IEEE Transactions on Quantum Engineering

Received 6 November 2023; revised 24 January 2024; accepted 25 January 2024; date of publication 29 January 2024; date of current version 1 March 2024.

Digital Object Identifier 10.1109/TQE.2024.3359574

Network Anomaly Detection Using Quantum Neural Networks on Noisy Quantum Computers

ALON KUKLIANSKY[1], MARKO ORESCANIN[1] (Member, IEEE), CHAD BOLLMANN[1] (Senior Member, IEEE), AND THEODORE HUFFMIRE[1]

Naval Postgraduate School, Monterey, CA 93943 USA

Corresponding author: Alon Kukliansky (e-mail: alon.kukliansky.is@nps.edu).

This work was supported by the Internal Funding of the Naval Postgraduate School (NPS) and in part by LP-CRADA under Agreement LP-CRADA-NPS-23-0119 from IonQ, Inc. to NPS.

ABSTRACT The escalating threat and impact of network-based attacks necessitate innovative intrusion detection systems. Machine learning has shown promise, with recent strides in quantum machine learning offering new avenues. However, the potential of quantum computing is tempered by challenges in current noisy intermediate-scale quantum era machines. In this article, we explore quantum neural networks (QNNs) for intrusion detection, optimizing their performance within current quantum computing limitations. Our approach includes efficient classical feature encoding, QNN classifier selection, and performance tuning leveraging current quantum computational power. This study culminates in an optimized multilayered QNN architecture for network intrusion detection. A small version of the proposed architecture was implemented on IonQ's Aria-1 quantum computer, achieving a notable 0.86 F1 score using the NF-UNSW-NB15 dataset. In addition, we introduce a novel metric, certainty factor, laying the foundation for future integration of uncertainty measures in quantum classification outputs. Moreover, this factor is used to predict the noise susceptibility of our quantum binary classification system.

INDEX TERMS Intrusion detection, network intrusion detection system (NIDS), quantum neural network (QNN).

I. INTRODUCTION

In recent years, the proliferation of network-based attacks and security breaches has become a growing concern for organizations and individuals alike. Traditional intrusion detection systems (IDSs) often struggle to keep pace with the ever-evolving threat landscape due to the increasing complexity and sophistication of modern attacks [1], [2], [3]. Consequently, there is a pressing need for innovative approaches that can enhance the accuracy and efficiency of intrusion detection. Machine learning (ML) is one such approach that has shown promise in enhancing the accuracy and efficiency of IDSs [4], [5], [6], [7], [8]. ML-based network intrusion detection systems (NIDSs) have proven to be an essential, scalable tool in protecting networks against cyberattacks.

With the emergence of quantum computational research, the realm of quantum machine learning (QML) has garnered attention. Noteworthy advancements in various QML modalities, such as parameterized quantum circuits [9] and variational learning [10], have paved the way for developing efficient quantum classifiers [11], [12], [13].

While quantum computing presents immense potential, it is not without its challenges, particularly in the context of noisy intermediate-scale quantum (NISQ) [14] era machines. These devices are characterized by inherent imperfections stemming from a variety of sources, such as environmental interference, control imprecision, and decoherence effects [15].

Recent studies [16], [17], [18], [19] have substantiated the efficacy of quantum neural network (QNN) and quantum support vector machines (QSVMs) in classifying network activities. These findings have demonstrated exceptionally high detection rates in noiseless simulation, showcasing the promise of quantum approaches in the field of network security and intrusion detection; however, none of them have shown good classification performance on a physical quantum computer.

This work also investigates classifying network activities with a QNN. We began with a survey of various established QNN classifiers, culminating in the selection of the classifier demonstrating the most promising classification performance. But rather than attempting to improve performance by mitigating noise, we pursued the creation of an ultralean circuit, designed to suffer the least possible interference. This strategy seeks to harness the power of quantum computing while working within its current limitations. This deliberate choice is central to our exploration, as it allows us to push the boundaries of quantum-enhanced intrusion detection, even within the constraints of today's quantum computing limitations.

Another contribution of our approach lies in the development of an efficient procedure to encode classical NetFlow information from the NF-UNSW-NB15 dataset [20]. This procedure minimizes the quantum resources required, ultimately leading to a one-qubit representation for each classical feature and a single rotation gate for its state preparation. We conclude by successfully testing our novel algorithms and achieving a significant improvement over the state of the art using IonQ's quantum computers.

The key contributions made in this research include the following.

1) Development of an ultralean QNN architecture, optimized for NISQ devices, achieving network activity classification performance on par with classical ML techniques
2) A novel encoding scheme to efficiently represent classical NetFlow data in qubit rotations
3) The introduction of a certainty factor that enriches the analysis by evaluating noise's impact on classification performance and allowing for nuanced assessments beyond binary classification
4) Advanced the state-of-the-art quantum classification F1 score on a quantum computer from 0.838 to 0.86 with a simplified architecture (6 layers versus 8).

The rest of this article is organized as follows. Section II introduces QNN, our chosen dataset, and related work. Section III provides a detailed description of our research methodology. The results are presented in Section IV, and the discussion of the results is presented in Section V. Finally, Section VI summarizes a concise overview of the main points discussed in this article.

II. BACKGROUND

A. QUANTUM NEURAL NETWORKS

QNNs manipulate qubits that hold quantum information. Qubits can exist in the standard basis states $|0\rangle$ or $|1\rangle$, or in a superposition of both, denoted as a general state $|\psi\rangle$. This superposition enables qubits to encode more complex quantum information compared with classical binary bits.

A convenient representation of qubit states is the Bloch sphere (see Fig. 1), where points on the sphere correspond to state vectors $|\psi\rangle = \cos(\theta/2)|0\rangle + e^{i\varphi}\sin(\theta/2)|1\rangle$. Here, the angular coordinates θ and φ parameterize the superposition of $|0\rangle$ and $|1\rangle$ that defines the qubit state $|\psi\rangle$. For example, $\theta = 0$ corresponds to the $|0\rangle$ state while $\theta = \pi$ indicates $|1\rangle$. Fig. 1 provides a graphical depiction of representing qubits on the Bloch sphere.

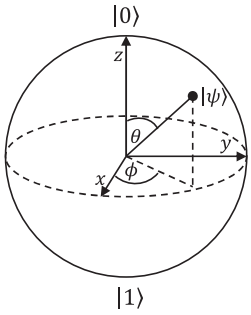

FIGURE 1. Bloch sphere—A qubit state is represeted as $|\psi\rangle = \cos(\frac{\theta}{2})|0\rangle + e^{i\varphi}\sin(\frac{\theta}{2})|1\rangle$

Before employing a QNN on classical information, the initial quantum state must be prepared according to the classical features, also known as state preparation. This initialization of the input qubits to a desired starting state is a key component of nearly all quantum algorithms [21].

The training process for QNNs has conceptual parallels to classical neural networks in utilizing feedforward and backpropagation techniques. However, the nature of what is learned during training differs fundamentally. For classical networks, training primarily involves adjusting weight and bias parameters to optimize performance. In contrast, quantum network training shifts towards fine-tuning quantum gates parameters that represent rotations and the type of entanglement operations performed on the qubits. Fig. 5 shows several known QNN architectures. A recent review paper on quantum classifiers and QNN can be found in [22].

B. DATASET

The UNSW-NB15 dataset [8], [23], [24], [25], [26] is a widely utilized [27], [28], [29], [30] compilation of generated network traffic data, specifically designed for evaluating NIDSs. It provides a realistic and diverse representation of network behavior, containing normal traffic as well as various types of attacks. Sarhan et al. [20] transformed the packet-based UNSW-NB15 dataset into a NetFlow dataset, denoting it as NF-UNSW-NB15. The conversion was undertaken to establish a standardized feature set, facilitating the consistent comparison of ML architectures across diverse NIDS datasets, and ultimately merging all different datasets into a single NetFlow-based dataset.

