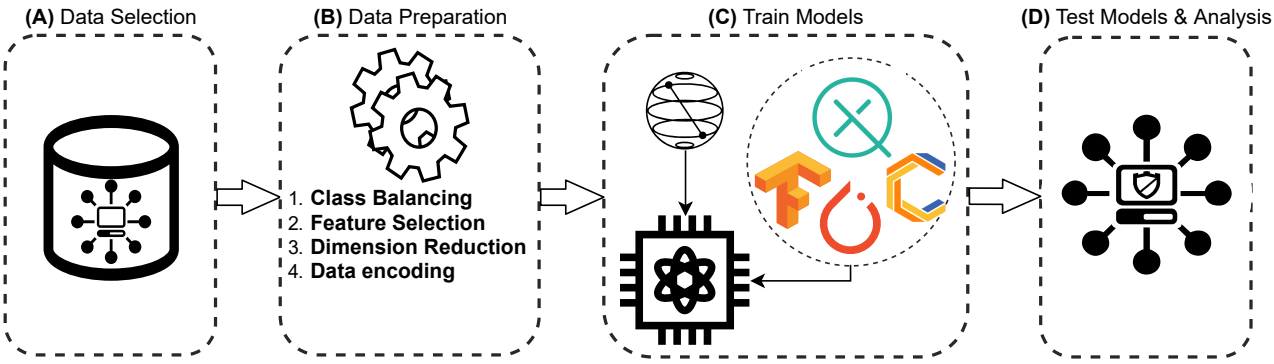

Figure 1. Overview of the methodology process. (A) Search and selection of the dataset. (B) Data preprocessing to improve models performance. (C) Construction and training of the models. (D) Generation of results, analysis and final conclusions.

2.1 Data Description

To build the models is necessary to have a dataset available. For this case study, the suggested data set to train our models is DDoS Evaluation Dataset (CIC-DDoS2019), which contains benign and the most up-to-date common DDoS attacks, which resembles the actual real-world data (PCAPs). It also includes the network traffic analysis results using CICFlowMeter-V3 with labeled flows based on the time stamp, source, and destination IPs, source and destination ports, protocols, and attack.[10] We use a sample of 38 features and 2950 data points, which contains two classes: Benign, which means that the data does not represent any threat, and Simple Service Discovery Protocol (SSDP) type DDoS attacks.

2.2 Preprocessing Data

Data preparation for the generation of results carries out three steps, class balancing, features selection & dimension reduction, and data encoding. This process obtains the best possible results, depending on each particular model's specifications.

2.2.1 Class Balancing

Since we target an intrusion detection problem, our data represents network traffic and event logs per machine. As in real-world situations, this data is imbalanced or evenly distributed. To solve this problem, we implemented undersampling and oversampling algorithms, practically of the random type.

2.2.2 Feature Selection

Initially, our selected dataset had 80 features. However, we eliminate those features that interfere with the performance of the models. This process consists of five different approaches:

- Remove features with a missing percentage more significant than a specified threshold.
- Remove features with a single unique value.
- Remove collinear variables with a correlation more significant than a specified correlation coefficient.

- Remove features with 0.0 feature importance from a gradient boosting machine (GBM) and remove those that do not contribute to specified cumulative feature importance from the GBM.

2.2.3 Normalization

We used two types of normalizations for the data. On the ensemble model side, we scaled the data from $-\pi$ to π, while the other models used a -1 to 1 scaling to improve the performance of the algorithms.

2.2.4 Principal Component Analisys (PCA)

As described above, we implemented a careful feature extraction process. However, for the construction of our models, we considered that our data's high dimensionality could be a problem due to the limitations of the Frameworks. The overall PCA is a widely used option to deal with large datasets when processing data for the use of machine learning models, a multivariate technique that analyzes the data table in which several interrelated quantitative dependent variables describe the observations. Its objective is to extract the critical information from the table, represent it as a set of new orthogonal variables called principal components, and show the similarity pattern of the observations and variables as points.[11] This process reduces our data to only two features, as show Figure 2.

