Security intrusion detection using quantum machine learning techniques

Maxim Kalinin · Vasiliy Krundyshev

Received: 11 January 2022 / Accepted: 27 April 2022 / Published online: 24 June 2022
© The Author(s), under exclusive licence to Springer-Verlag France SAS, part of Springer Nature 2022

Abstract
Conventional machine learning approaches applied for the security intrusion detection degrades in case of big data input (10^6 and more samples in a dataset). Model training and computing by traditional machine learning executed on big data at a common computing environment may produce accurate outputs but take a long time, or produce poor accuracy by quick training, both disparate to malicious activity. The paper observes the quantum machine learning (QML) methods overcoming the barriers of big data and the computing abilities of common hardware for the purpose of high performance intrusion detection. Quantum support vector machine (QSVM) and quantum convolution neural network (QCNN) as concurrent methods are discussed and evaluated comparing to the conventional intrusion detectors running on the traditional computer. The QML-based intrusion detection utilizes our own dataset that implements the grouping of the network packets into the input streams eatable for the QML. We have developed the software solution that encodes the network traffic streams ready to the quantum computing. Experimental results show the ability of the QML-based intrusion detection for processing big data inputs with high accuracy (98%) providing a twice faster speed comparing to the conventional machine learning algorithms utilized for the same task.

Keywords Big data · Intrusion detection · Stream dataset · Quantum machine learning

1 Introduction
Since the creation of the first computer network by the US Advanced Research Projects Agency, the network data volume has been permanently increasing. By the Cisco’s report, the number of the connected stationary and mobile devices will amount around 28.5 billion (avg. 4 devices per a human), and more than a half of all connections will be arranged between the mobile devices, smart sensors, Internet of Things (IoT) components [1]. The researchers of Nokia Bell Labs believe that in 2022 the total traffic volume is reaching 330 exabytes per month [2]. At the same time, the COVID-19 forces the digital world to stay alive even after suppressing the pandemic. In digital era, the information security, data protection, and cybersecurity are even more acute than before. At new network infrastructures, a large number of the connected mobile hosts has weak built-in security capabilities and thus can be compromised by the intruders and induced to a malicious activity [3,4]. The remote access to hosts, the broadband Internet, the IoT and machine-to-machine networks, the next generation infrastructures (e.g., smart factories and smart cities) have led us to huge volumes of data defining the network features, specifically, the network security. To protect the network environment, the intrusion detection systems (IDS) based on the signature or anomaly analysis have been successfully utilizing against the security intrusions and malicious activity for a long time. However, in case of huge data inputs, traditional machine learning (ML) methods meet several barriers:

- a wide variety of possible intrusions forces us to create and handle extremely big databases of the malicious samples. Any ML-based classification done on a big data input significantly degrades the IDS’s performance on the training and testing stages [5,6];
- new intrusions specific to smart networks (e.g., the forced power consumption, the dynamic topology re-building)
have a wide class of polymorphic attacks that have mutations, namely, local differences, omissions, blank spaces in the operational sequences. New attacks dynamically adjust to changing work conditions and protection environments, and thus can avoid the ML-based detectors [7]. For new attacks, hybrid ML models are built and big databases of the signatures are utilized to inspect all possible variations of attacks.

To overcome the mentioned obstacles, the security researchers and IDS developers are focused on the advanced ML methods for the intrusion detection (e.g., [8–11]). For instance, the extra-deep neural networks with billions of weights running on the supercomputer are able to identify complex and subtle patterns at a big dataset of attack signs. The major goal of such works is to reduce the dependence of the ML methods on the long search at large-size databases of the malicious signatures, enrich an ability of the ML models to self-training at the attack identification, identify unknown and polymorphic intrusions, and attain the higher performance of the intrusion detection.

In case of big data inputs (that contain more than 10^6 samples in datasets), even the advanced ML-based detectors degrade in quality and performance [12]: model training and application using a traditional ML detector executed on a big data input on a common computing device may produce accurate outputs but takes an extremely long time, or it can be quickly trained and calculated but it performs a poor accuracy of outputs; both ways disparate to malicious activity characteristics such as intrusion variability, intensity, adaptability. In this work we focus on the novel approach of the quantum machine learning applied to the intrusion detection, specifically, in case of big inputs.

Quantum machine learning (QML) is an integration of the quantum computing with the ML solutions. The most common use case of the QML refers to the ML algorithms for analysis of classical data executed on a quantum computer, i.e. the quantum-enhanced ML. While the ML algorithms are used to compute immense quantities of data, the QML utilizes the quantum system to improve computational speed and data processing done by the algorithms, where computationally difficult routines are outsourced to the quantum computer. Instantaneous processing of huge datasets and solving the optimization task make the quantum computing technology one of the most promising tools in the ML area [13]. Processing a huge dataset input, the quantum computer offers quadratic and even exponential acceleration against the conventional ML as proposed in [14–16]. This advantage of the quantum computing has been already applied to various fields of information security. For instance, the quantum distribution of cryptographic keys successfully solves the problem of guaranteed distribution of the keys between the users. In 2014, the QML was applied to handwriting recognition.