The NF-UNSW-NB15 dataset has been used in several ML studies [31], [32], [33], and provides a robust foundation for our QML investigation. It has 1 623 118 labeled NetFlow samples, where 72 406 (4.4%) are malicious and each sample has 12 features. To avoid biasing and to further reduce the number of features we dropped the source and destination IP

IEEE Transactions on Quantum Engineering

TABLE 1 Selected Features From the NF-UNSW-NB15 Dataset Including the Range of Each Selected Feature and the Number of Its Unique Values

Feature Name	Type	Range	Unique Values
IP Protocol	int	0 to 255	255
TCP Flags	int	0 to 31	15
Layer 7 Protocol	float	0 to 249	265
In Byte Count	int	52 to $2.68 \cdot 10^7$	12256
Out Byte Count	int	0 to $1.46 \cdot 10^7$	16047
In Packet Count	int	1 to 20038	892
Out Packet Count	int	0 to 11024	1207
Flow Duration [ms]	int	0 to $4.3 \cdot 10^6$	17377

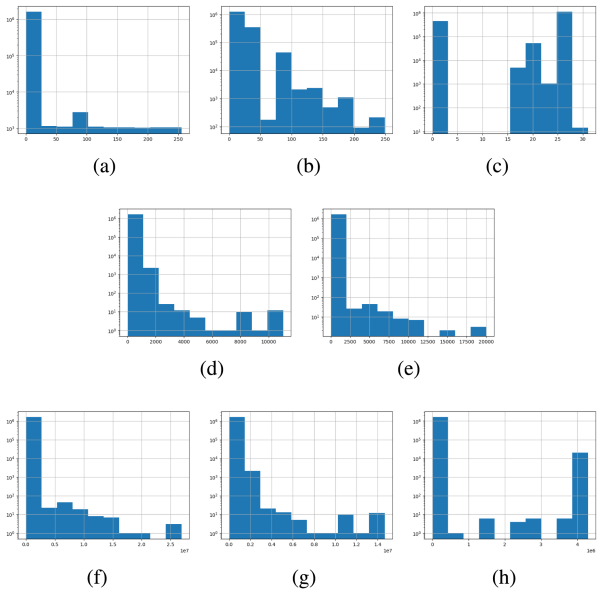

FIGURE 2. Log scale histograms of the selected features extracted from the NF-UNSW-NB15 dataset. (a) IPv4 protocol. (b) Layer 7 protocol. (c) TCP flags. (d) Out packets count. (e) In packets count. (f) In bytes count. (g) Out bytes count. (h) Flow duration.

and port features, leading us to consider only eight features. An overview of selected features and dataset statistics are detailed in Table 1 and in Fig. 2. The dataset was complete and had no missing values. The layer 7 protocol feature was classified by us as float since it has the structure of "<Number >.< Number> ." Although the TCP flags consist of distinct eight flags, we have only observed 15 unique values for that feature.

In addition, given the natural imbalance nature of networking data, we resampled the benign NetFlows to produce a balanced dataset. For that, we used the resample utility of scikit-learn with *random_state*=123. The final dataset used for our model development is balanced with 144 812 samples and eight features. Also using scikit-learn, we implemented a 15%–85% test-train split with *random_state*=1.

C. RELATED WORK

Several studies considered the use of QML for the purpose of anomaly detection in computer networks. For instance, the Moore dataset [34] was used in [16] to train a single layer QNN, consisting of a single two-qubit general unitary acting on every feature qubit and the result qubit. Similar to our approach, they produced a balanced dataset and normalized the feature values to be in the [0,1] range, followed by a single-qubit binary encoding. Their experimentation used 16 features in the first trial and 12 features in the second, yielding F1 scores of 0.8 and 0.77, respectively.

A different study [17] used a quantum convolutional neural network (QCNN) and a QSVM to perform multiclass classification. They synthesized their dataset by creating streams out of packets generated by *nmap* and *hping*. Each stream had 58 features, and all were encoded into a single qubit by successive rotation around the x-axis. They reported an accuracy 0.98 in simulation for the QCNN and also for the QSVM.

The NSL-KDD [35] and the UNSW-NB15 datasets were used in [18] to train a classical support vector machine (SVM) and a QSVM. The reported QSVM accuracies in simulation for the NSL-KDD and UNSW-NB15 datasets were 0.92 and 0.64, while the classical SVM achieved accuracies of 0.93 and 0.75, respectively. Notably, the training process utilized only 150 data samples.

The only reported quantum-based intrusion detection, with model evaluation tested on a quantum computer, is [19]. The authors subsampled the KDD CUP99 dataset [36] to obtain 700 balanced training and 300 balanced test samples. Previous studies on feature importance for this dataset led them to select five out of the original 41 features. The features were encoded using a Hadamard gate followed by a Z rotation. The proposed QNN architecture had eight layers of arbitrary single-qubit rotations and a staggered controlled not (CNOT) ring. In a noiseless simulation, they achieved a 0.983 F1 score; running on IBM's noisy quantum computers and averaging over 1000 shots for a test set containing 100 samples, they achieved an F1 score of 0.838.

In the original NF-UNSW-NB15 paper Sarhan et al. [20] employed the extra trees classifier to perform binary classification and reported 0.986 accuracy and 0.85 F1 score. A subsequent study [33] surpassed these results with F1 scores of 0.9 and 0.92 using novelty and outlier detection. Another work [31] reported an impressive accuracy of 0.998, though it did not provide an F1 score. The best F1 score is documented in [32], where they combined NF-UNSW-NB15 with other NetFlow NIDS datasets into a single dataset named NF-UQ-NIDS. They used the unified dataset to train different classifiers and calculated F1 scores for each classifier and original dataset. For the NF-UNSW-NB15 dataset, they achieved F1 scores between 0.87 and 0.987 depending on the classifier. The best performance was achieved using a deep neural network (DNN) with five dense hidden layers, nine feature input layers, and a 21-neuron output layer.

Feature analysis of the NF-USNW-NB15 dataset has been conducted in [37]. Three distinct algorithms were employed to ascertain the importance ranking of each feature, which were subsequently evaluated using both DNN and random forest classifiers. The findings revealed that all eight features were requisite for optimal accuracy performance with the DNN, whereas the random forest classifier demonstrated

