Received 16 September 2023, accepted 25 September 2023, date of publication 9 October 2023, date of current version 12 October 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3322723

RESEARCH ARTICLE

DEQSVC: Dimensionality Reduction and Encoding Technique for Quantum Support Vector Classifier Approach to Detect DDoS Attacks

AHMAD ALOMARI AND SATHISH A. P. KUMAR, (Senior Member, IEEE)
Department of Electrical Engineering and Computer Science, Cleveland State University, Cleveland, OH 44115, USA

Corresponding author: Sathish A. P. Kumar (s.kumar13@csuohio.edu)

This work was supported by the National Science Foundation under Grant OMA 2231377.

ABSTRACT Distributed Denial of Service (DDoS) attacks pose a significant threat to the security of networking systems, as they can cause widespread disruption and even bring down entire distributed systems platforms. In this paper, we propose an approach called the DEQSVC that leverages quantum machine learning techniques to detect DDoS attacks with high accuracy. The DEQSVC integrates the most efficient dimensionality reduction techniques, a robust feature map method, and an efficient kernel estimation technique to improve data encoding, learning process, and detection accuracy. To evaluate the performance of the proposed DEQSVC, we conducted simulations using the Qiskit platform and executed the approach on an IBM quantum computer. Our results demonstrate that the DEQSVC outperforms several benchmark algorithms commonly used in intrusion detection systems. Specifically, the DEQSVC achieves a detection accuracy of 99.49, indicating its effectiveness as a highly accurate and efficient method for detecting DDoS attacks.

INDEX TERMS DEQSVC, QSVM, quantum machine learning, entanglement, encoding, DDoS attacks, cybersecurity, LDAP protocol.

I. INTRODUCTION AND BACKGROUND INFORMATION

In recent years, there has seen a substantial number of security attacks on networking architecture and distributed systems. These attacks are due to both quantum and classical security attacks [1], [6], [10]. One of the most prominent attacks that threaten cybersecurity is the DDoS attacks [2], [9], [32]. The idea of these attacks is to use multiple machines to send an extensive number of malicious requests to capture a network or harm the distributed system components.

Many quantum machine learning solutions have been developed to counter DDoS attacks, but these solutions have performance gaps that make them unreliable in detecting these attacks. For example, some of the existing quantum solutions suffer from qubit decoherence. This limitation generates an insufficient correlation factor between the entangled qubits, decreasing their performance in detecting cybersecurity attacks [3], [4], [5]. Also, many quantum machine learning models need to be trained continuously and they have a notable low detection performance when the same classifier is applied to several attacks [17], [20]. Moreover, many quantum machine learning techniques have complex tasks that may take a long time to complete, and computational errors may occur [6], [7], [11].

We intend to address the above-mentioned problems by using the proposed DEQSVC, which integrates effective dimensionality reduction techniques, a robust feature map encoding method, and an efficient Quantum Support Vector Classifier (QSVC) to detect DDoS attacks [8]. The proposed DEQSVC uses a ZZFeatureMap to encode the classical dataset into quantum states and entangle them to capture the complex relationship between the input data features. Then the entangled quantum states are passed to the Pegasos QSVC to detect DDoS attacks. The PQSVC algorithm uses a quantum linear systems solving algorithm to efficiently estimate the kernel function without explicitly computing it, allowing the algorithm to handle high-dimensional feature spaces and large datasets more efficiently [33], [34]. This

The associate editor coordinating the review of this manuscript and approving it for publication was Wei Huang.

can potentially lead to improved classification accuracy by finding the hyperplane that maximally separates the two classes.

In the proposed work, we intend to detect DDoS attacks that occur on the Lightweight Directory Access Protocol (LDAP). The LDAP protocol is a lightweight client-server cross-platform that is used for directory service authentication [21]. The reason for selecting this protocol is that it is widely adopted and highly susceptible to DDoS attacks. This is because the LDAP protocol is built on top of the TCP/IP protocol and serves to set up authentication policies for a large number of users. By leveraging the LDAP protocol, it becomes possible to authenticate all employees in an organization simultaneously, eliminating the need for individual authentication policies [2], [21], [22]. As a result, it is crucial to implement robust security approaches for the LDAP protocol.

To verify the performance of the proposed DEQSVC we conducted experiments on the IBM quantum computer and compared it with traditional Quantum Support Vector Machine (QSVM) approaches and other quantum machine learning techniques [11], [15], [17]. The results show the superiority of the proposed DEQSVC over the other approaches.

The following are the contributions of our work.

1) Developed an approach that integrates the most efficient dimensionality reduction techniques with quantum entanglement to improve data encoding and detection accuracy.
2) Applied a robust feature map method to encode the classical data into quantum data.
3) Demonstrated that DEQSVC enhances the cybersecurity of communication protocols by achieving a significant detection accuracy of DDoS attacks, outperforming well-known DDoS attacks detection models.
4) Conducted performance evaluation experiments to evaluate DEQSVC on the genuine IBM quantum computer compared to the Qiskit simulation environment.

The remainder of this paper is organized as follows: In section II, we survey the most relevant existing quantum machine learning security solutions against DDoS attacks, pointing out their pros and cons, and how they counter these attacks. Section III describes the methodology of our QSVC approach, including the dataset, standardization and scaling techniques, dimensionality reduction technique, quantum encoding method, and the training process of DEQSVC to detect DDoS attacks. In section IV, we illustrate the experimental results of DEQSVC using Qiskit and directly run it on an IBM quantum computer. In section V, we discuss future directions. Finally, section VI presents the conclusion of our proposed work.

II. RELATED WORK

In this section, we investigate several quantum machine learning solutions against DDoS attacks, pointing out their pros and cons in terms of protecting communication systems.

A. QUANTUM MACHINE LEARNING SECURITY SOLUTIONS

This section discusses the most recent quantum machine learning approaches used to detect cybersecurity DDoS attacks on communication systems, as well as their pros and cons. This section also illustrates tools used to develop the surveyed quantum machine learning solutions, as well as the datasets that are utilized to conduct the proposed experiments for these solutions.

Soliman et al. developed a network intrusion detection algorithm called Quantum Vaccine Immune Clonal Algorithm (QVICA) [12]. The proposed algorithm uses bio-inspired quantum techniques and the Estimation of Distribution Algorithm (EDA) to detect DDoS attacks. The experimental results indicate that the proposed algorithm achieved 94% classification accuracy of DDoS attacks.

Dong et al. proposed the quantum beetle swarm optimization algorithm to detect malicious network intrusions [13]. The algorithm utilizes the least-square quantum regression to decompose the features matrix of intrusion attacks, which results in minimizing the computational complexity of classical extreme machine learning. The results showed that the proposed model enhances the convergence rate for detecting malicious security attacks.

Gong et al. developed an efficient DDoS attacks detection approach based on Quantum Genetic Optimization and the BP neural network (DQGA-BP). The proposed approach enhanced the rotation angle of the quantum revolving gates of the QGO algorithm; thus, it has better convergence and searching abilities [14]. The experimental results show that this algorithm has 0.51% average detection and s 0.37% false alarm rates.

Bang et al. developed a quantum machine learning approach based on a single measurement and a first-in-first-out (FIFO) memory technique to detect cybersecurity attacks on communication protocols, such as TCP/IP and LDAP. This approach allows a receiver to learn a unitary transformation related to the quantum task identified by a sender, which enables the detection of a malicious third-party attacker [16].

Gouveia et al. developed an unsupervised Quantum-Assisted Support Vector Machine (QASVM) approach to detect cybersecurity attacks against communication protocols (e.g., LDAP) [17]. The proposed QASVM approach enhances the security of NIDs in terms of quantum-assisted computing. The approach shows promising results; it achieved 93% accuracy when it was applied to the NSL-KDD dataset and 75% accuracy when it was applied to the NB15 dataset.