Concerning the intrusion detection, there are several works introducing the application of the QML to detect concrete kinds of cyber attacks. For example, [17] reviews a quantum support vector machine (QSVM) constructed to detect DDoS attacks. [18] presents the experimental study of the QSVM running with conventional NSL-KDD and NB-15 datasets, that denotes the QSVM’s readiness to the intrusion detection on common attack datasets.

The aim of our work is to investigate and compare the basic QML models and understand their superiority while processing a big data input. And the relative task is the transforming of the data input in to a different representation form shorting a retardation of the QML’s training phase. Respecting these tasks, the next paper is organized as follows: Section 2 reviews the basic principles of the quantum computing reasonable for the high performance intrusion detection; Section 3 observes two basic QML methods, the quantum support vector machine (QSVM) and the quantum convolutional neural network (QCNN), applied to detect the malicious activity; Section 4 provides a survey of the QML frameworks supporting the QML-based intrusion detection; Section 5 discusses our stream dataset developed for training and testing the QML-based intrusion detectors; Section 6 presents our input encoding method ready to be processed on a quantum device; Section 7 shows the effectiveness evaluation of the implemented QML classifiers; and, finally, Section 8 concludes our work.

2 The quantum machine learning

Today, the QML becomes a new trend in quantum computing utilization combining the conventional ML models and the quantum processing at one solution, in which the computationally complex routines are executed on a quantum device [19,20]. Classical computer works with the bit registers of physical systems, which can be in one of two mutually exclusive states: "0" and "1". In case of optical circuit, two mutually orthogonal polarizations of photons denote the states of information. In quantum physics, a bit can be stated as "0" and "1" with probabilities different from zero and one. The state of a bit can be described using a quantum state vector:

\[ |\psi\rangle = c_0 |0\rangle + c_1 |1\rangle; |0\rangle, |1\rangle \in \mathcal{H}; c_0, c_1 \in \mathbb{C}; |c_0|^2 + |c_1|^2 = 1, \]

where the Hilbert space \( \mathcal{H} \) is a complex linear space with scalar product \( \langle | \rangle \). The probability of detecting a bit in one of two orthogonal quantum states, \( \langle 0 | 0 \rangle = 1 = \langle 1 | 1 \rangle, \langle 0 | 1 \rangle = 0 \) is determined by the squared moduli of the state amplitudes:
A bit whose evolution obeys the Schrödinger equation
\[ i \frac{\partial}{\partial t} |\psi\rangle = \hat{H} |\psi\rangle \]
is called a quantum bit, or qubit.

In practice, along with the quantum nature of the qubit, one has to take into account its interaction with a fluctuating environment (thermostat) in terms of the density matrix, according to the methods of the theory of open quantum systems [21,22]. However, already the representation (1) made it possible to extend the known information theory to the theory of quantum information [23]. The main idea of the quantum information theory is that a large class of problems that require resources and time exponentially in size of input data for solving on a classical computer can be solved much faster, in polynomial time, if one uses a quantum computer [24].

The quantum computer, just like a classical computer, can be built on the principle of a Turing machine and perform any prescribed sequence of operations with register bits, classical or quantum [25]. If the Boolean algebra of operations is used, then the algorithm remains classical, regardless of what bits are used in this case, classical or quantum. In this case, the operations performed on the device are not reversible in sense of quantum mechanics, and time and resources required to execute the algorithm are approximately have the same order for quantum and classical computers. It is possible to obtain superiority in computational efficiency over classical algorithms if, instead of classical Boolean operations, quantum computing is applied. The unitary evolution of a quantum register (if there is no interaction with the environment) is reversible in time, and all possible quantum states of the register can be processed in parallel if special devices, the quantum gates, are used. Therefore, any quantum algorithm is one that requires a quantum computer to execute.

The most famous quantum algorithms we know today are the decomposition of a number with the quantum Fourier transform (the Shor’s algorithm [26]) and the unordered database search (the Grover’s algorithm [27]). These algorithms have opened a new abilities to utilization of a quantum computer. The QML methods generalize the ML methods in such a way that quantum advantages can be used [28,29].

As a computing paradigm, the QML summarizes several data-driven methods:

1. Linear quantum classifiers based on the support vector machine (QSVM) [30].
2. Quantum neural networks, including quantum convolutional neural network (QCNN) [31] and quantum direct-forwarded neural networks (quantum perceptron) [32–34].
3. Hopfield-type networks [35].
4. Quantum Boltzmann machines [36].
5. Weightless quantum network [37].
6. Quantum walk on a graph [38].

Among the above listed methods, QSVM is an effective computing model in case of working with a large amount of data – it does not tend to overfit, provides high accuracy when working with a big feature map, and allows use of the kernel trick [39,40]. QCNN has exponential storage capacity, resistance to noise in input data and high performance [41]. The intrusion detection as a security-relative task requires quick and accurate detectors, therefore QSVM and QCNN have been selected for the further analysis.