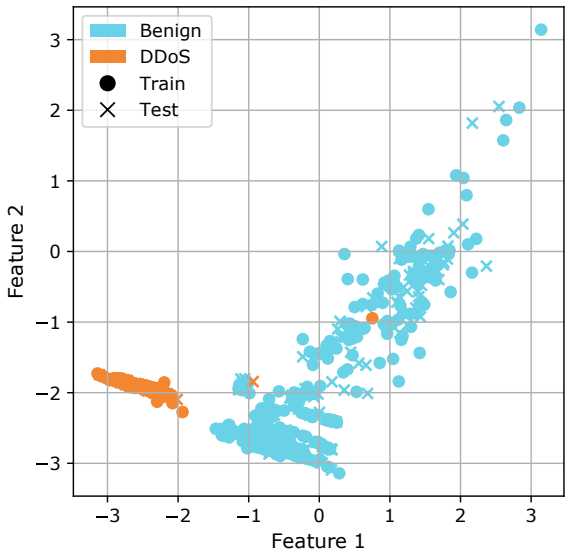

Figure 2. Data used.

2.2.5 Data Encoding

In general, there are many methods, but they all have the same principle: represent classical data as quantum states in a high-dimensional Hilbert space using a quantum feature map.[12] It takes a classical datapoint x and encodes it into a set of gate parameters in a quantum circuit, creating a quantum state $|\psi_x\rangle$.

In this particular case, we used the angle embedding method, which consists of encoding a set of N features into the rotations angles of n qubits, where $N \leq n$, using the Rx Gate that is one of the Rotation Operators. The Rx gate is a single-qubit rotation around the angle θ (radians) along the x-axis of the Bloch Sphere (1).

$$R_x(\theta) = \begin{bmatrix} \cos\left(\frac{\theta}{2}\right) & -i\sin\left(\frac{\theta}{2}\right) \\ -i\sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{bmatrix} \tag{1}$$

2.3 Models

The models used in this work are practical adaptations using algorithms, concepts, and the necessary tools to obtain a result through quantum architectures, which will be briefly described in this section, to be as replicable as possible.

2.3.1 Quantum Support Vector Machine

Support vector machines (SVM) are a type of supervised machine learning algorithms for binary classifications problems. this method as is widely known, maps the data into a higher dimensional input space and constructs an optimal separating hyperplane in this space. This basically involves solving a quadratic programming problem. Least Squares SVM (LS-SVM) is a version of SVM.[13] It approximates the hyperplane finding procedure of SVM by solving the linear equation (2).

$$\left[\begin{array}{cc|c} 0 & \vec{1}^T & b \\ \vec{1} & \mathbf{K}+\gamma^{-1} & \vec{\alpha} \end{array}\right] = \begin{bmatrix} 0 \\ \vec{y} \end{bmatrix} \tag{2}$$

The quantum version of SVM performs the LS-SVM algorithm using quantum computers.[14,15] It calculates the kernel matrix using the quantum algorithm for the inner product on quantum random access memory. It solves the linear equation using a quantum algorithm for solving linear equations[15] and performs the classification of query data using the trained qubits with a quantum algorithm.[14] However, we can only use this method if data is a coherent superposition. Since this is not the case, for convenience, we decided to implement the approach presented in,[3] which proposes a classifier that processes data that is provided classically and uses the quantum state space as feature space, through the QSVM algorithm of the Qiskit framework.

2.3.2 Ensemble Model

The quantum property of superposition uses a specific framework to store sets of parameters, thereby generating an ensemble of quantum classifiers computed in parallel. The idea stems from classical ensemble methods, where one attempts to build a more robust model by averaging the results of several different models.[16]

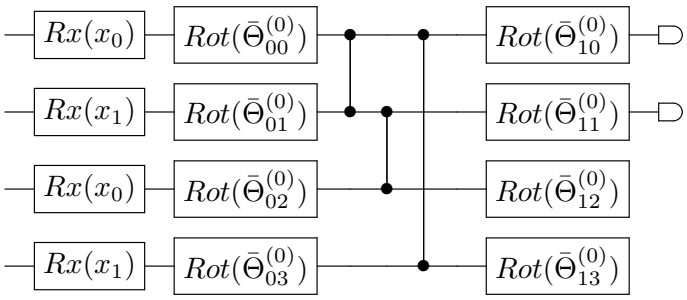

Figure 3. Circuit for QPU-0.