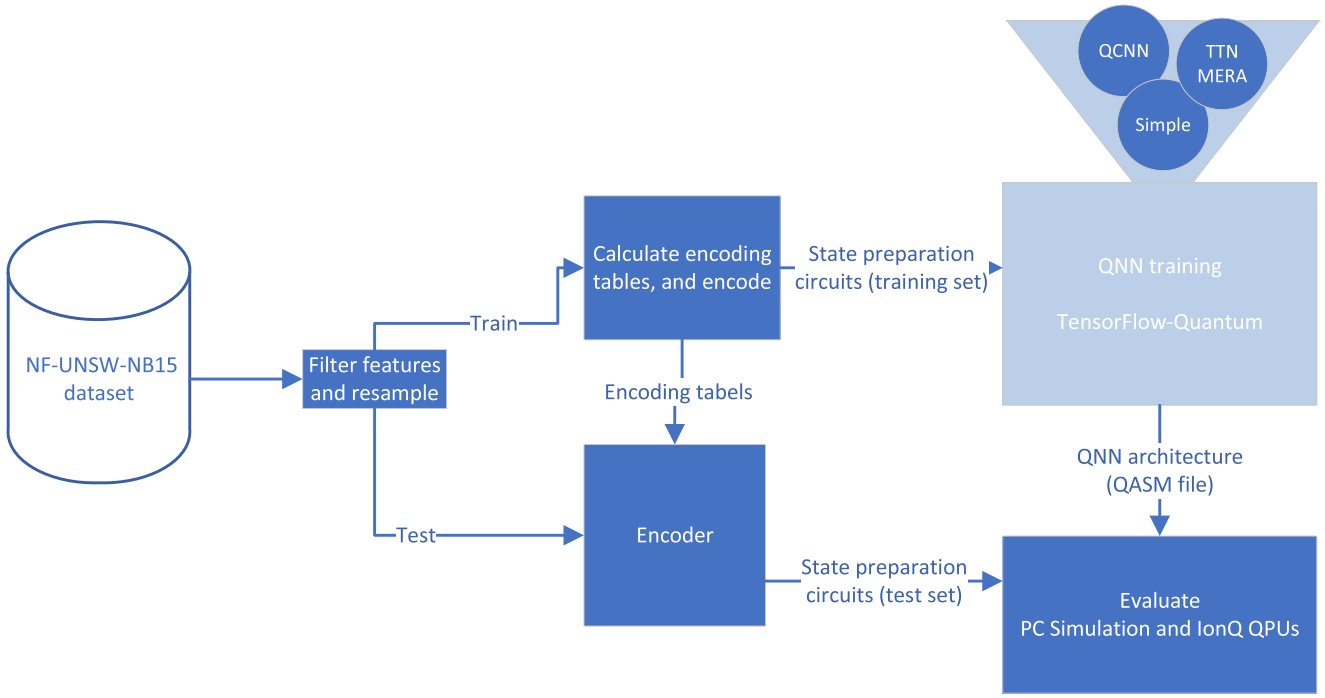

FIGURE 3. Flowchart illustrating the training and testing process of various QNNs. The workflow includes feature filtering and benign NetFlow resampling, dataset splitting, creation of classical-to-quantum encoding tables, training of networks using the TensorFlow-Quantum package, and performance evaluation in both a noisy simulation and on IonQ's quantum computers.

peak accuracy performance with just four features–namely, TCP flags, in byte count, and out byte count. In addition, the study highlighted instances where utilizing all features may not be advantageous.

It is important to note that the NF-UNSW-NB15 dataset is imbalanced. Not all studies have addressed this imbalance through resampling, leading us to focus on the F1 score, which provides an equal-weighted combination of precision and recall. Due to the lack of standardization of splits (e.g., which samples constitute training versus testing) results vary over reported studies. Therefore, reproducing and comparing results is a nontrivial task. In Section II-B we provided details on how we performed the resampling and splitting of the dataset, aiming to enhance comparability with future studies.

III. METHODOLOGY

This section explains in detail our research methodology. In Section III-A we introduce an approach for encoding classical features into qubits. We then provide an overview of different QNN architectures we evaluate in this work in Section III-B. To enhance the analysis of the performance we suggest a new performance metric, certainty factor, and provide details on its definition and the intuition behind it in Section III-C. Details of the training process including hyperparameters are provided in Section III-E and III-D. In Section III-F we detail the methodology for evaluation of the best performing QNN architecture through a realistic simulation with a noise model as well as evaluation on the IonQ's quantum computers. A high-level overview of the research process can be seen in Fig. 3.

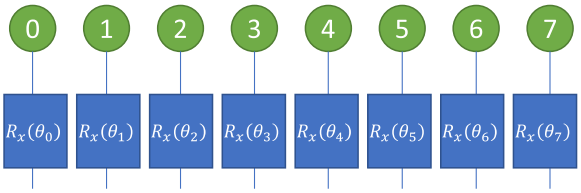

FIGURE 4. State preparation circuit used for the qubit encoding of the classical features. Each feature value is projected to an angle in a quantized range $[0, \pi]$, ensuring a minimum granularity of 0.25°.

A. DATA ENCODING

In order to use classical features in a QNN, the features need to be encoded into quantum information stored in qubits. Comparing several ways to perform the encoding [5], [13], we decided to encode each feature value in a different qubit, similarly to *qubit encoding* in [13]. In our IDS, we have a limited number of features, hence we chose a single qubit per feature. This approach avoids the need for a more intricate entangled state that would use fewer qubits but require complex state preparation, which introduces undesired noise.

For a given dataset $S = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{S}$, we encode each feature $x_i[j]$ (where j represents the feature and i denotes the sample) as a rotation around the x-axis. To perform the rotation we first project each feature onto an angle $\theta_i[j] \in [0, \pi]$. Then, we executed an $R_x(\theta_i[j])$ on each feature qubit, transforming it from the default $|0\rangle$ state into $\psi_i[j] = \cos(\frac{\theta_i[j]}{2})|0\rangle + \sin(\frac{\theta_i[j]}{2})|1\rangle$; see Fig. 4.

A naive way to perform the projection is to normalize the feature space, with the formula

$$\theta_i[j] = \pi \frac{\boldsymbol{x}_i[j]}{\max_k \boldsymbol{x}_k[j]}.$$

Although attractive due to simplicity, this approach may create extremely fine rotations. In practice, such an encoding schema is impossible to generate with current quantum hardware due to imperfections in the implementation of the quantum gates that limit the rotation accuracy.

In order to mitigate this issue, we avoid fine rotations by projecting each feature space to a quantized $[0, \pi]$ range, where we limit the rotation's granularity to 0.25°. For features that have a limited number of unique values (see Table 1), we split the $[0, \pi]$ range according to the number of unique values, and then assign a projected value categorically. For features that have many unique values, the projection from the feature space to $[0, \pi]$ was done by binning the values according to their percentile. For example, if we were to split the range $[0, \pi]$ into 100 bins, then a feature value that is in the x percentile will be projected to $x \cdot \frac{\pi}{100}$.

B. QNN ARCHITECTURES

In this work, we investigated four different QNN architectures: TTN [13], MERA [13], Quantum Convolutional Neural Network (QCNN) [12], and a simple architecture built by stacking two-qubit Pauli rotation gates blocks, similar to what was used in [38]. We refer to that architecture as "Simple"; we illustrate all architectures in Fig. 5. U_i and D_i represent a general quantum gate. In the QCNN architecture, see Fig. 5(d), all the gates with the same index have identical parameters.

In each of the architectures, we modeled a generic two-qubit gate using two generic one-qubit gates followed by a CNOT, as shown in Fig. 6. We note that one needs three CNOTs to accurately decompose any general two-qubit gate into CNOTs and general one-qubit gates [39]. However, our goal is to design a QNN that can run on NISQ computers, hence we chose to have only a single CNOT as a balance between the expressibility of the gate and its fidelity on a noisy quantum computer.

All evaluated architectures have a single qubit output, which we measure over many shots, and the majority vote of these measurements is the network's prediction. In a noiseless simulation, we can look at the state of the output qubit and predict accordingly. The result qubit in the "Simple" architecture is initialized in the $|-\rangle$ state and measured at the end in the x-basis, while in all the other architectures z-basis measurement is used.