Yamany et al. developed an Optimized Quantum-based Federated Learning (OQFL) framework to counter adversarial attacks in intelligent transportation systems [18]. The proposed security defense approach utilizes a Quantum Particle Swarm Optimization algorithm (QPSO) to update the learning rate hyperparameters and a cyber defense approach to detect adversarial attacks. The results indicate that the OQFL technique is robust against cybersecurity attacks.

Chen et al. proposed the Quantum-inspired Ant Lion Optimized Hybrid K-means (QALO-K) for malicious intrusion detection in communication systems [19]. The hybrid

TABLE 1. Quantum machine learning security detection approaches of DDoS attacks.

Authors	Methods	Descriptions	Dataset	Tools	Metrics	Limitations of the solutions
Soliman et al. [12]	QVICA	The bio-inspired quantum classification algorithm	The KDD-Cup 99 data dataset	The KDD-Cup 99 data mining tools	Accuracy	Limited parameters of DDoS attack. High complexity
Dong et al. [13]	Quantum beetle swarm optimization	Uses the least-square quantum regression to decompose the features matrix of intrusion attacks	The KDD-Cup 99 data dataset	N/A	Accuracy and convergence	Suffer from qubit decoherence
Gong et al. [15]	DQGA-BP	Enhances the rotation angle of the quantum revolving gates	The KDD-Cup 99 data dataset	Matlab	Average detection and false alarm	It has complicated and costly quantum chromosome mutation computations
Bang et al. [16]	Quantum machine learning based on single measurement FIFO	Allows a receiver to learn a unitary transformation	N/A	Monte Carlo simulator	Accuracy and learning time	Works only for single qubits communication protocols
Gouveia et al. [18]	QSVM	Enhances the security of the NIDSs in terms of quantum-assisted computing	NSL-KDD and NB15 datasets	IBM QX simulator	Accuracy	Weak encoding algorithm. No statistical similarity between the training and the testing models
Yamane et al. [18]	OQFL	Uses a QPSO technique and a cyber defense strategy to detect adversarial attacks	MINST and Fashion-MINST datasets	Python	Accuracy	Cannot handle large-scale heterogeneous datasets that have complex hyperparameters
Chen et al. [19]	QALO-K	Takes advantage of quantum computing, swarms intelligence methods, and k-means clustering	UCI and KDD Cup 99 datasets	MATLAB	Accuracy, detection rate, false-positive rate, and F-measure	It cannot identify the types of intrusions (misclassifying normal network accesses)
Payares et al. [11]	QEM	Uses quantum superposition to store sets of parameters to generate an ensemble model of quantum classifiers	CICDDoS2019 dataset	Python. Qiskit	Accuracy. Precision. Recall. F-measure, and error rate	Complex computations. Requires a lot of memory. The circuits are not accurate in generating the detection results
Islam et al. [31]	HCQNN	Extract high-dimensional features using a neural network and process them using qubits	CICDDoS2019 dataset	Python. Qiskit	Accuracy	Complex computations.

approach employs quantum computing techniques and swarm intelligence methods to enhance the k-means algorithm to converge towards the global optimal direction rather than falling into a local direction. This results in increasing the accuracy of detecting malicious network intrusions.

Payares et al. developed a Quantum Ensemble Model (QEM) for detecting DDoS attacks [11] on LDAP and SSDP. The idea behind the proposed model is to use quantum superposition to store sets of parameters to generate an ensemble model of quantum classifiers measured in a parallel environment. The proposed ensemble approach utilizes the angle encoding method to generate four qubits to detect DDoS attacks. The detection process works by measuring a Pauli Z gate operator on every two qubits, then the results are passed into a SoftMax function to produce two-dimensional probability vectors that classify data into two classes: DDoS and benign. The results show that the proposed approach achieves high detection accuracy, recall, precision, and F-score for DDoS attacks.

Islam et al. developed a hybrid classical-quantum neural network to detect cybersecurity attacks in the cloud-supported in-vehicle environment [15]. The idea of the proposed approach is to extract high-dimensional features using the neural network and process them using qubits to detect amplitude-shifted cybersecurity attacks. The results show that the proposed hybrid approach achieved 94% detection accuracy.

As seen in Table 1, many of the existing quantum machine learning detection techniques for DDoS attacks have limitations that limit their detection performance. The security detection approaches in [11], [12], [14], and [15] have complex training computations, which result in inefficient detection accuracy. The quantum models in [13], and [19] suffer from qubit decoherence, which results in misclassifying normal network behaviors as DDoS attacks. The approach in [16] is limited to one qubit and cannot use more than one qubit to detect DDoS attacks, which results in insufficient detection accuracy. The approaches in [17] have weak encoding algorithms, and there is no statistical similarity between the training and the testing models, which indicates that the results are not accurate. Furthermore, the approaches in [11], and [18] cannot handle large-scale datasets. Additionally, most of the methods provided use PCA, while our method uses the fast-independent component analysis (ICA) to find independent variables in our feature set and reduce its dimensions.

III. DEQSVC FOR DETECTING DDOS ATTACKS

In this section, we will illustrate the workflow of the DEQSVC algorithm and how quantum computing is used to enhance the performance of the classical SVM in the detection of DDoS attacks.

As mentioned in section II, the existing quantum, classical, and hybrid approaches suffer from gaps that limit their performances, such as qubit decoherence, low detection performance, and inaccurately calculating complex tasks that take a long time to solve (e.g., the kernel trick) [3], [4], [5], [6], [7]. However, to overcome these limitations and provide an accurate detection model for cybersecurity attacks, DEQSVC as seen in Figure 1 and Algorithm 1 consists of five phases: data selection, preprocessing, encoding, DEQSVC training, and detection.

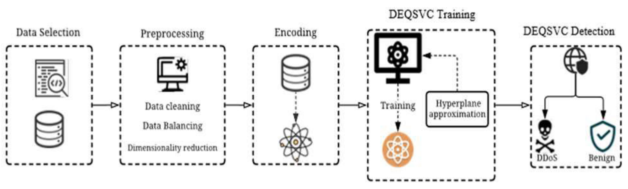

FIGURE 1. DEQSVC detection approach.

A. THE DDOS ATTACKS DATASET

To train our models, we will use the DDoS Evaluation Dataset (CIC-DDoS2019), which contains Benign and the most up-to-date common DDoS attacks [11], [21]. It contains the results of the network traffic analysis (e.g., attack time) using CICFlowMeter-V3 with labeled flows based on the time stamp, source and destination IP addresses, source and destination ports, protocols, and attack vectors. Moreover, it includes the event logs (windows and Ubuntu event logs) per device.

In our proposed experiment, we will use all the 80 features of the dataset and a sample dataset that has 5000 data points of type DDoS_LDAP protocol, which means that we are using the CSV file that contains DDoS attacks on the LDAP protocol as well as benign traffic. Also, we will perform the necessary preprocessing to have only two class labels: Benign, which means that the analyzed data does not represent any security threats and LDAP-type DDoS attacks.

B. DATA PREPROCESSING

The preprocessing step has three parts: data cleaning, data balancing (scaling and normalization), and dimensionality reduction. This step is very crucial in terms of detection model preparation because we are dealing with real-world network traffic data that is not correctly balanced, scaled, and distributed. Therefore, we need to handle this data with the necessary preprocessing to improve the performance of DEQSVC and generate accurate results.