3 The quantum machine learning models

3.1 The quantum support vector machine (QSVM)

One of the fundamental supervised ML methods designed to classify objects into two classes \{-1, +1\} is a support vector machine (SVM). The SVM’s work task is to classify the presented arbitrary \(N\)-dimensional vectors based on the available training sample of \(M\) reference vectors:

\[
\left\{ (x_j, y_j) : x_j \in \mathbb{R}^N, \ y_j = \pm 1 \right\}_{j=1}^M.
\]  (3)

To perform such a classification, an optimal hyperplane is found, defined by the vector \(w\) in the data space, which divides \(\mathbb{R}^N\) into two required classes: \(wx_j + b \geq +1\) for class \(+1\), and \(wx_j + b \leq -1\) for class \(-1\), where \(b/|w|\) is displacement of the hyperplane.

To find the parameters \(w \) and \(b\) that define the hyperplane, classical linear algebra algorithms need \(O(\log(\frac{1}{\varepsilon}) \text{poly} (N,M))\) steps, with accuracy \(\varepsilon\) [42]. In this case, finding the separating hyperplane is reduced to minimizing the norm \(|w|^2/2\) with the constraint \(y_j(wx_j + b) \geq 1\).

To build a separating hyperplane, it is convenient to use the correlation matrix of vectors of training sample \(J_{ij} = x_i x_j\). Having this matrix defined, the solution of maximization problem for the Karush-Kuhn-Tucker function

\[
L(\alpha) = \sum_{j=1}^{M} y_j \alpha_j - \frac{1}{2} \sum_{j,k=1}^{M} \alpha_j J_{jk} \alpha_k
\]  (4)

subject to the constraints \(\sum_{j=1}^{M} \alpha_j = 0\) and \(y_j \alpha_j \geq 0\). Then we can put \(w = \sum_{j=1}^{M} \alpha_j x_j\) and \(b = y_j - wx_j\) for those \(j\) for which \(\alpha_j \neq 0\). The vectors \(x_j\) for which \(\alpha_j \neq 0\) are called support vectors. Since computing the dot product requires \(O(N)\) steps, executing the entire algorithm with accuracy \(\varepsilon\) requires \(O(\log(\frac{1}{\varepsilon}) M^2 \times (M+N))\) steps.

In a quantum version, to construct correlation matrices, we can introduce an additional register corresponding to the
works [45]. Recently, optical quantum neural networks have been proposed, where quantum neural networks are identical to optical neural networks [44]. In this sense, excluding nonlinear effects, the visual IBM Circuit Composer is plotted in Figure 1.

The inference model is associated with the generalization of the error propagation method used for training a conventional perceptron [44]. In this sense, excluding nonlinear effects, quantum neural networks are identical to optical neural networks [45]. Recently, optical quantum neural networks have a progress [46,47]. In the known models of the quantum neural networks, the procedure for setting up a neural network consists of a sequence of classical operations of measuring and preparing states. The question of whether the network configuring can be done at a purely quantum level is related to the role of consciousness in quantum mechanics and is still open [48].

Convolutional neural network (CNN) is a success case of quantum neural networks that has demonstrated its quantum advantage in recognition of complex objects (e.g. [49,50]). When it comes to image processing, a CNN generally consists of a sequence of different (interleaved) layers of image processing; in each layer, an intermediate two-dimensional array of pixels, a feature map, is produced from the previous one [31]. A CNN consists of different types of layers: convolutional layers, subsampling (or pooling) layers and layers of a regular perceptron. The first two layers (convolutional and subsampling) alternating with each other, form an input feature vector for a multilayered perceptron. Averagely, the detection accuracy of the CNN exceeds the same one of the conventional neural networks by 10–15%. The CNN can learn quickly by purely parallelizing the convolution process for each map, as well as reverse convolution when the error propagates over the network.

To specify the mathematical model of the considered neural network, we apply the following notation.

Thus, theoretically, the training process of a binary classifier can be exponentially accelerated if the described classification algorithm is run on a quantum computer [29]. This fact has inspired us to adapt the SVM-based intrusion detector to run it on a quantum device and test it working with big datasets. Our implementation of the QSVM algorithm in the visual IBM Circuit Composer is plotted in Figure 1.

3.2 The quantum convolutional neural network (QCNN)

First artificial quantum neural network has been proposed in [43]. This research presented replacing classical signals arriving at the input of neurons with quantum states with amplitude and phase. A quantum state is also formed at the output of the neuron, which depends on the linear superposition of the incoming states. The weights are complex numbers (which change as the network is trained), so that each input quantum state is not only weighted in amplitude, but also phase shifted. The use of the signal phase in a neural network model is associated with the generalization of the error back-propagation method used for training a conventional perceptron [44]. In this sense, excluding nonlinear effects, quantum neural networks are identical to optical neural networks [45]. Recently, optical quantum neural networks have been proposed in [43].

where $|x_i\rangle = \frac{1}{\sqrt{N}} \sum_{k=1}^{N} (x_k)_{k} |k\rangle$. Correlation matrix becomes a trace of the projection operator $P_X = |\chi\rangle \langle \chi|$ on the second argument:

$$Tr P_X = N_X^{-1} \sum_{i,j=1}^{M} |x_j \langle x_i | i \rangle j \langle j | = \frac{J}{Tr J},$$

where $N_X = \sum_{i=1}^{M} |x_i|^2$, and, respecting to the quantum parallelism, can be calculated on a network quantum computer in $O(\log NM)$ steps.