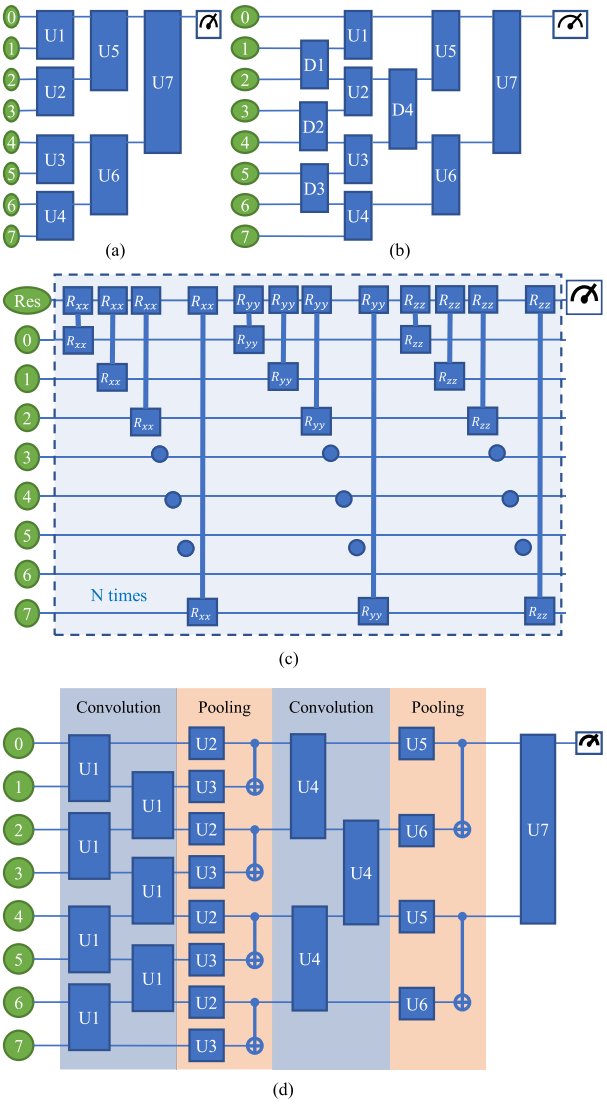

FIGURE 5. Evaluated QNN architectures. Each generic two-qubit gate was modeled using two generic one-qubit gates followed by a Controlled Not (CNOT), as shown in Fig. 6. Each of the eight features was encoded as a rotation around the x-axis for qubits 0–7. In (c) the Res qubit is initialized to the $|-\rangle$ state, and measured in the x-basis, while all the other measurements are z-basis. (a) TTN. (b) MERA. (c) "Simple." (d) QCNN.

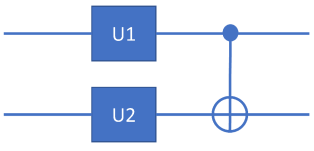

FIGURE 6. Two-qubit gate modeled by two generic one-qubit gates and a CNOT.

C. CERTAINTY FACTOR

We introduce a confidence metric, referred to as the certainty factor and denoted $\mathcal{C}$, to quantify the model's inherent sharpness and degree of separation between the predicted class probability distribution versus other classes. This continuous measure of predictive confidence ranges from -1 to 1 for each sample, enabling analysis of prediction quality on a per-sample basis.

Specifically, the certainty factor approximates the relative entropy or divergence between the predicted outcome probability distribution and an ideal fully confident singular distribution. For a labeled True example where the model

```python
def hinge_accuracy(y_true, y_pred):
    y_true = tf.squeeze(y_true) > 0.0
    y_pred = tf.squeeze(y_pred) > 0.0

    result = tf.cast(y_true == y_pred,
                     tf.float32)

    return tf.reduce_mean(result)
```

LISTING 1: Listing 1: Custom accuracy function that correctly handles the ±1 labels [38].

predicts a $|\psi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle$ state, with $|0\rangle$ expected for true, the certainty factor is defined as $\mathcal{C} = |\alpha_0|^2 - |\alpha_1|^2$.

When $\mathcal{C} = 1$, the predicted distribution is maximally peaked on the expected outcome, equivalent to a fully confident singular distribution. If $\mathcal{C} = -1$, the distribution is inverted and predictions will be incorrect. For $\mathcal{C} = 0$ the distribution is maximally flat, indicating completely random guesses. As $|\mathcal{C}|$ increases from 0, the predicted distribution sharpens, reflecting greater confidence.

By tracking certainty factor distributions, we can thus assess the degradation of predictive sharpness and separation margins as noise intensifies. $\mathcal{C}$ identifies vulnerable predictions likely to be impacted by errors before noise is introduced.

D. LEARNING INFRASTRUCTURE

In this work, we used TensorFlow-Quantum [40] v0.7.2 with Python 3.8, and TensorFlow v2.7.0. Using KERAS, we performed the learning procedure and used the Hinge loss function [41], not before converting the True and False labels to 1 and −1. For the accuracy metric, we used a custom accuracy function that correctly handles a set of ±1 labels, see Listing 1.

E. HYPERPARAMETERS SEARCH

In order to gain insight into the performance and trainability of the different architectures, we performed a detailed grid search over hyperparameters. We tried batch sizes of 16 and 32. The different learning rates were 0.1, 0.05, 0.02, 0.015, 0.01, 0.005, and 0.001. We used two different optimizers, Adam [42] and stochastic gradient descent (SGD), with momentum 0, 0.2, 0.3, and decay rate 0, 0.001, 0.01. Finally, we tried all four architectures, and in the "Simple" architecture we tried 1, 2, 4, and 6 layers. For each layer, we tried ZZ XX YY; ZZ XX; XX YY; and ZZ YY as the possible entanglement rotations.

F. EVALUATING THE BEST ARCHITECTURE WITH NOISE

After we identified that the "Simple" QNN architecture had the best performance in the noiseless simulation (see Section IV-A), we decided to see whether and how noise affects the performance of our classification system. We choose to use IonQ's harmony quantum processing unit (QPU) [43] as our target hardware because its native gates match perfectly with the R_{xx} and R_{yy} gates used in the "Simple" network. Moreover, the ion trap has all-to-all connectivity that is heavily used in our QNN architecture.

1) HARMONY EVALUATION

As a first step, we used IonQ's noisy simulator to retest the QNN performance. This required creating a different quantum circuit for every sample in our test set. These circuits are a concatenation of two circuits—the feature embedding circuit (state preparation) and the network itself. We use the serialization capabilities of Cirq-IonQ Python package to[1] convert the combined circuit to JSON format. Next, we use IonQ's REST API to send the inference simulation tasks and, finally, retrieve the results using the same API.

The last evaluation step was to run the inference circuits on the harmony machine. It is very expensive to run many circuits with many shots on a physical quantum computer; thus, we decided to evaluate our proposed QNN using a smaller random sample of our test set. Due to our funding sources, the experiments on the QPU were done via Amazon Braket Cloud API and not directly using IonQ's RESET API. In our first trial, we ran a balanced test set of 60 samples with 120 shots per task. The second trial consisted of 40 samples and 200 shots per task.

2) ARIA-1 EVALUATION

The harmony experiments yielded disappointing results, demonstrating low classification performance as discussed in Section IV-C. After contacting IonQ's experts, we were advised to try their error mitigation techniques [44], [45] or to use the newer QPU Aria-1 [46] for its improved noise error rates. We decided to experiment with the newer hardware. Aria-1 is also an all-to-all machine that has the same native gates as harmony, so it was a natural fit for our "Simple" QNN architecture.

Similar to the harmony experiments, our first step was to evaluate the QNN on IonQ's noisy simulator. To that end, we used the same procedure described in Section III-F1. Next, we again used Amazon Braket to run on the QPU itself. To our surprise, we were not able to run our six-layer architecture on Aria-1.