Algorithm 1 The Pseudocode of the DEQSVC

```
1: Import all the necessary quantum libraries from Qiskit
2: Import all the necessary machine learning libraries

—-Preprocessing—-
3: Manual data cleaning

4: Class labels = ['BENIGN', 'DDoS_LDAP']
5: Data: Read dataset (DDoS_LDAP.csv)
6: Standardize the dataset:
StandardScaler.fit_transform(Data)

7: Split the preprocessed dataset into training and testing datasets:
Initialize the Seed
Initialize the training dataset size
Initialize the testing dataset size
8: Reduce the Dimensions of the dataset:
FastICA(the resulting required features, random states)
FastICA.transforms(Training dataset)
FastICA.transforms(Testing dataset)

9: Normalize the dataset:
MinMaxScaler.transform(Training dataset)
MinMaxScaler.transform(Testing dataset)

—-Encoding:—-
10: FeatureMap = ZZFeatureMap(transforms the training and the testing
datasets into quantum states) #two qubits

—-Training and testing processes—-

#Training and testing on a quantum simulator on the local classical computer
11: backend = ('statevector simulator')
12: QuantumInstance = QuantumInstance(backend, shots, seed,
transpiler=seed)
13: Kernel = QuantumKernel(featureMap, quantumInstance)
14: DEQSVC = PegasosQSVC(kernel, training parameters)
15: DEQSVC.fit(training data)
16: DEQSVC.score(test data)

#Training and testing on the IBM quantum computer
11: IBMQ.save_account(access token)
12: Provider = IBMQ.get_provider('ibm-q')
13: IBMQC = provider.get_backend(IBM quantum computer)
14: QuantumInstance = QuantumInstance(IBMQC, shots, seed,
transpiler=seed)
15: Kernel = QuantumKernel(featureMap, quantumInstance)
16: DEQSVC = PegasosQSVC(kernel, training parameters)
17: DEQSVC.fit(training data)
18: DEQSVC.score(test data)

—- Output—-
Print prediction classes: DDoS_LDAP as 1, BENIGN as 0
Calculate and print Accuracy
Calculate and print recall
Calculate and print precision
Calculate and print F-score
Calculate and print the error rate
```

The preprocessing step has three parts: data cleaning, data balancing (scaling and normalization), and dimensionality reduction. This step is very crucial in terms of detection model preparation because we are dealing with real-world network traffic data that is not correctly balanced, scaled, and distributed. Therefore, we need to handle this data with the necessary preprocessing to improve the performance of DEQSVC and generate accurate results.

1) DATA CLEANING

The dataset has real-world data points that represent network traffic, and due to this, the dataset contains inaccurate information that results in poor detection accuracy. To solve this problem, we cleaned up the dataset as follows:

1) Eliminate all the data points that have null values or white spaces.
2) Remove data points with a missing percentage more than a specified threshold in the original network traffic analysis using the CICFlowMeter-V3.
3) Remove features with a single unique value that does not belong to a formal range distribution.
4) Eliminate collinear data points with a correlation more than a specified correlation coefficient.

2) DATA BALANCING

The dataset contains real-world network traffic data that is imbalanced and has an informal distribution range. However, to increase the detection accuracy of the DEQSVC model, the dataset must be standardized and normalized. Thus, we used the StandardScaler and MinMaxScaler methods to balance our dataset and normalize it to a formal distribution range [24].

StandardScaler is the process of standardizing features by eliminating the mean and scaling to a variance unit. The equation for standardizing a dataset sample x is as follows. $z = (x - v)/s$, where v is the Mean and s is the standard deviation of the training dataset [24]. However, we scale each feature independently by calculating the relevant statistics on the data points in the training set, and then the Mean and standard deviation are stored to be utilized on later data using the transform method. After we standardize the dataset, we reduce its dimensions and transform each of them into a range of $(-1,1)$ using the MinMaxScaler.

In the DEQSVC model, we standardized and normalized the dataset because most of the selected dataset features have a center around 0 and have variance in the same order [23]. However, if a feature has a variance that is orders of magnitude larger than others, it will control the kernel function and make the training model unable to learn from other features accurately. Thus, by performing the standardization and normalization processes, we increase the detection accuracy of our proposed model.

3) DIMENSIONALITY REDUCTION

As described above, we intend to develop a QSVC approach that has the highest possible detection accuracy of cybersecurity DDoS attacks. However, the high dimensionality of the dataset may cause poor detection accuracy as well as poor training times. To overcome this issue, we used the fast ICA Dimensionality reduction technique to reduce the dimensions of the training dataset to two dimensions. The fast ICA is an enhanced version of the ICA and is a common dimensionality technique that is used to separate variables that have more than one outcome into independent sub-components [25].

The fast ICA method works by representing the sample dataset as a random vector $x = (x_1, x_2, \ldots, x_m)^T$ and the random states associated with the observed variables as random vector $s = (s_1, s_2, \ldots, s_n)^T$. The goal is to transform x using a linear static transformation into a vector of maximally sub-independent components estimated by some function $F(s_1, s_2, \ldots, s_i)^T$ of independence [25].

In our case, the dataset contains linearly independent data points. Therefore, we used the following fast ICA equation to generate the reduced features:

$$x_i = d_{i,1}s_1 + \ldots + d_{i,k}s_k + \ldots + d_{i,n}s_n. \quad (1)$$

Here, $d_{i,k}$ is the mixing weight of the data point d_i and $x = (x_1, x_2, \ldots, x_m)^T$ is the sum of the independent sub-components of the random states s_k, $k = 1, 2, 3, \ldots, n$. The results of the fast ICA reduce the dataset to two features, as illustrated in Figure 2.

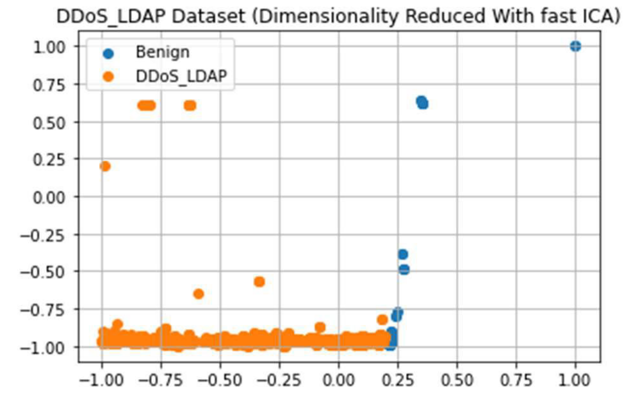

FIGURE 2. The obtained dimensions using the fastICA.

As seen in Figure 2, the fast ICA process generates two features, and we have only two labels: Benign and DDoS_LDAP. We first applied the PCA, but it caused DEQSVC to generate poor detection accuracy [25]. While the ICA looks for independent variables, the PCA technique only searches for uncorrelated relationships. Uncorrelated variables have a linear relationship; however, our dataset contains linearly independent data points. Therefore, the fast ICA allows DEQSVC to generate high accuracy.

C. DATA ENCODING

There are various encoding techniques used to represent classical datasets as quantum states. The idea is to transform a classical datapoint x and encode it into a quantum circuit as gate parameters to create a quantum state $|\psi\rangle$ [26], [27]. In DEQSVC, we use the ZZFeatureMap to transform our data. Havcilek et al. proposed a feature map family to encode n-dimensional data into n qubits [28]. The map is generated first by the unitary:

$$|\psi\rangle = U_{\phi}(\vec{x}) = U_{\phi(\vec{x})} H^{\otimes n} U_{\phi(\vec{x})} H^{\otimes n} \tag{2}$$

where the unitary gate $U_{\phi(\vec{x})}$ is defined as:

$$U_{\phi(\vec{x})} = \exp\left(i \sum_{S \subseteq [n]} \phi_s(\vec{x}) \prod_{i \in S} P_i \right) \tag{3}$$

With Pauli operators $P_n \in \{I, X, Y, Z\}^{\otimes n}$. Some examples of first-order Pauli operators in matrix form are:

$$I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$$
$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \tag{4}$$

The index S describes the connectivity between qubits, $S \in \left\{ \binom{n}{k}\, combinations, k = 1, \ldots, n \right\}$. In Qiskit the data mapping function ϕ_s defaults to:

$$\phi_s(\vec{x}) = \left\{ \begin{array}{ll} x_0 & if\ k = 1 \\ \prod_{j \in S} (\pi - x_j) & otherwise \end{array} \right\} \tag{5}$$

Originally, we used this feature map named the PauliFeatureMap in Qiskit. This map is the more general form of the feature map where Pauli operators are used to describe the connections between features and qubits. The 2-Qubit Pauli feature map takes the circuit form:

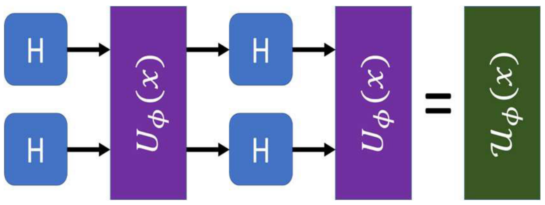

FIGURE 3. Pauli feature map family circuit.