As is shown in Figures 3 and 4, for the circuits for both QPUs, we used four qubits devices from the PennyLane default qubit and the PennyLane-Cirq plugin simulator. The data is input using the angle embedding process briefly explained in 2.2.5. Afterward, each of the circuits is enacted for each system with a particular set of trainable parameters. The output of both circuits is a Pauli Z (3) operator for measurement on two qubits. The result is passed into a softmax function, which results in two two-dimensional probability vectors corresponding to the two classes. Finally, the ensemble model takes the QPU that is most positive in its prediction (i.e., the class with the highest average likelihood over all QPUs) and uses it to make a prediction.

$$\sigma_z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \tag{3}$$

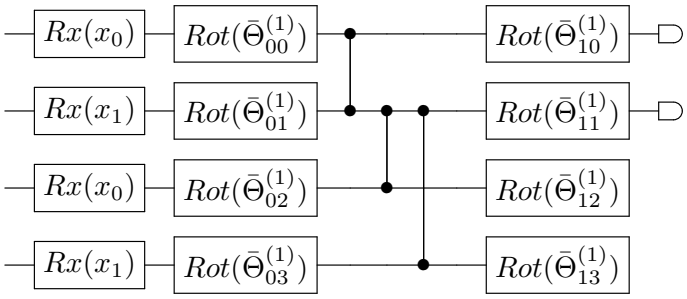

Figure 4. Circuit for QPU-1.

2.3.3 Hybrid Quantum-Classical Neural Network

Near-term quantum processors are still relatively small and noisy. Therefore quantum models cannot disassociate and generalize quantum data using quantum processors alone. NISQ processors would need to operate in conjunction with classical co-processors to be efficient.[17] Many experimental proposals for noisy intermediate quantum systems include training a parameterized quantum circuit with a classical optimization loop. These hybrid quantum-classical algorithms are popular for quantum simulation, optimization, and machine learning applications.[18]

Our model consists of three layers, two Classical Dense Layers, and one Quantum Layer. The process is a little bit different from the other two models. Here we created two qubits, which represent two bias neurons, corresponding to the two features evaluated from our post-processed data fed to the quantum layer via angle embedding (2.2.5). This layer consists of single-qubit rotations and entanglers, which follows the circuit-centric classifier design.[19] In the same way as 2.3.2, two Pauli Z operators with a matrix defined in (3), for measurements are created.

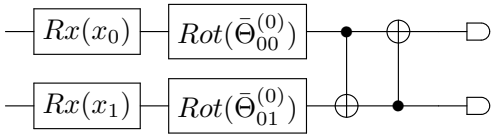

Figure 5. Quantum layer

The general architecture of this model consists, as mentioned above, of three layers. The first layer is a Rectified Linear Unit (ReLU) function. The second is the described quantum variational circuit of two qubits shown in Figure 5. The last layer consists of a Softmax function.

3. RESULTS AND DISCUSSION

The results* obtained in this study give us a clear vision of the behavior of quantum models concerning cybersecurity problems and how addressing them through QML can be successful, depending on the application. However, it is essential to note some points. Our results are summarized in Table 1.

Table 1. Summary of results.

Model	Accuracy	Recall	Precision	F-score	Missclass	CPU Time	Memory usage
ENSEMBLE	0.96836	0.96832	0.97024	0.96832	0.0316	32.1 s	546.69 MiB
QSVM	0.99661	0.99660	0.99661	0.99661	0.0034	122.4×10^3 s	5.73 GiB
H-QNN	0.99887	0.99887	0.99887	0.99887	0.0011	5.265×10^3 s	520.05 MiB

*Code available at: https://github.com/PCesteban/QMLforDDoS

Figures 6, 7, 8, 9, and 10 show the predictions and confusion matrices for each model. The metrics presented in Table 1 represent the evaluation of each model. Accuracy is the percentage of correct predictions for the test data. Precision is the fraction of relevant examples (true positives) among all examples that belong to a given class. Recall is the fraction of examples that belong to a class concerning all examples that belong to the class. F-score is a combination of the recall and precision metrics to give an overall accurate picture of the model's evaluation in question. Finally, Missclass is the percentage of incorrect predictions.