With support from IonQ, we determined that our circuits exceeded the current limit for the number of unique gates allowed in a single channel. We considered consolidating gates with close rotations or evaluating a shallower architecture. We chose the latter, evaluating a two-layer XY architecture with 200 shots per task. Keeping our limited resources in

[1] https://github.com/quantumlib/Cirq/tree/master/cirq-ionq

IEEE Transactions on Quantum Engineering

TABLE 2 Comparison of the Best F1 Score and Average Convergence Epochs for Different Network Architectures and Optimizers in a Noiseless Simulation

Model	Max F1 Score	Average last epoch of improvement	
		Adam	SGD
CNN	0.636	4.1	13.2
MERA	0.585	2.5	10.7
Simple	0.907	2.7	11.4
TTN	0.586	1.8	9.8

It is clear that our Simple architecture greatly outperforms all the others. Moreover, Adam converges much faster than SGD, but both of them converge relatively fast.

TABLE 3 Simple Architecture F1 Score Results for Different Number of Layers and Their Type in a Noiseless Simulation

Number of Layers	Best F1 Score			
	XY	ZX	ZY	ZXY
1	**0.82**	0.77	0.77	0.81
2	0.877	0.847	0.786	**0.879**
4	**0.894**	0.867	0.837	0.88
6	**0.907**	0.893	0.819	0.86

Bold font identifies the best score for each number of layers.

TABLE 4 Best F1 Score Comparison Between Adam and SGD Optimizers for Different Learning Rates in the Simple Architecture in a Noiseless Simulation

Learning Rate	Adam	SGD
0.1	0.874	0.894
0.05	0.873	0.881
0.02	0.858	0.907
0.01	0.873	0.885
0.005	0.893	0.873
0.001	0.869	0.877

mind, we began with a fairly small balanced test set, ultimately increasing to 776 balanced samples (3.5% of our complete test set).

IV. RESULTS

In this section, we present and discuss the results of our research in the following sections. Section IV-A holds the key results from our noiseless architecture exploration experiments. In Section IV-B, we evaluate our simple architecture on IonQ's harmony and Aria-1 noisy simulators. We analyze noise impact on classification using the introduced certainty factor. Then, in Sections IV-C and IV-D we set forth the results from our Harmony and Aria-1 QPU experiments.

A. NOISELESS QNN ARCHITECTURE COMPARISON

Tables 2–4 summarize our findings from architecture exploration experiments. Table 2 gives the best F1 score value obtained for each QNN architecture in our hyperparameter search. It demonstrates that the simple architecture has outperformed all the other architectures with an F1 score of 0.907. In addition, as given in Table 3, the F1 score improves with an increase in network depth. Moreover, one can also observe that the XY layer type achieves the best performance compared to the other layer types.

To assess the convergence speed of different architectures, we determined the last epoch, at which the loss exhibited

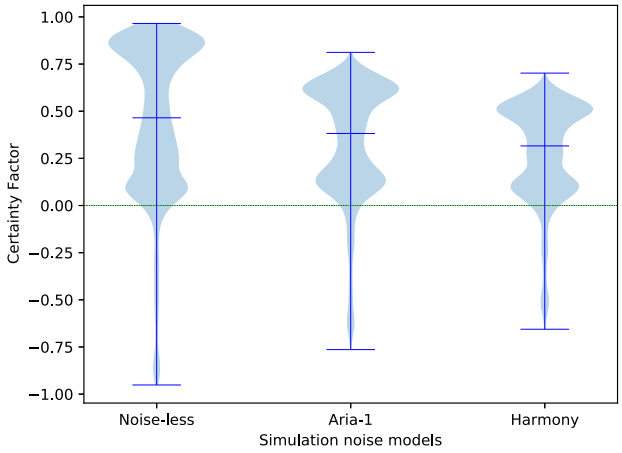

FIGURE 7. Violin plots depicting the distribution of certainty factor for our six-layer Simple QNN architecture in three simulation modes: noiseless, Aria-1 noise model, and harmony noise model. The green dashed line represents zero certainty, where everything above it is a correct prediction while everything below is a wrong prediction. It is clear to see from the figure that as we have more noise, the distribution is squashed into zero certainty, and as expected Harmony has more noise compared to Aria-1.

improvement. The results are summarized in Table 2, demonstrating that the Adam optimizer achieves significantly faster convergence compared with the SGD optimizer. Furthermore, both optimizers outperform classical neural networks in terms of convergence speed. A comparison between different learning rates coupled with the optimizer used for the training of the simple architecture is given in Table 4. Overall, Adam converges faster, but SGD yields a better classification network.

Ultimately, the best F1 score was achieved using the following hyperparameters.

1) Optimizer—SGD with a decay of 0.001.
2) Learning rate—0.02.
3) QNN architecture—Simple with six layers of type XY.
4) Batch size—32.

We did not observe any obvious benefit from using a batch size of 16 versus 32.

B. NOISY SIMULATION AND CERTAINTY FACTOR

The F1 scores obtained on IonQ's noisy simulator for our six-layer architecture using harmony and Aria-1 noise models are 0.8789 and 0.886, respectively, whereas the noiseless simulation had an F1 score of 0.907.

To analyze the noise impact on the performance of our QNN predictions, we use the certainty factor, which we defined in Section III-C, and plot its distribution violin plot for the noisy and the noiseless simulation on our test set. Fig. 7 shows the comparison; from it, we can see that as the noise increases the distribution is squeezed toward the middle. This symmetric deformation is expected, as the noise should have the same effect on correct and wrong predictions.

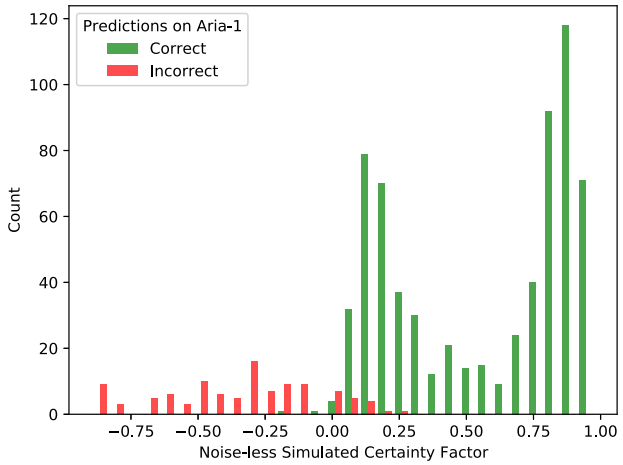

FIGURE 8. Histogram of the 776 Aria-1 inference runs, binned according to their original certainty factor in the noiseless simulation. The green and red bins represent correct and incorrect Aria-1 inference predictions, respectively. The figure demonstrates that samples with a certainty factor close to zero are more likely to be affected by noise and result in a classification flip.

TABLE 5 Confusion Matrices for Both of the Harmony Experiments

Experiment	120 shots – 60 examples		200 shots – 40 examples	
	Predicted 0	Predicted 1	Predicted 0	Predicted 1
Actual 0	12	18	20	0
Actual 1	29	1	20	0

In the first experiment, the noise flipped the prediction result, while in the second experiment, all of the predictions were 0.

Fig. 8 shows the impact of noise on the classification in our 776 Aria-1 inference runs using the two-layer simple QNN architecture. It demonstrates how samples with a certainty factor closer to zero are more likely to be affected and flip the noiseless prediction. The certainty factor can predict the susceptibility of a prediction. In the absence of noise, and with a sufficient number of shots, all the bins corresponding to a positive certainty factor would be green, while those corresponding to a negative certainty factor would be red.

C. HARMONY EXPERIMENTS

In our first experiment, we used a balanced test set of 60 examples. For each of them, we ran a 120-shot task. After seeing the disappointing results, we decided to run another 40 examples and increased the number of shots to 200. Table 5 gives the confusion matrices for both of our harmony experiments.