Figure 3 shows the Pauli Feature Map Family Circuit. The ZFeatureMap is a first-order Pauli Z-Evolution circuit such that there is no interaction between features in the encoded data leading to a circuit with no entanglement gates. As such the ZFeatureMap is the case for $k = 1$ where the Pauli strings are fixed as ['Z']. This map can be computed efficiently classically and thus provides no quantum advantage [26], [28] If it is allowed to use general real values $\phi_s(x)$, $d = 2$ is sufficient to encode P-Hard problems in the output probability. This assumption is possible in the scenario where the output probability can only be approximated using multiplicative error [29]. For a single repetition of the ZFeatureMap, we may approximate the kernel with an additive error that scales with the number of samples on the uniform distribution over n classical bits. As such, for this case we may compute the kernel classically, proving no quantum advantage [27], [28].

Furthermore, the ZZFeatureMap used in DEQSVC is a second-order Pauli Z-Evolution circuit. As opposed to the first-order counterpart, this is the case for k = 2 with the Pauli strings ['Z','ZZ'] which can be described using the matrix form:

$$ZI = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} \tag{6}$$

$$IZ = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} \tag{7}$$

$$ZZ = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{8}$$

where $Z_i \in \{IZ, ZI, ZZ\}$, and $U_{\phi(\vec{x})}$ is represented by the following formula:

$U_{\phi(\vec{x})} = \exp(i \sum_{S \subseteq [n]} \phi_s(\vec{x}) \prod_{i \in S} Z_i)$ as seen in [28].

The ZZFeatureMap data mapping function ϕ_S for 2 qubits:

$$\phi_S = (\pi - x_i)(\pi - x_j) \tag{9}$$

The ZZFeatureMap contains two local interactions and must have two or more qubits to have a potential for quantum advantage. The kernel that is to be estimated must also be classically hard to compute to gain any quantum advantage. Recall that our data has been reduced to two dimensions, so we consider the two-qubit case.

Using the following one and two qubit gates we can describe our unitary:

$$U_{\phi\{i,j\}(x)} = \exp\left(i\phi_{\{i,j\}}(x) Z_i Z_j\right) \tag{10}$$
$$U_{\phi\{i\}(x)} = \exp\left(i\phi_{\{i\}}(x) Z_i\right) \tag{11}$$
$$U_{\phi(x)} = \exp\left(i\phi_i(x) Z_i + i\phi_j(x) Z_j + i\phi_{i,j}(x) Z_i Z_j\right) \tag{12}$$

According to $U_{\phi(\vec{x})}$ and equation (5), $Z_i = ZI$ and $Z_j = IZ$ [27].

Although this encoding method is effective for our purpose, it is clear to see how adding more qubits would exponentially increase the cost of quantum operations. More work must be done if this feature map is to be applied efficiently for large values of n, the number of qubits.

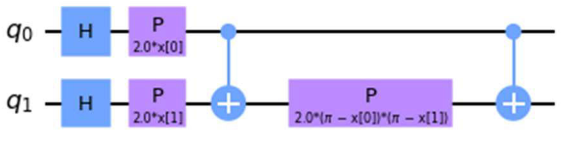

FIGURE 4. One repetition of the ZZFeatureMap.

As seen in Figures 4 and 5, the execution of the encoding technique will transform the dataset x into an n-qubits dataset with the form of $|\psi\rangle = U_{\phi(x)} |\psi_i\rangle^{\otimes 2}$ [26], [27].

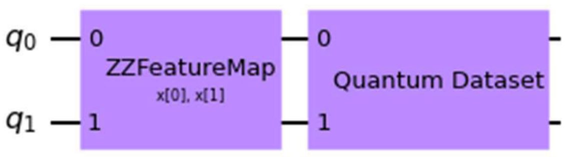

FIGURE 5. The encoding circuit – encodes a classical dataset $\vec{x}$ into a quantum dataset $|\psi\rangle$

Figure 4 illustrates the quantum data encoding circuit that is illustrated above to generate two qubits datasets: $|\psi\rangle = \{|q_0\rangle, |q_1\rangle\}$. The encoding circuit consists of two qubits q_0, q_1, ZZFeatureMap with two shifting gates $X^{\otimes 2}$, unitary operator $U_{\psi(x)}$ (two Hadamard gates $H^{\otimes 2}$, and one unitary gate $u_{\varphi(x)}$). The results of this circuit are the quantum dataset $|\psi\rangle$ that has all data points in the classical dataset mapped into two qubits.

Once the data is represented as quantum states, the kernel function estimates the inner products of the states, and the kernel is finally used to generate the hyperplane in the Hilbert space for classifying the dataset [27]. The kernel must be classically hard to estimate up to an additive polynomial small error for there to be potential for quantum advantage [30]. If the kernel can be estimated classically there will be no conceivable benefits from the quantum nature of our system.

D. TRADITIONAL QUANTUM KERNEL ESTIMATION

The kernel trick allows us to use a higher dimensional space to represent our features without having direct access to those features. Kernels do so by making use of the inner products of the feature space, preserving the necessary information for calculation.

Originally, we used Qiskit QSVM to solve the traditional dual quadratic SVM problem where we must first maximize

$$L_D(\alpha) = \sum_{i=1}^{t} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{t} y_i y_j \alpha_i \alpha_j K(x_i, x_j) \tag{13}$$

where $\sum_{i=1}^{t} a_i y_i = 0$ and $\alpha_i \geq 0$. Which can be solved using standard quadratic programming solvers [28]. The solution will be given by the vector $\alpha^* = (\alpha_1^*, \ldots, \alpha_t^*)$.

The density operator expansion is defined as:

$$\rho(x) = \sum_{i=1}^{4^n} a_i(x) \sigma_i = |\phi(x)\rangle\langle\phi(x)| \tag{14}$$

where the Pauli decomposition coefficients $a_i \in R$, and the density matrix $\sigma_i \in P_n \in \{I, X, Y, Z\}^{\otimes n}$, and recall our two-qubit ZZ Feature map with Pauli strings ['Z',' ZZ'] corresponding to the Pauli operators $[IZ, ZI, ZZ]$. According to equation (2), equation (4), and [27], we then represent the kernel as a form of the density operator $\rho(x)$:

$$K(x,z) = |\langle\phi(x) \mid \phi(z)\rangle|^2 = tr[\rho(x)\rho(z)] \tag{15}$$

$$K(x,z) = tr[\rho(x)\rho(z)] = tr\left[\sum_{i=1}^{4^n} a_i(x)\sigma_i a_i(z)\sigma_i\right] \tag{16}$$

As Pauli matrices are mutually orthogonal in terms of the trace inner product, we have the relation $tr[P_\alpha P_\beta] = \delta_{\alpha,\beta} 2^n$. Which we can expand for only the real coefficients [27].

$$K(x,z) = tr\left[\sigma_i^2\right] \sum_{i=1}^{4^n} a_i(x) a_i(z) = 2^n \sum_{i=1}^{4^n} a_i(x) a_i(z) \tag{17}$$

We may now optimize the Lagrange multipliers such that the equation above corresponds to the kernel we have estimated using DEQSVC. Any quadratic programming solver may be used to solve for a_i [27]. Now we may finally classify our test data using the sign of the traditional SVM equation:

$$\sum_{i=1}^{N} \alpha_i y_i K(x_i, x) \tag{18}$$

E. DEQSVC MODEL – ENHANCED QSVM APPROACH

In our experiment, we selected the Pegasos QSVC algorithm implemented in Qiskit. Pegasos is a stochastic sub-gradient descent algorithm used "for solving the optimization problem cast by Support Vector Machines" [30]. Rather than treating the problem as a constrained quadratic programming problem, Pegasos treats the SVM problem as an unconstrained loss minimization problem. While there are various methods of generating an SVM, Pegasos uses stochastic subgradient descent, that is, it uses noisy unbiased gradients in addition to a confined set of step sizes to solve the convex minimization problem.