Below we present some considerations to be taken into account to validate our results for each of the models. The models do not behave in the same way with the same test data, and it was necessary to adjust them to obtain better results.

Normalization or scaling of the data had a significant influence, as expected, on the results. For instance, in the case of the QSVM model, we evidenced an increase in performance from 89% to 99.6% and an approximate 50% reduction in CPU time.

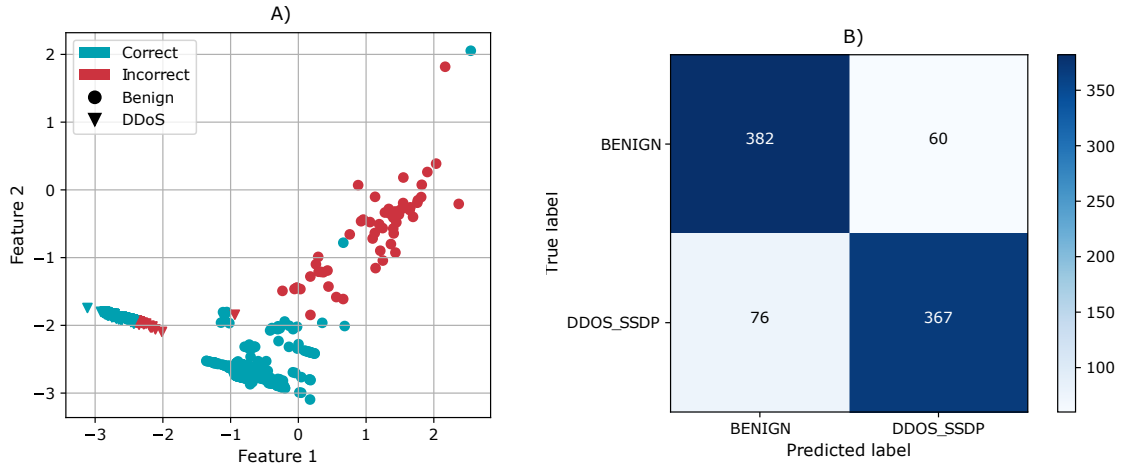

Figure 6. QPU-0 model. A) Predictions for the test dataset. B) Confusion matrix for the test dataset.

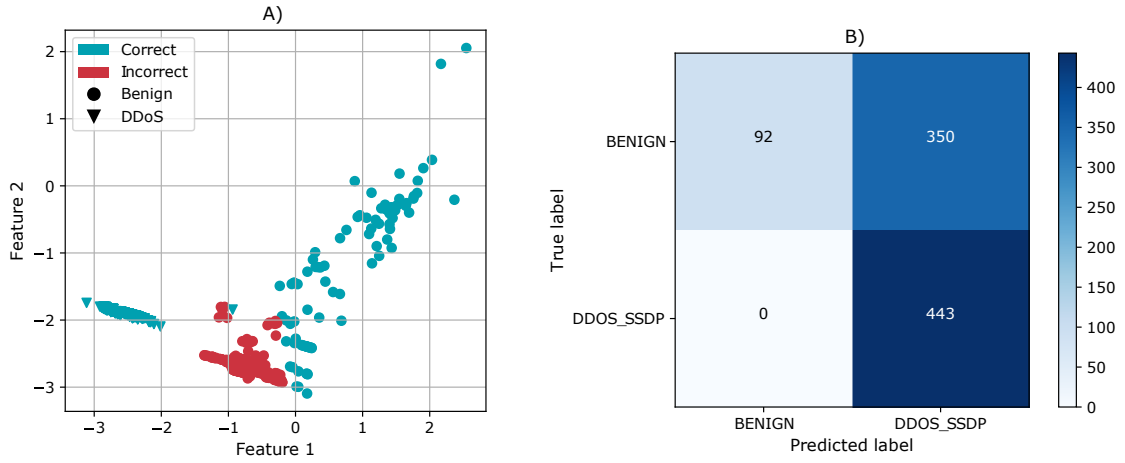

Figure 7. QPU-1 model. A) Predictions for the test dataset. B) Confusion matrix for the test dataset.