Following another set of disappointing results, we wanted to rule out the possibility that the source of the discrepancy between the simulation and the quantum computer has to do with rotation angle precision. Fig. 9 shows the noiseless simulated prediction performance as we decrease the number of digits after the decimal point. The graph clearly shows a degradation in F1 performance, but not so severe as we see while running on harmony QPU. Thus, we conclude that the introduced error is the result of overall quantum computer noise error vice a loss of precision when moving from simulation to instantiation. As discussed above, these poor results and discussions with IonQ led us to port our experiment to Aria-1.

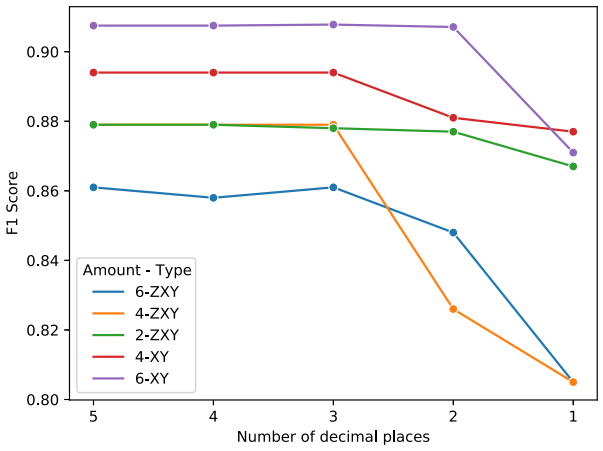

FIGURE 9. Noiseless Harmony classification performance as a function of the number of decimal points used in the quantum gate rotation angles, plotted for different numbers and types of layers. The performance degradation is most significant when transitioning from two decimal places to one.

TABLE 6 F1 Score Comparison of Our two-Layer Architecture for the Complete Test Set and the Randomly Selected Subset of 776 Examples

Scenario	Complete Test Set	Random Test Subset
Noiseless Simulation	0.877	0.876
Noisy Simulation	0.848	0.852
Aria-1 QPU	-	0.86

The noisy simulation had 1000 shots while the the hardware runs had 200. We can see that the noiseless performance for the two datasets is almost identical and that there is a good agreement between the noisy simulation performance and the hardware performance.

TABLE 7 Confusion Matrices for the Aria-1 Noisy Simulation and Hardware Experiments

Experiment	Noisy simulation		Hardware	
	Predicted 0	Predicted 1	Predicted 0	Predicted 1
Actual 0	348	42	342	48
Actual 1	68	318	58	328

In both of the experiments, we had 776 examples with 1000 shots in simulation and 200 shots for the hardware runs.

D. ARIA-1 EXPERIMENTS

Table 6 presents a summary of the F1 score performance achieved by our two-layer architecture. The evaluation encompasses both the complete test set and a randomly selected subset of 776 examples. The scenarios include a noiseless simulation, a noisy simulation utilizing IonQ's Aria-1 noise model, and execution on the Aria-1 machine. Notably, it is worth mentioning that the F1 scores in the noiseless simulation for both the complete test set and the random subset are nearly identical. Furthermore, the performance in the noisy simulation serves as a good approximation to the results obtained on the Aria-1 machine. Table 7 presents the confusion matrices for the subset classification on the noisy simulation and the Aria-1 machine.

IEEE Transactions on Quantum Engineering

TABLE 8 F1 Score Comparison Between This Work and Other Classical and Quantum NIDSs Found in the Literature

Classifier	Dateset	F1 Score	Evaluation Method	Comments
Extra Trees [20]	NF-UNSW-NB15	0.85	Classical	
Novelty Detection [33]	NF-UNSW-NB15	0.9	Classical	
Outlier Detection [33]	NF-UNSW-NB15	0.92	Classical	
Random Forest [32]	NF-UNSW-NB15	0.919	Classical	Trained on a combination of several NIDS datasets
Seven layers DNN [32]	NF-UNSW-NB15	0.987	Classical	
SVM [18]	UNSW-NB15	0.75	Classical	
QSVM [18]	UNSW-NB15	0.64	Noise-less quantum simulation	
Eight layers QNN [19]	KDD CUP99	0.983	Noise-less quantum simulation	Eight layers of arbitrary single-qubit rotations and a staggered CNOT ring
	KDD CUP99	0.838	IBM's quantum computers	
Simple 6 XY layers	NF-UNSW-NB15	0.907	Noise-less quantum simulation	
Simple 6 XY layers	NF-UNSW-NB15	0.886	Aria-1 noisy quantum simulation	
Simple 2 XY layers	NF-UNSW-NB15	0.848	Aria-1 noisy quantum simulation	
Simple 2 XY layers	NF-UNSW-NB15	0.86	IonQ's Aria-1 quantum computer	Tested on 776 random samples from the test set

Our work has achieved the best F1 score on a quantum computer known to date. This achievement underscores the robustness of our feature encoding and lean QNN architecture, which exhibit minimal performance degradation due to noise. Moreover, our results demonstrate comparable performance to more complex ML techniques trained on the same dataset. We note that some of the works below were trained and tested on different datasets, and for those that used NF-UNSW-NB15 for training, they didn't specify the train-test split. In Section II-B we have explained how to recreate our test-train split.

As discussed in Section III-F, we were only able to simulate our proposed six-layer simple architecture; this six-layer architecture yielded an F1 score of 0.886 using Aria-1's noisy simulator. Given the strong alignment between the two-layer simulation and the QPU experiment results, we anticipate being able to reproduce this classification performance in the six-layer architecture on Aria-1 once IonQ implements the necessary enhancements to run circuits with more unique two-qubit gates per channel.

V. DISCUSSION

The results of our study demonstrate a significant advancement in the accuracy of quantum computing-based classification for network intrusion detection. Through the implementation of a simple QNN classifier, we achieved F1 score performance comparable to those obtained by a more complex classical ML techniques [20], [33], [32]; see Table 8. It is noteworthy to mention that some of the existing works in this domain utilized different datasets, and among those that employed the same dataset as ours, the specific splits for training were not specified. Moreover, some studies [32] have tested against the NF-UNSW-NB15 dataset but trained using a combination of datasets. Our achievement not only underscores the potential of QML but also demonstrates the practicality of employing quantum algorithms in real-world cybersecurity applications.

Furthermore, our success in instantiating the classifier on a current NISQ-era machine was enabled by a combination of innovative classical feature encoding techniques and the lean architecture of our custom QNN. The strategic choice of an all-to-all quantum computer, whose native gates were a perfect fit with our QNN's R_{xx} and R_{yy} quantum gates, ensured minimal performance degradation attributable to noise.

Incorporating the certainty factor into the analysis, provided a deeper understanding of how classification performance is affected by noise. This novel metric enabled us to systematically analyze the impact of quantum noise on classifier performance at the system level.

Moreover, the introduction of the certainty factor opens up intriguing possibilities for future research endeavors. This factor can serve as a foundation for extending the classification output to incorporate measures of uncertainty, akin to the principles underlying Bayesian networks. By integrating uncertainty assessments, we move beyond binary classifications, allowing for nuanced interpretations of the results. The certainty factor not only enriches the depth of information provided by our intrusion detection system but also lays the groundwork for more sophisticated decision-making processes in complex and dynamic network environments. Such an advancement holds promise for advancing the state of the art in intrusion detection, which would lead to more resilient and adaptive cybersecurity architectures.

Throughout our experimentation, we worked within an interesting constraint. Given our limited computational resources, we found ourselves tasked with the decision of allocating an optimal number of shots for each inference process. This scenario encapsulates an interesting tradeoff: allocating more shots per inference to enhance confidence levels for individual results, or distributing shots across more inferences, thereby maximizing the overall reliability of our evaluation procedure. To address this challenge, we adopted a randomized approach to sample our testset using various seeds and sample sizes. Using the statistical insights derived from our noisy simulation enabled us to make informed decisions about the allocation of shots. In addition, in an operational setting, one might monitor the certainty factor and adaptively run additional shots in case of low confidence predictions.