In this section, we illustrate how we use the Pegasos QSVC algorithm to estimate the kernel for our training and test samples using Qiskit. After we encode the classical n-dimensional dataset x into the n-qubits dataset, the QSVC needs to estimate the kernel $K(x,z)$ for all pairs of training samples. Which will be used by our SVM to generate the hyperplane separating the two classes of datapoints, ['BENIGN',' DDOS_LDAP'] or [0,1]. This is done as follows:

Once classical data $\vec{x}$ is transformed into the quantum dataset $|\psi\rangle$ using the ZZ feature map:

$$|\psi(x)\rangle = U_{\phi(x)} |0\rangle^{\otimes n} \tag{19}$$

The kernel to be estimated $K(x,z)$ is defined as:

$$K(x,z) = |\langle\phi(x) \mid \phi(z)\rangle|^2 \tag{20}$$

Our QSVC treats the training as a loss minimization problem using the following representation [30]:

$$\frac{\lambda}{2}\|w\|^2 + \frac{1}{m} \sum_{(x,y)\in S} l(w; (x,y)) \tag{21}$$

Here, the weight vector w and $\langle w, x\rangle$ denotes the standard inner product between the two vectors. The training set S and $l(w; (x,y))$are given by.

$$S = \{(x_i, y_i)\}_{i=1}^{m}, x_i \in \mathbb{R}^n \text{and} y_i \in \{+1, -1\} \tag{22}$$

$$l(w; (x,y)) = \max\{0, 1 - y\langle w, x\rangle\} \tag{23}$$

By the representer theorem, we can solve the loss minimization problem by expressing the optimal solution as a linear combination of the training instances. Using only our kernel operator we can access the inner products of our data and express our loss minimization in terms of the implicit data mapping function $\phi(x)$:

$$\frac{\lambda}{2}\|w\|^2 + \frac{1}{m}\sum_{(x,y)\in S} l\left(w; (\phi(x), y)\right) \tag{24}$$

And according to equation (18), equation (19), and [30], $l(w; (\phi(x), y))$ is represented by the following formula:

$$l\left(w; (\phi(x), y)\right) = \max\{0, 1 - y\langle w, \phi(x)\rangle\} \tag{25}$$

Recall that the data mapping function can be estimated using kernel evaluations, also for simplicity, we continue to use the data mapping representation $\phi(x)$ when we refer to kernel estimation. Shalev-Shwartz proves for all t we can rewrite w_{t+1} as:

$$w_{t+1} = \frac{1}{\lambda t}\sum_{i=1}^{t} 1\left[y_{i_t}, \langle w_t, \phi(x_{i_t})\rangle < 1\right] y_{i_t}\phi(x_{i_t}) \tag{26}$$

where $\mathbb{1}$ takes the value 1 if the binary statement is true, 0 otherwise. Rather than solving a dual quadratic problem, as do most SVM classifiers, Pegasos minimizes the primal problem using the kernels.

Algorithm 2 Kernelized Pegasos Algorithm as Seen in [30, Figure. 3]

```
INPUT: $S, \lambda, T$
INITIALIZE: Set $\alpha_1 = 0$
  FOR $t = 1, 2 \ldots, T$
  Choose $i_t \in \{0, \ldots, |S|\}$ uniformly at random
  For all $j \neq i_t$, set $\alpha_{t+1}[j] = \alpha_t[j]$
  If $y_{i_t}\frac{1}{\lambda t}\sum \alpha_t[j] y(i_t) K(x_{i_t}, x_j) < 1$, then:
    set $\alpha_{t-1}[i_t] = \alpha_t[i_t] + 1$
  Else:
    Set $\alpha_{t+1}[i_t] = \alpha_t[i_t]$
OUTPUT: $\alpha_{T+1}$
```

As seen in algorithm II, For each t, $\alpha_{t+1} \in \mathbb{R}^m$ counts how many times j has been selected with non-zero loss [30].

$$\alpha_{t+1}[j] = \left|\left\{t' \leq t : i_{t'} = j \wedge y_j\langle w_{t'}, \phi(x_j)\rangle < 1\right\}\right| \tag{27}$$

We now represent our weight vector w_{t+1} in terms of α_{t+1}:

$$w_{t+1} = \left(\frac{1}{\lambda t}\right)\sum_{j=1}^{m}\alpha_{t+1}[j]\, y_j\phi(x_j) \tag{28}$$

By the representer theorem, we find that the optimal solution to our minimization problem is spanned by the training instances in the form of $w = \sum_{i=1}^{m}\alpha[i]\phi(x_i)$. According to equation (20), we may now rewrite the problem in terms of our kernel and α.

$$\min_{\alpha}(\lambda/2)\alpha[i]\,\alpha[j]\,K(x_i, x_j) + (1/m)\sum_{i=1}^{m} \times \max\left\{0, 1 - y_i\sum_{j=1}^{m}\alpha[j]\,K(x_i, x_j)\right\} \tag{29}$$

Pegasos does not ever access the inner products using the data mapping function but rather via the kernel evaluations. We then take gradients concerning α with stochastic gradient descent. The Pegasos QSVC is well suited for large datasets as the runtime does not depend directly on the size of the training set [30]. However, for our experiment, we will not benefit much from this quality as our training set is relatively small.

Now that we have an optimized kernel, we present the QSVC used to generate the hyperplane which binarily classifies our test data. Our SVC will then classify test data x depending on the sign of the following:

$$\sum_{i=1}^{N}\alpha_i y_i K(x_i, x)\ (30) \tag{30}$$

where the Lagrange multipliers are $\alpha_i \in R^n$, and pairs of training data are (x_i, y_i). The Lagrange multipliers are parameters optimized by the SVC to solve for the optimal hyperplane.

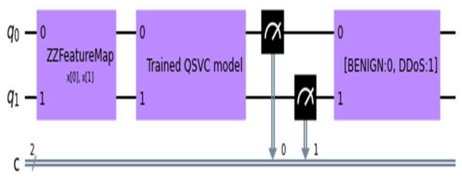

FIGURE 6. The trained DEQSVC model.

Figure 6 shows the DEQSVC circuit. Here, we apply the necessary measurements on the quantum dataset to generate the trained DEQSVC model that will form a hyperplane in the Hilbert space to detect DDoS cybersecurity attacks. Once the model is trained, using our measurement gates on each qubit, we can then collapse the quantum states to produce a classical result of either 0 or 1. Therefore, DEQSVC will detect and classify the data into two labels: DDoS_LDAP as 1, which represents the malicious DDoS attacks on the LDAP protocol, and BENIGN as 0, which means there is no malicious threat.

IV. EXPERIMENTS AND RESULTS

This section provides a clear view of the computational detection performance of DEQSVC compared to the benchmark approaches, the QEM, QSVM, Quantum Neural Network (QNN), QASVM, and Hybrid Classical Quantum Neural Network (HCQNN) [11], [15], [17], [35]. Moreover, the results show the behavior of quantum models against malicious attacks and the potential for quantum advantage in cybersecurity.

TABLE 2. The setup of the experimental environment on the dell i-5 intel core computer.

Elements	Value	Description
Package used	Numpy, scipy, matplotlib, mpl_toolkits, sklearn, pandas, and Qiskit (aqua, circuit, and tools)	All the classical and quantum packages that are used to process the QSVM and the benchmark approaches
Training dataset size	4500	The size of the training data
Testing dataset size	500	The size of the testing data
Features	80	Number of the features in the dataset
Seed	10598	The random seed used in the quantum instance that runs the quantum simulator
Shots	5000	The number of times the experiment is run
Qubits	2	Number of qubits used in the QSVM and the benchmark models
Feature Map repetitions	4	Number of repetitions of the ZZFeatureMap encoding method
Quantum simulator	statevector_simulator	Local qiskit experiments simulation

A. QSVM EXPERIMENTATION USING LOCAL QISKIT SIMULATION

We used Qiskit through Python Jupyter on an I-5 Intel core, 6GB RAM computer to conduct our detection experiments of DDoS attacks and evaluate the performance of the traditional QSVM approach against the benchmark models.