Thus, theoretically, the training process of a binary classifier can be exponentially accelerated if the described classification algorithm is run on a quantum computer [29]. This fact has inspired us to adapt the SVM-based intrusion detector to run it on a quantum device and test it working with big datasets. Our implementation of the QSVM algorithm in the visual IBM Circuit Composer is plotted in Figure 1.

$$y_n^l = f_l \left( \sum_{m \in V_n^l} y_{m}^{l-1} \otimes w_{m,n}^l + b_n^l \right),$$

where $w_{m,n}^l = w_{m,n}(l,j)$ is convolution applied to feature map $m$ of layer $(l-1)$, on layer $l$ with feature map $n$; $b_n^l$ is threshold values attached to the feature map $n$ on layer $l$; $V_n^l$ is a list of all layer levels $(l-1)$ that connect to feature map $n$ on layer $l$; and operator $\otimes$ is a mathematical operation of two-dimensional convolution.

Suppose that the size of the input feature map $y_m^{l-1}$ is equal to $H^{(l-1)} \times W^{(l-1)}$, and the size of the $w_{m,n}^l$ convolution applied to them equals $r^l \times c^l$, then the size of the output feature map $y_n^l$ is calculated as:

$$\left(H^{l-1} - r^l + 1\right) \times \left(W^{l-1} - c^l + 1\right).$$
The sub-sampling layer \( l \) has an even number, \( l = 2, 4, \ldots, 2a \). For the feature map \( n \), we use the following notation: \( w_{\cdot n, m}^{l} \) is the weight applied to \( n \) on layer \( l \), \( b_{n}^{l} \) is the bias term. We divide the feature map \( n \) of the \((l - 1)\)-th layer into non-overlapping blocks of \( 2 \times 2 \) pixels. Then we sum up the values of the four pixels in each block and, as a result, we get the matrix \( z_{n}^{(l-1)} = z_{n}^{(l-1)}(i, j) \), the elements of which are the corresponding values of the sums. Therefore, the formula for calculating the matrix’s values is next:

\[
z_{n}^{l-1} = y_{n}^{l-1}(2i - 1, 2j - 1) + y_{n}^{l-1}(2i - 1, 2j) \\
+ y_{n}^{l-1}(2i, 2j - 1) + y_{n}^{l-1}(2i, 2j).
\]

The feature map \( n \) of sub-sampling layer \( l \) is calculated as:

\[
y_{n}^{l} = f_{l}(z_{n}^{l-1} \times w_{\cdot n, m}^{l} + b_{n}^{l}).
\]

The feature map \( y_{n}^{l} \) in sub-sampling layer \( l \) has the size \( H^{l} \times W^{l} \), where \( H^{l} = \frac{H^{l-1}}{2} \), \( W^{l} = \frac{W^{l-1}}{2} \).

The output layer \( L \) consists of sigmoidal neurons. Let denote \( w_{\cdot n, m}^{L} \) a weight, applied to the feature map \( m \) of the last convolutional layer, to neuron \( n \) of the output layer. \( b_{n}^{L} \) points to a bias term associated with the neuron \( n \) of the layer \( L \). This way we obtain the formula for the output of the neuron \( n \):

\[
y_{n}^{L} = f_{L} \left( \sum_{m=1}^{N_{L}} y_{m}^{L-1} w_{m, n}^{L} + b_{n}^{L} \right),
\]

where \( N_{L} \) is a number of output sigmoidal neurons on output layer \( L \).

Therefore, the output of the CNN is a vector:

\[
y = \left[ y_{1}^{L}, y_{2}^{L}, \ldots, y_{N_{L}}^{L} \right].
\]

Concerning the quantum implementation of the CNN (QCNN), an input to a QCNN is an (unknown) quantum state \( \psi_{\alpha} \). A convolution layer applies a single quasi-local unitary \((U_{i})\) in a translationally invariant manner for finite depth. For sub-sampling, a fraction of qubits are measured, and their outcomes determine unitary rotations \((V_{j})\) applied to nearby qubits. Hence, non-linearities in the QCNN arise from reducing the number of degrees of freedom. The convolution and sub-sampling layers are performed until the system size is sufficiently small; then, a fully connected layer is applied as a unitary \( F \) on the remaining qubits. Finally, the outcome of the circuit is obtained by measuring a fixed number of output qubits. As in a classical CNN, the QCNN hyperparameters such as the number of convolution and subsampling layers are fixed, and the unitaries themselves are learned [31].