The current methodology could be improved in several ways to further optimize performance. One such improvement is using fewer features from the dataset, balancing the information gained from every feature addition with the introduced noise from a more complex QNN. One might use the feature importance analysis done in [37].

Another avenue could be to utilize multiple QNNs where through bagging and random feature selection, similar to random forest algorithm, an ensemble of weak classifiers could outperform a strong classifier (e.g., larger QNNs) operating on all features. By effectively leveraging the parallel computational power of quantum machines in this ensemble framework, we could further amplify the efficacy of our

intrusion detection system. Finally, as suggested by IonQ, implementing noise mitigation strategies represents another possible avenue for classification performance improvement.

VI. CONCLUSION

This article represents a significant advancement in the utilization of quantum computing for intrusion detection. Compared with sophisticated classical ML models we have achieved similar F1 score performance while having fast convergence, indicating the practical potential of quantum algorithms in real-world security applications. We are also among the pioneers in leveraging QNN on a physical quantum computer for network intrusion detection, potentially achieving the highest performance known to date. Through innovative feature encoding techniques and a streamlined QNN architecture tailored to a compatible quantum platform, we have mitigated deterioration in performance caused by noise. The introduction of a certainty factor enriched our analysis, offering insights into the impact of noise, and paves the way for incorporating uncertainty measures into future research, yielding advances in intrusion detection and cybersecurity frameworks and in QML in general. Moving forward, avenues for future work include exploring alternative features from NetFlow data, employing multiple QNNs in an ensemble fashion, and implementing noise mitigation techniques.

REFERENCES

[1] "2023 Global Threat Report|CrowdStrike," [Online]. Available: https://www.crowdstrike.com/global-threat-report/

[2] "2023 Data Breach Investigations Report," [Online]. Available: https://www.verizon.com/business/resources/reports/dbir/

[3] K. Shaukat, S. Luo, V. Varadharajan, I. A. Hameed, and M. Xu, "A survey on machine learning techniques for cyber security in the last decade," *IEEE Access*, vol. 8, pp. 222310–222354, 2020, doi: 10.1109/ACCESS.2020.3041951.

[4] Y. Liu, X. Wang, S. Li, and J. Wu, "Machine learning and deep learning methods for intrusion detection: A review," *Appl. Sci.*, vol. 9, no. 20, 2019, Art. no. 4396, doi: 10.3390/app9204396.

[5] S. Kaddoura, "Classification of malicious and benign websites by network features using supervised machine learning algorithms," in *Proc. 5th Cyber Secur. Netw. Conf.*, 2021, pp. 36–40, doi: 10.1109/CSNet52717.2021.9614273.

[6] P. Podder, S. Bharati, M. R. H. Mondal, P. K. Paul, and U. Kose, "Artificial neural network for cybersecurity: A comprehensive review," 2021, doi: 10.48550/arXiv.2107.01185.

[7] M. Rabbani et al., "A review on machine learning approaches for network malicious behavior detection in emerging technologies," *Entropy*, vol. 23, no. 5, May 2021, Art. no. 529, doi: 10.3390/e23050529.

[8] V. Kanimozhi and P. Jacob, "UNSW-NB15 dataset feature selection and network intrusion detection using deep learning," *Int. J. Recent Technol. Eng.*, vol. 7, no. 5S2, pp. 443–446, Jan. 2019. [Online]. Available: https://www.ijrte.org/portfolio-item/ES2080017519/

[9] M. Benedetti, E. Lloyd, S. Sack, and M. Fiorentini, "Parameterized quantum circuits as machine learning models," *Quantum Sci. Technol.*, vol. 4, no. 4, Nov. 2019, Art. no. 043001, doi: 10.1088/2058-9565/ab4eb5.

[10] F. Tacchino et al., "Variational learning for quantum artificial neural networks," *IEEE Trans. Quantum Eng.*, vol. 2, pp. 1–10, 2021, doi: 10.1109/TQE.2021.3062494.

[11] W. Li, Z. Lu, and D.-L. Deng, "Quantum neural network classifiers: A tutorial," *SciPost Phys. Lecture Notes*, Aug. 2022, Art. no. 61, doi: 10.21468/SciPostPhysLectNotes.61.

[12] I. Cong, S. Choi, and M. D. Lukin, "Quantum convolutional neural networks," *Nature Phys.*, vol. 15, no. 12, pp. 1273–1278, Dec. 2019, doi: 10.1038/s41567-019-0648-8.

[13] E. Grant et al., "Hierarchical quantum classifiers," *NPJ Quantum Information*, vol. 4, no. 1, Dec. 2018, Art. no. 65, doi: 10.1038/s41534-018-0116-9.

[14] J. Preskill, "Quantum computing in the NISQ era and beyond," *Quantum*, vol. 2, Aug. 2018, Art. no. 79, doi: 10.22331/q-2018-08-06-79.

[15] N. P. de Leon et al., "Materials challenges and opportunities for quantum computing hardware," *Science*, vol. 372, no. 6539, Apr. 2021, Art. no. eabb2823, doi: 10.1126/science.abb2823.

[16] M. Zhang, B. Lv, and Z.-S. Liu, "Network attack traffic recognition based on quantum neural network," in *Proc. 7th Int. Conf. Comput. Intell. Appl.*, 2022, pp. 71–75, doi: 10.1109/ICCIA55271.2022.9828461.

[17] M. Kalinin and V. Krundyshev, "Security intrusion detection using quantum machine learning techniques," *J. Comput. Virol. Hacking Techn.*, vol. 19, pp. 125–136, Jun. 2022, doi: 10.1007/s11416-022-00435-0.

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

[19] C. Gong, W. Guan, A. Gani, and H. Qi, "Network attack detection scheme based on variational quantum neural network," *J. Supercomputing*, vol. 78, no. 15, pp. 16876–16897, Oct. 2022, doi: 10.1007/s11227-022-04542-z.

[20] M. Sarhan, S. N. Layeghy Moustafa, and M. Portmann, "NetFlow datasets for machine learning-based network intrusion detection systems," in *Big Data Technologies and Applications,* ser. *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, Z. Deze, H. Huang, R. Hou, S. Rho, and N. Chilamkurti, Eds. Berlin, Germany: Springer, 2021, pp. 117–135, doi: 10.1007/978-3-030-72802-1_9.

[21] X.-M. Zhang, T. Li, and X. Yuan, "Quantum state preparation with optimal circuit depth: Implementations and applications," *Phys. Rev. Lett.*, vol. 129, no. 23, Nov. 2022, Art. no. 230504, doi: 10.1103/PhysRevLett.129.230504.

[22] W. Li and D.-L. Deng, "Recent advances for quantum classifiers," *Sci. China Phys., Mechanics Astron.*, vol. 65, no. 2, Dec. 2021, Art. no. 220301, doi: 10.1007/s11433-021-1793-6.

[23] N. Moustafa and J. Slay, "UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in *Proc. Mil. Commun. Inf. Syst. Conf.*, 2015, pp. 1–6, doi: 10.1109/MilCIS.2015.7348942.

[24] N. Moustafa, G. Creech, and J. Slay, "Big Data analytics for intrusion detection system: Statistical decision-making using finite Dirichlet mixture models," in *Data Analytics and Decision Support for Cybersecurity: Trends, Methodologies and Applications,* ser. *Data Analytics*, I. Palomares Carrascosa, H. K. Kalutarage, and Y. Huang, Eds., Berlin, Germany: Springer, 2017, pp. 127–156, doi: 10.1007/978-3-319-59439-2_5.