In Tables 2 and 3, we depict all the crucial components of the setup of our experimental environment. Other variables concerning the benchmark approaches are set based on the experiments in [11], [15], and [17]. For example, the epoch number of the QNN in [15] is 30, so in our experiment, we set this value to 30 for consistency. We selected the state vector quantum simulator since the data encoding technique and the kernel function of the QSVM model will represent the quantum states as a state vector. Therefore, this simulator is the best option for local testing of our experiment.

B. DEQSVC EXPERIMENTATION USING LOCAL QISKIT SIMULATION

Using Qiskit on the same Dell machine, we implement DEQSVC to estimate our kernel and hyperplane. Unlike the QSVM approach which solves the loss minimization problem in the form of a dual quadratic, DEQSVC solves the loss minimization using sub-primal stochastic gradient descent.

TABLE 3. The setup of the experimental environment for DEQSVC.

Elements	Value	Description
Package used	Numpy, scipy, matplotlib, sklearn, pandas, and Qiskit (circuit and tools)	All the classical and quantum packages that are used to process DEQSVC
Training dataset size	1700	The size of the training data
Testing dataset size	200	The size of the testing data
Features	80	Number of the features in the dataset
Shots	4000	Number of times the experiment is run
Qubits	2	Number of qubits used in the DEQSVC and the benchmark models
Feature Map repetitions	4	Number of repetitions of the ZZFeatureMap encoding method
Quantum simulator	statevector_simulator	Local qiskit experiments simulation

TABLE 4. The results of the comparison in terms of the performance metrics.

Approach	Accuracy	Recall	Precision	F-score	Error rate	Wall-clock time (sec)
DEQSVC	99.5%	99%	99%	99%	0.5%	92
QSVM	97.14%	96.14 %	97.19%	96%	2.86%	190
QNN [15]	86%	89%	89.5%	86.3%	14%	5256
QEM [11]	65.35%	49.6%	43.18%	52%	34.6%	31
QASVM [17]	76.75%	82. 3%	86.15%	77.2%	23.2%	91
HCQNN [35]	97.52%	97.62 %	98.71$	97.85 %	2.48 %	7560

In our first experiment, we applied the PCA, but it decreased the performance of DEQSVC by generating 72% detection accuracy. The PCA technique searches solely for uncorrelated variables while the fast ICA looks for independent variables. Uncorrelated variables have a linear relationship; however, our dataset contains linearly independent data points. Therefore, the fast ICA increases the detection accuracy of the DEQSVC by more optimally minimizing the features of our dataset to two features [24].

As seen in Table 4, DEQSVC generated 99.49% detection accuracy, 99% recall, 99% precision, and 99% F-score. This means that DEQSVC outperformed the benchmark models [11], [15], [17], [35] in generating higher performance results. Note that the QEM in [11] generated the lowest detection performance results compared with the other detection models.

As seen in Fig. 7 and Table 4 the DEQSVC model has the lowest detection error rate among all the benchmark

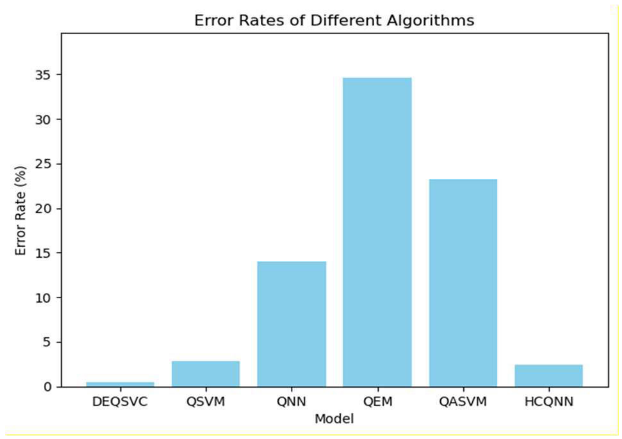

FIGURE 7. The error rate of the DEQSVC compared to the benchmark models.

approaches at 0.5%. The QSVM generated a 3.31% error rate, which means that it wrongly classified 3.31% of the data. Here, the QEM has the highest error rate of 34.65.

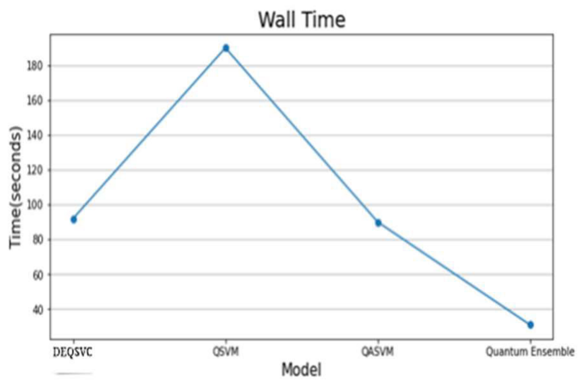

FIGURE 8. The wall time of the DEQSVC compared to the benchmark models.

Fig. 8 and Table 6 show the wall time (the total running time of each model) in seconds. Note that the QNN and HCQNN wall time (5256 and 7560 seconds) were omitted for visual clarity. The DEQSVC wall time is comparable to that of the QASVM while. The reason for the faster performance of DEQSVC lies in the data encoding and kernel estimation approaches III. This preserves our data in the that are explained in section form of inner products, which speeds up the detection process of DDoS attacks while also improving accuracy. DEQSVC also performs better than the benchmark models by making use of the fast ICA to reduce the dimensionality, while also preserving the independent variables in our data. The HCQNN had the longest run time among all the models with 7560 seconds of wall time and only 97% accuracy, proving far inferior to our model.

C. DEQSVC EXPERIMENTATION USING IBM QUANTUM COMPUTER

In this section, we describe our experimental results of DEQSVC on an IBM quantum computer. The purpose of conducting this experiment is to demonstrate that DEQSVC is accurate even with the properties of noisy intermediate scale quantum (NISQ) devices.

TABLE 5. The setup of the experimental environment on the IBM quantum computer.

Elements	Value	Description
Packaged used	Numpy, scipy, matplotlib, sklearn, pandas, and Qiskit (circuit, tools, IBMQ)	All the classical and quantum packages that are used to process the QSVM approach
Training dataset size	1700	Described in Table III
Testing dataset size	200	Described in Table III
Qubits	2	Described in Table III
Access API token	Code in Hexadecimal	An API access token that is used to access IBM quantum devices to run quantum codes
Feature Map repetitions	4	Number of repetitions of the ZZFeatureMap encoding method
IBM quantum devices service provider	ibm-q	The provider of the IBM quantum service that allows the execution of the QSVM on the backend that runs the quantum computer
Backend	ibm_oslo	7-qubit IBM quantum computer located in Oslo, Norway

The first four elements of Table 5 are fixed in both the classical and quantum computer setups. We were successfully able to execute DEQSVC by creating an account on the IBM quantum website and then using Jupyter to integrate the access token of our account into our DEQSVC code. Then, we identified the provider of the IBM quantum service (ibm-q) that provides access to the backend cloud service that runs the IBM Quantum computer. We then chose a quantum computer for which we chose IBM's 7-qubit quantum computer, ibm_oslo. Finally, we executed DEQSVC to run on the quantum computer. The current computational resources provided by IBM suffer from various API issues as well as long queue times making it extremely difficult to calculate the true wall-clock time. The results of the execution are illustrated in Table 6.