To classify \( N \)-qubit input states, a QCNN is thus specified by \( O(\log(N)) \) parameters. This corresponds to a double exponential reduction comparing to a generic quantum circuit-based classifier and promises efficient training and implementation. For example, given a set of \( M \) classified training vectors \( \{ (|\psi_{\alpha}\rangle, y_{\alpha}) : \alpha = 1, \ldots, M \} \), where \( |\psi_{\alpha}\rangle \) are input states and \( y_{\alpha} = 0 \) or \( 1 \) are corresponding binary classification outputs, one could compute the mean squared error (MSE):

\[
MSE = \frac{1}{2M} \sum_{\alpha=1}^{M} (y_{\alpha} - f(U_{i}, V_{j}, F)(|\psi_{\alpha}\rangle))^{2},
\]

where \( f(U_{i}, V_{j}, F)(|\psi_{\alpha}\rangle) \) denotes the expected QCNN’s output value for input \( |\psi_{\alpha}\rangle \). The training consists of initializing all unitaries and successively optimizing them until convergence, e.g. by a gradient descent.

Effectiveness of the QCNN has also inspired us to implement the QCNN-based intrusion detector to run it on a quantum device and test it working with big datasets. Figure 2 presents a general QCNN’s circuit model.

4 Platforms and software frameworks supporting the QML

The quantum devices D-Wave [52] and IBM Q [53] are the most developed quantum platforms that can support the QML:

- D-Wave has the status of an “analog quantum computer”, as it is able to solve only a narrow range of the quantum annealing tasks, but, at the same time, they declare its capacity about 2,000 qubits;
- IBM Q is a general purpose quantum computer on which arbitrary quantum algorithms can be run. At this moment, systems with 20 qubits (for commercial use), and open systems IBM Q Experience with 16 and 5 qubits are provided.
The development kits supporting the quantum computing are observed and compared in Table 1. Table 2 summarizes the software frameworks that support the QML. The most promising kit is Qiskit, which allows us to manage the resources and adapt the developed ML applications for the specific quantum computers. In addition, this framework contains the pre-implemented QML models, e.g. QSVM, VQC, QGAN. Tensorflow Quantum library was selected for its main advantages: flexibility, ready-to-use ML models, box application packages, scalability in hardware and software, large online community, and compatibility with Keras library.

### 5 Synthesis of a stream dataset for the quantum classifiers

Existing network datasets, such as IEEEDataPort’s IoT Network Intrusion Dataset \[54\], Stratosphere Lab’s Malware on IoT Dataset \[55\] and Canberra’s BoT-IoT Dataset \[56\], contain the network packets of two classes: "with attack" and "without attack". However, in practice, the approach of classifying the individual packages has a few significant drawbacks:

- many packets from the available datasets are not malicious in terms of their content. For example, Denial of Service (DoS) packets do not contain malicious signatures, and the classifier identifies such packets as malicious by a sender’s IP address;
- there is a class of attacks that can spread the payload across different packets;
- many attacks are carried out in stages, and an IDS based on a packet classification is ineffective in this case. Modern IDSs, like Snort and Suricata, use the packet bundling technique to analyze a network traffic.

To bypass these issues, we introduce the converting a dataset containing attack types and normal traffic into a stream dataset. The fields in the stream are the transformed fields of the packets entering that stream. For example, in IoT Network Intrusion Dataset, packet fields like tcp.srcport, tcp.dstport, udp.srcport, udp.dstport, tcp.checksum.status, udp.checksum.status are merged in to srcport, dstport, ip.checksum.status to get rid of the Layer 4 protocol dependency. The tcp.stream and udp.stream fields are combined in to the stream field, which is used to group packets to the streams.

---

**Table 1** Development kits supporting the QML

| Kit                      | QDK          | Qiskit                  | ProjectQ                | Forest       |
|--------------------------|--------------|-------------------------|-------------------------|--------------|
| Company, year            | Microsoft, 2018 | IBM, 2017               | ETH Zurich, 2017        | Rigetti, 2017|
| Open source              | +            | +                       | +                       | +            |
| Supported OS             | Windows, Unix, Mac | Windows, Unix, Mac     | Windows, Unix, Mac      | Windows, Unix, Mac |
| Prog. language           | Q#           | Qiskit                  | ProjectQ               | pyQuil       |
| Quantum language         | No           | QASM                    | No                      | Quil         |
| Quantum platform         | No           | IBMQX (5 qubits, 16 qubits), QS1 (20 qubits)) | Can be connected to IBM backend | No |
| Qubits in simulator      | 30 locally, 40 in cloud | 25 locally, 30 in cloud | 28 locally | 20 locally, 30 in private access |
| Specifics                | Built-in algorithms and samples | QASM code generation, topology-specific compiler, Slack community channel, schematic dashboard, Aqua library | Diagramming, connection to IBM server modules, availability of multiple library plugins | Quil code generation, sample algorithms in Grove, compiler for specific topologies, Slack community channel |