As described in 2.3.2, the ensemble model results are the choices made by the algorithm depending on the performance of each QPU in parallel. However, it is essential to see what influence each QPU had on the final result. For that reason, we present a record of it that can see in Figures 6 and 7. On the one hand, the QPU-0 model performed acceptably with an accuracy of approximately 84.6%, correctly classifying most of the data with the "Benign" class, while on the other hand, the QPU-1 model, although performed poorly with an accuracy of 60.4%, entirely classified all the data labeled as a "DDoS" type threat.

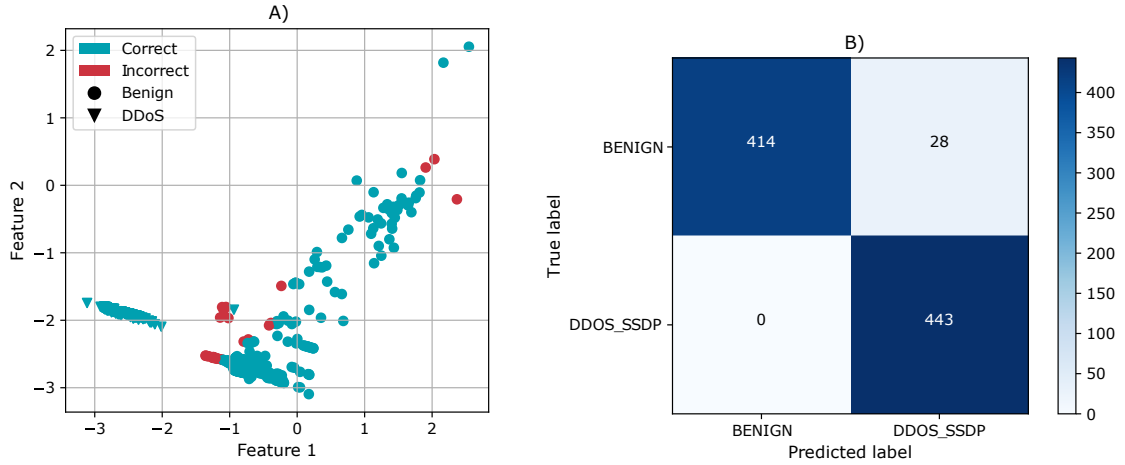

Figure 8. Ensemble model (QPU-0||QPU-1). A) Predictions for the test dataset. B) Confusion matrix for the test dataset.

On the computational resource consumption side, we found that the most expensive model is the QSVM model. It is normal since this model focuses on being advantageous when the feature vector kernel computation is not classically efficient, presenting an increase of more than 100% over the resources consumed by the ensemble model, which was the most efficient.

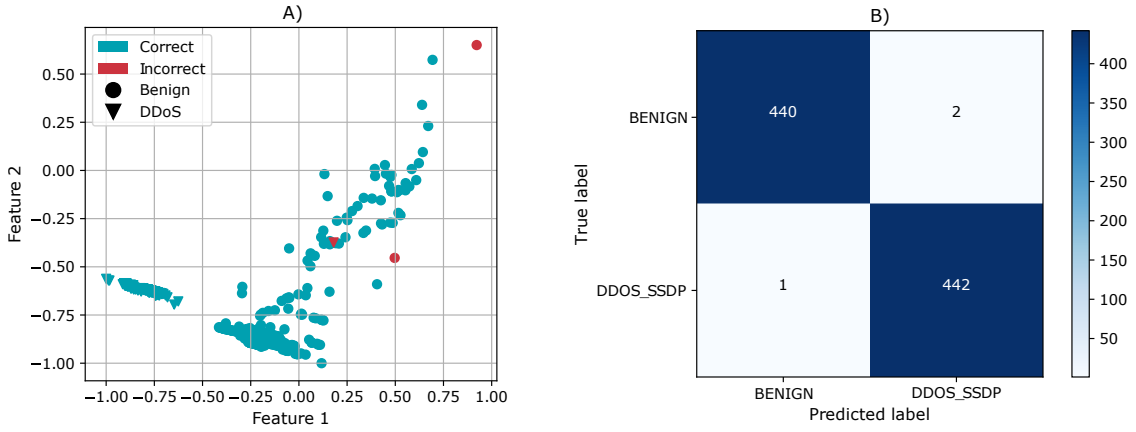

Figure 9. QSVM model. A) Predictions for the test dataset. B) Confusion matrix for the test dataset.