IBM uses a fair share queuing system to ensure that a process sent to IBM Quantum will not dominate the resources. If a job sent to IBM is deemed too large to run in a single job, IBM will split the job into multiple jobs, which are queued after each job is finished, increasing the time spent in the queue. Another limitation lies in the job/circuit submission, if a job error is encountered due to a failure on IBM's side, the job will not be resent to the queue for resubmission of the circuit(s). This can hinder accuracy as was encountered various times in the testing of DEQSVC. Certain IBM Quantum machines tend to encounter these sorts of errors more frequently than others in our experimentation. IBM Oslo was the most reliable machine in the list of available providers though it suffered from large queue times as many other jobs had been submitted to the machine.

TABLE 6. Results of DEQSVC on the IBM quantum computer.

Metrics	Generated values
Accuracy	92%
Recall	92%
Precision	92%
F-score	92%
Error rate	8%
Metrics	Generated values

Table 6 illustrates that the results obtained from running DEQSVC on the IBM Quantum device are slightly less accurate than the model on the state vector simulator. This can be attributed to the noise and faults from the decoherence of qubits. The current error correction schemes used by these quantum systems must be improved to further extend the use of quantum computers.

V. FUTURE WORK RECOMMENDATIONS

In this paper, we developed DEQSVC to detect DDoS attacks against the networking architecture of cybersecurity. The QSVC approach achieved a high detection accuracy of 99.49%, with an error rate of 0.51% using a simulator. DEQSVC on IBM's quantum computer only generated an accuracy of 92% with an error rate of 8%. Therefore, future work should be conducted to improve the hyperparameter tuning of DEQSVC and adopt error-correcting mechanisms, which will enhance the data processing of the QSVC approach and generate higher detection accuracy.

We intend to extend the application DEQSVC by applying it to different cybersecurity attacks datasets to see the resultant behaviors. Furthermore, we are also interested in applying the proposed model to medical issues, such as cancer datasets, and seeing the resultant behaviors.

To provide a more reliable quantum system IBM must improve upon its backend service by limiting the number of errors certain machines are subject to. They must also ensure that failed jobs due to IBM's errors are retried before they are discarded. More work must also be done in the development and implementation of error correction to provide more accurate results for all sorts of quantum algorithms.

VI. CONCLUSION

Quantum machine learning is a significant technology that possibly enhances the computational power of a system and increases its performance in terms of data processing. In this work, we proposed the DEQSVC that integrates effective dimensionality reduction techniques, a robust feature map encoding method, and an efficient QSVC to detect DDoS attacks.

The implementation of DEQSVC has three main phases: first, the dataset is preprocessed, then encoded into a quantum dataset (quantum states), and then the model is trained, in our case to detect DDoS attacks. In the preprocessing phase, we first manually clean the dataset, standardize it using the StandardScaler method, and then reduce its dimensionality using the FastICA technique. Finally, the generated dataset is normalized using the MinMaxScaler. In the encoding process, the ZZFeatureMap encoding technique is used, to transform the classical dataset into a quantum dataset. Finally, the DEQSVC is trained by estimating the kernel matrix that is used to generate the hyperplane in the Hilbert space. The hyperplane is then used to classify our data as benign network traffic or as a DDoS attack.

To verify the performance of the DEQSVC approach, we compared it with a traditional QSVM and benchmark algorithms using a widely used DDoS attack dataset named CIC-DDoS2019. The results showed that DEQSVC outperformed the benchmark algorithms by generating 99.49% accuracy, 99% recall, 99% precision, 99% F-score, and a 0.51% error rate. Furthermore, we demonstrated the accuracy of the QSVC by running it on the IBM quantum computer. DEQSVC surpasses the benchmark algorithms by use of its feature encoding and dimensionality reduction techniques as well as the stochastic gradient descent training method used.

It is crucial to detect and prevent DDoS attacks using efficient security solutions because they can harm and shut down an entire networking platform. Therefore, this proposed work demonstrates that quantum machine learning can enhance the cybersecurity of NIDSs and make them more robust against DDoS attacks.

ACKNOWLEDGMENT

The authors would like to acknowledge Danny DeJesus for his help with coding and validation.

REFERENCES

[1] M. Husák, J. Komárková, E. Bou-Harb, and P. Celeda, "Survey of attack projection, prediction, and forecasting in cyber security," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 1, pp. 640–660, 1st Quart., 2019, doi: 10.1109/COMST.2018.2871866.

[2] A. Saritha, B. R. Reddy, and A. S. Babu, "QEMDD: Quantum inspired ensemble model to detect and mitigate DDoS attacks at various layers of SDN architecture," *Wireless Pers. Commun.*, vol. 127, no. 3, pp. 2365–2390, Dec. 2022, doi: 10.1007/s11277-021-08805-5.

[3] K. Sharma and S. Bhatt, "Jamming attack—A survey," *Int. J. Recent Res. Asp.*, vol. 5, no. 1, pp. 74–80, 2018.

[4] D. Aggarwal, G. Brennen, T. Lee, M. Santha, and M. Tomamichel, "Quantum attacks on Bitcoin, and how to protect against them," *Ledger*, vol. 3, pp. 1–21, Oct. 2018, doi: 10.5195/ledger.2018.127.

[5] B. Kelley, J. J. Prevost, P. Rad, and A. Fatima, "Securing cloud containers using quantum networking channels," in *Proc. IEEE Int. Conf. Smart Cloud (SmartCloud)*, Nov. 2016, pp. 103–111, doi: 10.1109/SmartCloud.2016.58.

[6] J. Xie, F. R. Yu, T. Huang, R. Xie, J. Liu, C. Wang, and Y. Liu, "A survey of machine learning techniques applied to software defined networking (SDN): Research issues and challenges," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 1, pp. 393–430, 1st Quart., 2019, doi: 10.1109/COMST.2018.2866942.

[7] T. M. Khan and A. Robles-Kelly, "Machine learning: Quantum vs classical," *IEEE Access*, vol. 8, pp. 219275–219294, 2020, doi: 10.1109/ACCESS.2020.3041719.

[8] F. Jazaeri, A. Beckers, A. Tajalli, and J.-M. Sallese, "A review on quantum computing: From qubits to front-end electronics and cryogenic MOSFET physics," in *Proc. 26th Int. Conf. 'Mixed Design Integr. Circuits Syst.*, Jun. 2019, pp. 15–25, doi: 10.23919/MIXDES.2019.8787164.

[9] A. Mubarakali, K. Srinivasan, R. Mukhalid, S. C. B. Jaganathan, and N. Marina, "Security challenges in Internet of Things: Distributed denial of service attack detection using support vector machine-based expert systems," *Comput. Intell.*, vol. 36, no. 4, pp. 1580–1592, Nov. 2020, doi: 10.1111/coin.12293.

[10] Y. Mao, W. Huang, H. Zhong, Y. Wang, H. Qin, Y. Guo, and D. Huang, "Detecting quantum attacks: A machine learning based defense strategy for practical continuous-variable quantum key distribution," *New J. Phys.*, vol. 22, no. 8, Aug. 2020, Art. no. 083073, doi: 10.1088/1367-2630/aba8d4.

[11] E. D. Payares and J. C. Martinez-Santos, "Quantum machine learning for intrusion detection of distributed denial of service attacks: A comparative overview," *Proc. SPIE*, vol. 11699, Mar. 2021, Art. no. 116990B, doi: 10.1117/12.2593297.

[12] O. S. Soliman and A. Rassem, "A network intrusions detection system based on a quantum bio inspired algorithm," *Int. J. Eng. Trends Technol.*, vol. 10, no. 8, pp. 370–379, Apr. 2014. [Online]. Available: http://www.ijettjournal.org

[13] Y. Dong, W. Hu, J. Zhang, M. Chen, W. Liao, and Z. Chen, "Quantum beetle swarm algorithm optimized extreme learning machine for intrusion detection," *Quantum Inf. Process.*, vol. 21, no. 1, p. 9, Jan. 2022, doi: 10.1007/s11128-021-03311-w.