**Table 2** QML frameworks

| Framework                  | Tensorflow Quantum | PyTorch | Cirq | Strawberry Fields | PennyLane |
|----------------------------|---------------------|---------|------|-------------------|-----------|
| Availability of examples and specs | High                | Low     | Low  | Med               | Med       |
| Neural network modeling    | High                | High    | Med  | Med               | Med       |
| Friendliness and modular interface | High                | Med     | Low  | Med               | Low       |
| Performance                | High                | High    | Med  | Med               | Med       |
| Multiple GPU support       | Med                 | Med     | Low  | Med               | Low       |
| Specifics                  | High speed          | Academic use | Study ML | Image processing | High speed |
Table 3  Stream fields

| Field name                                      | Description                                |
|------------------------------------------------|--------------------------------------------|
| ip.flags.rb.mean; ip.flags.df.mean; ip.flags mf.mean; tcp.flags.res.mean; tcp.flags.ns.mean; tcp.flags.ecn.mean; tcp.flags.urc.mean; tcp.flags.ack.mean; tcp.flags.push.mean; tcp.flags.reset.mean; tcp.flags.syn.mean; tcp.flags.fin.mean | Average flag values                        |
| frame.len.std; frame.len.min; frame.len.max; frame.len.mean | Minimum, maximum, mean values of packet length and standard deviation |
| frame.len.rate | Bandwidth bytes per sec |
| payload.std.mean; payload.min.mean; payload.max.mean; payload.mean.mean | Minimum, maximum, mean of packet data bytes and standard deviation |
| payload.print.mean | Average number of characters printed in packet data |
| srcport.std | Source port standard deviation |
| dstport.std | Destination port standard deviation |
| ip.ttl.std; ip.ttl.min; ip.ttl.max; ip.ttl.mean | Minimum, maximum, average packet lifetime and standard deviation |
| tcp.seq_raw.std; tcp.seq_raw.min; tcp.seq_raw.max; tcp.seq_raw.mean | Minimum, maximum, mean of packet sequence number and standard deviation |
| tcp.ack_raw.std; tcp.ack_raw.min; tcp.ack_raw.max; tcp.ack_raw.mean | Minimum, maximum, average batch confirmation number and standard deviation |
| tcp.window_size_value.std; tcp.window_size_value.min; tcp.window_size_value.max; tcp.window_size_value.mean | Minimum, maximum, average packet window size and standard deviation |
| int.std; int.min; int.max; int.mean | Minimum, maximum, mean interval between bursts and standard deviation |
| count | Number of packages |
| duration | Flow duration |
| prate | Packets per second |

For the fields ip.ttl, tcp.seq_raw, tcp.ack_raw, and tcp.window_size_value, the average, minimum and maximum values, and the standard deviation are calculated. The intervals between packets in the stream are analyzed separately, what makes it possible to judge the frequency of the sending packets. For example, an attack stream “Flooding” with equal intervals between the packets will give a low average deviation in the size of the intervals. Table 3 provides a fragment of the stream’s fields.

For large-scale experiments, additional traffic was generated using the nmap and hping programs; packet capture was carried out using the Wireshark program. The console version of the Wireshark, tshark, program receives a .pcap file and a set of parameters as input, then text data is read from the standard output stream and written in CSV format. As a result, the stream dataset has been developed containing over 10 million records, consisting of 58 fields.

6 Data coding for the QML

To train a QML model on real data, one needs to be able to translate data from a bit representation to a qubit one. For this purpose, the encoder has been developed. Firstly, using the Cirq library, a qubit is created, which is placed in the circuit:

```python
>>> qubit = cirq.GridQubit (1,1)
```

Next, the IP address is converted into the number, on the basis of which the angle of rotation is set up, then the Pauli gate is added to the circuit. The rotation angle is the value into which the IP address is converted:

```python
>>> C = cirq.Circuit ()
>>> a = make_num_from_ip (i [0])
>>> a = make_angle (a)
>>> C.append (cirq.rx (a) (qubit))
```
The algorithm then starts working with the fields that contain numeric values. There are 58 fields in each record. For each, a rotation angle is generated, and the next Pauli gate is added to the circuit:

```python
>>> for j in range (1.58):
>>> angle = make_angle (i [j])
>>> C.append (cirq.rx (angle) (qubit))
```

Therefore, each stream field is converted to a value from 0 to $\pi$, after that the qubit is created. Then Pauli gates are applied to the qubit. The value obtained in the previous step is utilized as a rotation angle. Then, the resulting quantum scheme is added to the list, which serves for the training of the classifier.

Figure 3 shows a quantum diagram of the encoding algorithm. An example of a quantum circuit is shown in Figure 4.

---

### 7 The experimental study

For the experiments with the QML-based intrusion detection, the test bench has been constructed:

- Ubuntu operating system provides interoperability between the hardware and software;
- Python 3.7 programming language sets links the program and the operating system;
- Cirq library executes the quantum circuits;
- NVidia Cuda provides the faster emulation for quantum circuits;
- Tensorflow provides the platform for building ML method;
- Tensorflow Quantum contains the framework structures for the QML such as qubits, gates, schematics, and measurement operators.