Finally, as shown in Table 1, the best performing model was the H-QNN, in terms of accuracy in proportion to computational efficiency. Note that we have not considered the possibility of overfitting for each of the models. It is possible because, parametrically, the models presented a quite robust behavior with the test data.

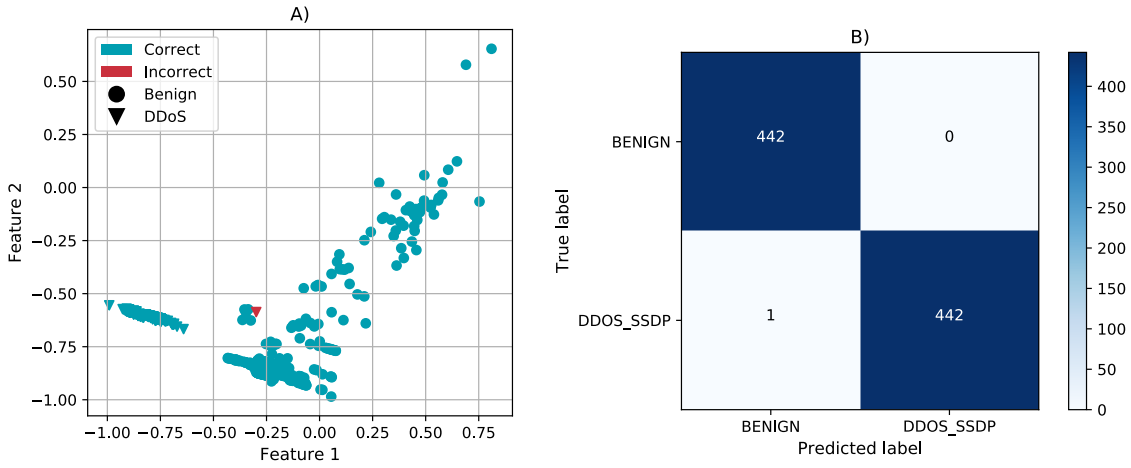

Figure 10. H-QNN model. A) Predictions for the test dataset. B) Confusion matrix for the test dataset.

4. CONCLUSIONS

Although quantum technologies are gaining momentum in recent years, there is still a long way to go. Therefore there is great potential in the quantum methods for artificial intelligence. However, the gap for the developed approaches and applications to solve conventional computation problems is still vast.

This work shows that detecting DDoS type threats is possible using quantum machine learning methods with high accuracy. Compared to other similar studies of the classical counterpart, we evidence a considerable performance improvement. Three quantum models were presented, trained, and tested, each with excellent results in their particular considerations. Future work includes:

- The use of more complex data.
- Theorized traffic representations of a quantum internet.
- A larger dataset to increase the reliability of the results.

ACKNOWLEDGMENTS

The authors would like to thank the research department of Universidad Tecnologica de Bolivar for supporting this project and the team of developers and community members of Qiskit and PennyLane for providing excellent frameworks, demos, and working environments that make quantum computing more accessible.

REFERENCES

[1] Havenstein, C., Thomas, D., and Chandrasekaran, S., "Comparisons of Performance between Quantum and Classical Machine Learning," *SMU Data Science Review* **1** (Jan. 2019).

[2] Killoran, N., Bromley, T. R., Arrazola, J. M., Schuld, M., Quesada, N., and Lloyd, S., "Continuous-variable quantum neural networks," *Physical Review Research* **1**, 033063 (Oct. 2019).

[3] Havlíček, V., Córcoles, A. D., Temme, K., Harrow, A. W., Kandala, A., Chow, J. M., and Gambetta, J. M., "Supervised learning with quantum-enhanced feature spaces," *Nature* **567**, 209–212 (Mar. 2019).

[4] Biamonte, J., Wittek, P., Pancotti, N., Rebentrost, P., Wiebe, N., and Lloyd, S., "Quantum machine learning," *Nature* **549**, 195–202 (Sept. 2017).

[5] Torlai, G. and Melko, R. G., "Machine-Learning Quantum States in the NISQ Era," *Annual Review of Condensed Matter Physics* **11**, 325–344 (Mar. 2020).