[14] C. Gong, T. Shi, M. Mu, L. Zhao, A. Gani, and H. Qi, "An improved quantum genetic algorithms and application for DDoS attack detection," in *Proc. IEEE Int. Conf. Parallel Distrib. Process. Appl., Big Data Cloud Comput., Sustain. Comput. Commun., Social Comput. Netw.*, Dec. 2019, pp. 427–434, doi: 10.1109/ISPA-BDCloud-SustainCom-SocialCom48970.2019.00068.

[15] M. Islam, M. Chowdhury, Z. Khan, and S. M. Khan, "Hybrid quantum-classical neural network for cloud-supported in-vehicle cyberattack detection," 2021, *arXiv:2110.07467*.

[16] J. Bang, S. W. Lee, and H. Jeong, "Protocol for secure quantum machine learning at a distant place," *Quantum Inf. Process.*, vol. 14, pp. 3933–3947, Jan. 2015, doi: 10.1007/s11128-015-1089-7.

[17] A. Gouveia and M. Correia, "Towards quantum-enhanced machine learning for network intrusion detection," in *Proc. IEEE 19th Int. Symp. Netw. Comput. Appl. (NCA)*, Nov. 2020, pp. 1–8, doi: 10.1109/NCA51143.2020.9306691.

[18] W. Yamany, N. Moustafa, and B. Turnbull, "OQFL: An optimized quantum-based federated learning framework for defending against adversarial attacks in intelligent transportation systems," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 1, pp. 893–903, Jan. 2023, doi: 10.1109/TITS.2021.3130906.

[19] J. Chen, X. Qi, L. Chen, F. Chen, and G. Cheng, "Quantum-inspired ant lion optimized hybrid k-means for cluster analysis and intrusion detection," *Knowl.-Based Syst.*, vol. 203, Sep. 2020, Art. no. 106167, doi: 10.1016/j.knosys.2020.106167.

[20] I. Sharafaldin, A. H. Lashkari, S. Hakak, and A. A. Ghorbani, "Developing realistic distributed denial of service (DDoS) attack dataset and taxonomy," in *Proc. Int. Carnahan Conf. Secur. Technol. (ICCST)*, Chennai, India, Oct. 2019, pp. 1–8.

[21] X. Wang, H. Schulzrinne, D. Kandlur, and D. Verma, "Measurement and analysis of LDAP performance," *IEEE/ACM Trans. Netw.*, vol. 16, no. 1, pp. 232–243, Feb. 2008, doi: 10.1109/TNET.2007.911335.

[22] K. Q. Wu and C. C. Liu, "Design and implementation of semantic resource management platform based on LDAP," in *Proc. 2nd Int. Conf. Civil, Mater. Environ. Sci.*, 2015, pp. 1–12, doi: 10.2991/cmes-15.2015.65.

[23] V. N. G. Raju, K. P. Lakshmi, V. M. Jain, A. Kalidindi, and V. Padma, "Study the influence of normalization/transformation process on the accuracy of supervised classification," in *Proc. 3rd Int. Conf. Smart Syst. Inventive Technol. (ICSSIT)*, Aug. 2020, pp. 729–735, doi: 10.1109/ICSSIT48917.2020.9214160.

[24] P. Spurek, J. Tabor, Ł. Struski, and M. Śmieja, "Fast independent component analysis algorithm with a simple closed-form solution," *Knowl.-Based Syst.*, vol. 161, pp. 26–34, Dec. 2018, doi: 10.1016/j.knosys.2018.07.027.

[25] M. M. Hossain, M. S. Ali, R. A. Swarna, M. M. Hasan, N. Habib, M. W. Rahman, M. M. Azad, and M. M. Rahman, "Analyzing the effect of feature mapping techniques along with the circuit depth in quantum supervised learning by utilizing quantum support vector machine," in *Proc. 24th Int. Conf. Comput. Inf. Technol. (ICCIT)*, Dec. 2021, pp. 1–5, doi: 10.1109/ICCIT54785.2021.9689853.

[26] Y. Suzuki, H. Yano, Q. Gao, S. Uno, T. Tanaka, M. Akiyama, and N. Yamamoto, "Analysis and synthesis of feature map for kernel-based quantum classifier," *Quantum Mach. Intell.*, vol. 2, pp. 1–9, Jun. 2019, doi: 10.1007/s42484-020-00020-y.

[27] V. Havlíček, A. D. Córcoles, K. Temme, A. W. Harrow, A. Kandala, J. M. Chow, and J. M. Gambetta, "Supervised learning with quantum-enhanced feature spaces," *Nature*, vol. 567, no. 7747, pp. 209–212, Mar. 2019, doi: 10.1038/s41586-019-0980-2.

[28] L. A. Goldberg and H. Guo, "The complexity of approximating complex-valued Ising and Tutte partition functions," 2014, *arXiv:1409.5627*.

[29] S. Shalev-Shwartz, Y. Singer, N. Srebro, and A. Cotter, "Pegasos: Primal estimated sub-gradient solver for SVM," *Math. Program.*, vol. 127, no. 1, pp. 3–30, Oct. 2010, doi: 10.1007/s10107-010-0420-4.

[30] D. Sierra-Sosa, M. Telahun, and A. Elmaghraby, "TensorFlow quantum: Impacts of quantum state preparation on quantum machine learning performance," *IEEE Access*, vol. 8, pp. 215246–215255, 2020, doi: 10.1109/ACCESS.2020.3040798.

[31] L. Gyongyosi and S. Imre, "A survey on quantum computing technology," *Comput. Sci. Rev.*, vol. 31, pp. 51–71, Feb. 2019, doi: 10.1016/j.cosrev.2018.11.002.

[32] A. Seifousadati, S. Ghasemshirazi, and M. Fathian, "A machine learning approach for DDoS detection on IoT devices," 2021, *arXiv:2110.14911*.

[33] S. Lloyd, M. Mohseni, and P. Rebentrost, "Quantum algorithms for supervised and unsupervised machine learning," 2013, *arXiv:1307.0411*.

[34] M. Schuld, M. Fingerhuth, and F. Petruccione, "Implementing a distance-based classifier with a quantum interference circuit," *EPL, Europhys. Lett.*, vol. 119, no. 6, p. 60002, Sep. 2017.

[35] A. Alomari and S. A. P. Kumar, "Hybrid classical-quantum neural network for improving space weather detection and early warning alerts," in *Proc. IEEE Cogn. Commun. Aerosp. Appl. Workshop (CCAAW)*, Cleveland, OH, USA, Jun. 2023, pp. 1–6, doi: 10.1109/CCAAW57883.2023.10219316.

AHMAD ALOMARI received the bachelor's and master's degrees in computer science from Yarmouk University, Irbid, Jordan, in 2014 and 2018, respectively. He is currently pursuing the Ph.D. degree in computer science and electrical engineering with the Department of Computer Science and Electrical Engineering, Cleveland State University. He is a Research Assistant in the area of quantum computing doing quantum machine learning research with the Intelligent Secure Cyber-Systems Analytics and Applications Research (ISCAR) Laboratory, Cleveland State University. He has publications in the areas of quantum machine learning, artificial intelligence, cybersecurity, and cloud computing. His research interests include quantum computing, artificial intelligence, machine learning, cybersecurity, and cloud computing.

SATHISH A. P. KUMAR (Senior Member, IEEE) received the Ph.D. degree in computer science and engineering from the University of Louisville, Louisville, KY, USA, in 2007. He is currently an Associate Professor of computer science with the Department of Electrical Engineering and Computer Science, Cleveland State University, Cleveland, OH, USA. He is also directing the ISCAR Laboratory, Cleveland State University. He has published more than 75 technical papers in international journals and conference proceedings. His current research interests include cybersecurity, machine learning, quantum computing, and secure distributed systems and their applications. He serves as an Associate Editor for IEEE Access, *PLOS One*, and *Machine Learning With Applications* (Elsevier), and as an Editorial Board Member for *Scientific Reports* (Nature).

• • •