Figures 5 and 6 show the results of the conducted experiments when solving the problem of classification on a big data input ($>10^6$ records) using the traditional SVM and its quantum implementation – QSVM. When using the conventional SVM, only HTTP Flooding and Port Scanning attacks are detected with high accuracy. The accuracy of most of the remaining classes is in the range from 0.4 to 0.8. The network streams containing the ACK Flooding attack were practically undefined. Using QSVM, the classification was done with accuracy of 98%.

Figures 7 and 8 show the results of the experiments with the conventional CNN and its quantum implementation – QCNN. The neural networks also demonstrates the superiority of the quantum method over the conventional one. According to the confusion matrix, the detection accuracy of normal packets is, like in case with the QSVM, 98%. Looking at the ROC curves, it can be noted that QCNN performs better than QSVM.

Table 4 compares the training time of the QML implementations to the QML algorithms varying volumes of input datasets. QSVM and QCNN can be trained approximately twice faster on a big input, and this priority keeps growing if the input data volume enlarges more. The priority of the QML classifier is also achieved on the same datasets using the QML on the Tensorflow Quantum framework. The results of the experiments have demonstrated abilities of the QML against the conventional ML-based detectors when classifying the big volumes of input data.
Security intrusion detection using quantum machine learning techniques

8 Conclusion

As the result of our research, the possibility of using the QML methods to solve the problem of analyzing big volumes of input data was considered. Analysis of platforms and frameworks for implementing the quantum computing has shown that the most promising are Qiskit platform and Tensorflow Quantum framework.

The stream dataset has been developed to improve the QML processing; it contains 58 parameters such as average flag values, minimal, maximal, mean values of packet length and standard deviation, bandwidth bytes/sec, number of packages, flow duration, minimal, maximal, mean interval between bursts and standard deviation, packets per second Layer 4 protocol, stream category. For the quantum processing, the encoding method has been implemented to transform the bit representation of the network streams into qubits.

When we have a large-scale network with a big volume of security-relevant data, the QML-based intrusion detection makes the protection more efficient than a traditional ML approach. Comparison of the QML detectors built on the QSVM and QCNN classifiers against the conventional SVM and QCNN detectors has shown the promise of the quantum apparatus on big data inputs. The QML-based methods have
Fig. 7  Intrusion detection results with the conventional CNN

Fig. 8  Intrusion detection results with the QCNN

**Table 4**  QML and conventional ML training time

| Input size (samples) | SVM training (hrs) | QSVM training (hrs) | CNN training (hrs) | QCNN training (hrs) |
|----------------------|--------------------|---------------------|--------------------|--------------------|
| 100,000              | 0.5                | 0.4                 | 0.7                | 0.5                |
| 200,000              | 1.4                | 0.8                 | 1.8                | 0.9                |
| 300,000              | 2.2                | 1.3                 | 2.5                | 1.4                |
| 400,000              | 3.1                | 1.7                 | 3.3                | 1.9                |
| 500,000              | 4.4                | 2.3                 | 4.9                | 2.7                |
| 600,000              | 5.9                | 3                   | 6.9                | 3.6                |
| 700,000              | 6.8                | 3.3                 | 7.7                | 3.9                |
| 800,000              | 7.7                | 3.8                 | 8.5                | 4.8                |
| 900,000              | 8.9                | 4.5                 | 9.3                | 4.9                |
| 1,000,000            | 9.6                | 5.2                 | 10.9               | 5.6                |
surpassed the ML-based implementations both in accuracy and performance. Comparison of the conventional ML classifiers and the QML classifiers on huge stream datasets has shown significant superiority of the quantum approach (e.g., QSVM and QCNN classification accuracy is 98%). Due to the QML, the training time has been reduced by more than twice.

Comparing the QSVM and QCNN detectors, we have concluded that the QCNN is more promising, despite the fact that the QSVM is faster. The QCNN makes it possible to select the most significant features with a higher probability, which reduces the complexity of the method. Another advantage of the QCNN is its ability to vary the number of layers and the size of the cores to solve the classification task.

Our further work is targeted at optimization of the quantum algorithms and parallelization schemes for the fast training of the QML models.

Acknowledgements Project results are achieved using the resources of supercomputer center of Peter the Great St.Petersburg Polytechnic University - SCC “Polytechnichesky” (http://www.spbstu.ru). The research is funded by the Ministry of Science and Higher Education of the Russian Federation under the strategic academic leadership program “Priority 2020” (Agreement 075-15-2021-1333 dated 30 November 2021).