[6] "The Quantum Computing Impact on Cybersecurity | QuantumXC."
[7] Mosca, M., "Cybersecurity in an era with quantum computers: Will we be ready?," *IEEE Security Privacy* **16**(5), 38–41 (2018).
[8] Abraham, H., AduOffei, Agarwal, R., Akhalwaya, I. Y., Aleksandrowicz, G., Alexander, T., Amy, M., Arbel, E., Arijit02, Asfaw, A., Avkhadiev, A., Azaustre, C., AzizNgoueya, Banerjee, A., Bansal, A., Barkoutsos, P., Barnawal, A., Barron, G., Barron, G. S., Bello, L., Ben-Haim, Y., Bevenius, D., Bhobe, A., Bishop, L. S., Blank, C., Bolos, S., Bosch, S., and *et. al.*, "Qiskit: An open-source framework for quantum computing," (2019).
[9] Bergholm, V., Izaac, J., Schuld, M., Gogolin, C., Alam, M. S., Ahmed, S., Arrazola, J. M., Blank, C., Delgado, A., Jahangiri, S., McKiernan, K., Meyer, J. J., Niu, Z., Száva, A., and Killoran, N., "PennyLane: Automatic differentiation of hybrid quantum-classical computations," *arXiv:1811.04968 [physics, physics:quant-ph]* (Feb. 2020). arXiv: 1811.04968.
[10] "DDoS 2019 | Datasets | Research | Canadian Institute for Cybersecurity | UNB."
[11] Abdi, H. and Williams, L. J., "Principal component analysis: Principal component analysis," *Wiley Interdisciplinary Reviews: Computational Statistics* **2**, 433–459 (July 2010).
[12] Lloyd, S., Schuld, M., Ijaz, A., Izaac, J., and Killoran, N., "Quantum embeddings for machine learning," (2020). arXiv: 2001.03622.
[13] Suykens, J. and Vandewalle, J. *Neural Processing Letters* **9**(3), 293–300 (1999).
[14] Rebentrost, P., Mohseni, M., and Lloyd, S., "Quantum Support Vector Machine for Big Data Classification," *Physical Review Letters* **113**, 130503 (Sept. 2014).
[15] J., A., Adedoyin, A., Ambrosiano, J., Anisimov, P., Bärtschi, A., Casper, W., Chennupati, G., Coffrin, C., Djidjev, H., Gunter, D., Karra, S., Lemons, N., Lin, S., Malyzhenkov, A., Mascarenas, D., Mniszewski, S., Nadiga, B., O'Malley, D., Oyen, D., Pakin, S., Prasad, L., Roberts, R., Romero, P., Santhi, N., Sinitsyn, N., Swart, P. J., Wendelberger, J. G., Yoon, B., Zamora, R., Zhu, W., Eidenbenz, S., Coles, P. J., Vuffray, M., and Lokhov, A. Y., "Quantum Algorithm Implementations for Beginners," *arXiv:1804.03719 [quant-ph]* (Mar. 2020). arXiv: 1804.03719.
[16] Abbas, A., Schuld, M., and Petruccione, F., "On quantum ensembles of quantum classifiers," *arXiv:2001.10833 [quant-ph]* (Jan. 2020). arXiv: 2001.10833.
[17] Broughton, M., Verdon, G., McCourt, T., Martinez, A. J., Yoo, J. H., Isakov, S. V., Massey, P., Niu, M. Y., Halavati, R., Peters, E., Leib, M., Skolik, A., Streif, M., Von Dollen, D., McClean, J. R., Boixo, S., Bacon, D., Ho, A. K., Neven, H., and Mohseni, M., "TensorFlow Quantum: A Software Framework for Quantum Machine Learning," *arXiv:2003.02989 [cond-mat, physics:quant-ph]* (Mar. 2020). arXiv: 2003.02989.
[18] McClean, J. R., Boixo, S., Smelyanskiy, V. N., Babbush, R., and Neven, H., "Barren plateaus in quantum neural network training landscapes," *Nature Communications* **9** (Nov. 2018).
[19] Schuld, M., Bocharov, A., Svore, K. M., and Wiebe, N., "Circuit-centric quantum classifiers," *Physical Review A* **101** (Mar 2020).