[25] N. Moustafa and J. Slay, "The evaluation of network anomaly detection systems: Statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set," *Inf. Secur. J., A Glob. Perspective*, vol. 25, no. 1-3, pp. 18–31, Apr. 2016, doi: 10.1080/19393555.2015.1124946.

[26] N. Moustafa, J. Slay, and G. Creech, "Novel geometric area analysis technique for anomaly detection using trapezoidal area estimation on large-scale networks," *IEEE Trans. Big Data*, vol. 5, no. 4, pp. 481–494, Dec. 2019, doi: 10.1109/TBDATA.2017.2715166.

[27] Y. Yang, K. Zheng, B. Wu, Y. Yang, and X. Wang, "Network intrusion detection based on supervised adversarial variational auto-encoder with regularization," *IEEE Access*, vol. 8, pp. 42169–42184, 2020, doi: 10.1109/ACCESS.2020.2977007.

[28] M. Shahin, F. F. Chen, H. Bouzary, A. Hosseinzadeh, and R. Rashidifar, "A novel fully convolutional neural network approach for detection and classification of attacks on Industrial IoT devices in smart manufacturing systems," *Int. J. Adv. Manuf. Technol.*, vol. 123, no. 5, pp. 2017–2029, Nov. 2022, doi: 10.1007/s00170-022-10259-3.

[29] J. B. Awotunde, C. Chakraborty, and A. E. Adeniyi, "Intrusion detection in Industrial Internet of Things network-based on deep learning model with rule-based feature selection," *Wirel. Commun. Mobile Comput.*, vol. 2021, Sep. 2021, Art. no. e7154587, doi: 10.1155/2021/7154587.

[30] J. Gu and S. Lu, "An effective intrusion detection approach using SVM with naïve Bayes feature embedding," *Comput. Secur.*, vol. 103, Apr. 2021, Art. no. 102158, doi: 10.1016/j.cose.2020.102158.

[31] P. Jayalaxmi, G. Kumar, R. Saha, M. Conti, T.-H. Kim, and R. Thomas, "Debot: A deep learning-based model for bot detection in Industrial Internet-of-Things," *Comput. Elect. Eng.*, vol. 102, 2022, Art. no. 108214, doi: 10.1016/j.compeleceng.2022.108214.

[32] M. Vishwakarma and N. Kesswani, "DIDS: A deep neural network based real-time intrusion detection system for IoT," *Decis. Analytics J.*, vol. 5, Dec. 2022, Art. no. 100142, doi: 10.1016/j.dajour.2022.100142.

[33] P. Russell, M. A. Elsayed, B. Nandy, N. Seddigh, and N. Zincir-Heywood, "On the fence: Anomaly detection in IoT networks," in *Proc. IEEE/IFIP Netw. Operations Manage. Symp.*, 2023, pp. 1–4, doi: 10.1109/NOMS56928.2023.10154271.

[34] A. Moore, D. Zuev, and M. Crogan, "Discriminators for use in flow-based classification," Dept. Comput. Sci., Queen Mary Univ. London, London, U.K., Tech. Rep., RR-05-13, 2013. [Online]. Available: https://qmro.qmul.ac.uk/xmlui/bitstream/handle/123456789/5050/RR-05-13.pdf

[35] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani, "Toward developing a systematic approach to generate benchmark datasets for intrusion detection," *Comput. Secur.*, vol. 31, no. 3, pp. 357–374, May 2012, doi: 10.1016/j.cose.2011.12.012.

[36] "KDD Cup 1999 Data," [Online]. Available: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

[37] M. Sarhan, S. Layeghy, and M. Portmann, "Feature analysis for machine learning-based IoT intrusion detection," 2022, doi: 10.48550/arXiv.2108.12732.

[38] "MNIST classification | TensorFlow Quantum," [Online]. Available: https://www.tensorflow.org/quantum/tutorials/mnist

[39] G. Vidal and C. M. Dawson, "Universal quantum circuit for two-qubit transformations with three controlled-NOT gates," *Phys. Rev. A*, vol. 69, no. 1, Jan. 2004, Art. no. 010301, doi: 10.1103/PhysRevA.69.010301.

[40] M. Broughton et al., "TensorFlow quantum: A software framework for quantum machine learning," 2021, doi: 10.48550/arXiv.2003.02989.

[41] C. Gentile and M. K. K. Warmuth, "Linear Hinge Loss and Average Margin," in *Advances in Neural Information Processing Systems*, vol. 11. Cambridge, MA, USA: MIT Press, 1998, doi: 10.5555/3009055.3009087.

[42] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017, doi: 10.48550/arXiv.1412.6980.

[43] "IonQ Harmony," [Online]. Available: https://ionq.com/quantum-systems/harmony

[44] "Debiasing and sharpening," [Online]. Available: https://ionq.com/resources/debiasing-and-sharpening

[45] A. Maksymov, J. Nguyen, Y. Nam, and I. Markov, "Enhancing quantum computer performance via symmetrization," 2023, doi: 10.48550/arXiv.2301.07233.

[46] "IonQ Aria," [Online]. Available: https://ionq.com/quantum-systems/aria

Alon Kukliansky received the B.Sc degree in electrical and computer engineering and the M.Sc degree in electrical engineering from Tel-Aviv University, Tel Aviv, Israel, in 2008 and 2016, respectively. He is currently working toward the Ph.D. degree in quantum computing with the Department of Electrical and Computer Engineering, Naval Postgraduate School, Monterey, CA, USA.

His research interests include quantum computing and computer architectures.

Marko Orescanin (Member, IEEE) received the Ph.D. degree in electrical and computer engineering from the University of Illinois Urbana-Champaign, Champaign, IL, USA, in 2010.

From 2011 to 2019, he was with Bose Corporation, MA, USA, where he primarily worked on research and advanced development of signal processing and machine learning algorithms for audio and speech enhancement in consumer electronics. He left Bose as a Senior Manager of AI and Data group with focus on consumer electronics business unit. Since 2019, he has been an Assistant Professor with the Computer Science Department, Naval Postgraduate School, Monterey, CA, USA. His research interests include signal processing, machine learning, artificial intelligence, Bayesian deep learning, and cyber-physical system security.

Chad Bollmann (Senior Member, IEEE) received the B.S. degree in ocean engineering from the U.S. Naval Academy, Annapolis, MD, USA, in 1996, the S.M.s degree in technology and policy and nuclear engineering from the Massachusetts Institute of Technology, Cambridge, MA, USA, in 1998, and Ph.D. degree in electrical engineering from Naval Postgraduate School (NPS), Monterey, CA, USA, in 2018.

After entering the U.S. Navy in 1992, from 1998–2014 he was the Submarine Warfare Officer, most recently as the Executive Officer of the USS NEVADA (GOLD) (SSBN-733). He is currently working as an Assistant Professor of electrical and computer engineering, NPS. He is currently a Captain in the US Navy and co-appointed as a Permanent Military Professor. He is the Director of the NPS Center for Cyber Warfare. His teaching and research interests include cyber-physical system security, network traffic analysis, and non-Gaussian statistical signal processing.

Theodore Huffmire received the A.B. degree in computer science from Princeton University, Princeton, NJ, USA, in 1997, and the Ph.D. degree in computer science from the University of California, Santa Barbara, Santa Barbara, CA, USA, in 2007.

He is currently an Associate Professor with the Department of Computer Science, Naval Postgraduate School, Monterey, CA, USA. His research interests include intersection of computer architecture and computer security.

Dr. Huffmire's was the recipient of the Science, Mathematics, and Research for Transformation Scholarship (U.S. Department of Defense) and the Navy Meritorious Civilian Service Award (U.S. Navy).