References

1. Cisco: Cisco Annual Internet Report (2018–2023) White Paper. www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html (2021)

2. Nokia: Nokia catalyzes the next chapter of the internet; innovations deliver massive performance and heightened security needed for cloud and machine era. www.nokia.com/about-us/news/releases/2017/06/14/nokia-catalyzes-the-next-chapter-of-the-internet-innovations-deliver-massive-performance-and-heightened-security-needed-for-cloud-and-machine-era (2021)

3. Kolias, C., Kambourakis, G., Stavrou, A., Voas, J.: Ddos in the iot: Mirai and other botnets. Computer 50(7), 80–84 (2017)

4. Zegzhda, D.P., Stepanova, T.V.: Efficiency estimation of the protection software for neutralization and elimination of botnets. Nonlinear Phenomena Complex Syst. 17(3), 290–295 (2014)

5. Lavrova, D., Poltavtseva, M., Shyrkina, A.: Security analysis of cyber-physical systems network infrastructure. In: Proceedings of 2018 IEEE Industrial Cyber-Physical Systems, pp. 818–823 (2018)

6. Pan, S., Morris, T., Adhikari, U.: Developing a hybrid intrusion detection system using data mining for power systems. IEEE Trans. Smart Grid 6, 3104–3113 (2015)

7. Kalinin, M., Krundyshev, V., Sinyapkin, B.: Development of the intrusion detection system for the internet of things based on a sequence alignment algorithm. Autom. Control. Comput. Sci. 54(8), 991–998 (2020)

8. Berger, I., Rieke, R., Kolomeets, M., Chechulin, A., Kotenko, I.: Comparative study of machine learning methods for in-vehicle intrusion detection. In: Lecture Notes in Computer Science (including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 11387 LNCS, pp. 85–101 (2019)

9. Yin, C., Zhu, Y., Fei, J., He, X.: A deep learning approach for intrusion detection using recurrent neural networks. IEEE Access 5, 21954–21961 (2017)

10. Lavrova, D., Semyanov, P., Shyrkina, A., Zegzhda, P.: Wavelet-analysis of Network traffic time-series for detection of attacks on digital production infrastructure. In: SHS Web Conferences, vol. 44 (2018)

11. Malyshev, E., Moskvin, D., Zegzhda, D.: Application of an artificial neural network for detection of attacks in vanets. Autom. Control. Comput. Sci. 53(8), 889–894 (2019)

12. Kalinin, M., Zegzhda, P.: AI-based Security for the Smart Networks. (2020)

13. Zhao, L., Xiaomei, L., Nanyang, X., Jiangfeng, D.: Experimental realisation of quantum artificial intelligence. (2014). arXiv:1410.1054

14. Schuld, M., Sinayskiy, I., Petruccione, F.: An introduction to quantum machine learning. Contemp. Phys. 56(2), 172–185 (2015)

15. Adcock, J., Allen, E., Day, M., Frick, S., Hinchliff, J., Johnson, M., Morley-Short, S., Pallister, S., Price, A., Stanisic, S.: Advances in quantum machine learning. Science (2015)

16. Bisio, A., Dariano, G.M., Perinotti, P., Sedlák, M.: Quantum learning algorithms for quantum measurements. Phys. Lett. Sect. A Gen. Atomic Solid State Phys. 375(39), 3425–3434 (2011)

17. Payares, E., Martinez-Santos, J.C.: Quantum machine learning for intrusion detection of distributed denial of service attacks: a comparative overview. In: Proceedings of SPIE Quantum Computing, Communication, and Simulation, p. 47 (2021)

18. Gouveia, A., Correia, M.: Towards quantum-enhanced machine learning for network intrusion detection. In: 2020 IEEE 19th International Symposium on Network Computing and Applications, pp. 1–8 (2020)

19. Biamonte, J., Wittek, P., Pancotti, N., Rebentrost, P., Wiebe, N., Lloyd, S.: Quantum machine learning. Nature 549(7671), 195–202 (2017)

20. Wiebe, N., Braun, D., Lloyd, S.: Quantum algorithm for data fitting. Phys. Rev. Lett. 109(5), 050505 (2012)

21. Breuer, H.P., Petruccione, F.: The Theory of Open Quantum Systems, pp. 1–656 (2007)

22. Feynman, R.P., Vernon, F.L.: The theory of a general quantum system interacting with a linear dissipative system. Ann. Phys. 24(1), 118–173 (1963)

23. Vogel, M.: Quantum computation and quantum information, by M.A. Nielsen and I.L. Chuang. Contemp. Phys. 52(6), 604–605 (2011)

24. Simon, D.R.: On the power of quantum computation. SIAM J. Comput. 26(5), 1474–1483 (1997)
25. Deutsch, D.: Quantum Theory, the Church-Turing Principle and the Universal Quantum Computer. vol. 400, pp. 97–117 (1985)

26. Shor, P.W.: Algorithms for quantum computation: discrete logarithms and factoring. In: Proceedings 35th Annual Symposium on Foundations of Computer Science, pp. 124–134 (2002)

27. Grover, L.K.: Quantum mechanics helps in searching for a needle in a haystack. Phys. Rev. Lett. 79(2), 325–328 (1997)

28. Ezov, A.A., Ventura, D.: Quantum neural networks, pp. 213–235 (2000)

29. Nawaz, S.J., Sharma, S.K., Patwary, M.N., Asaduzzaman, M.: Quantum machine learning for 6g communication networks: state-of-the-art and vision for the future. IEEE Access 7, 46317–46350 (2019)

30. Rebentrost, P., Mohseni, M., Lloyd, S.: Quantum support vector machine for big data classification. Phys. Rev. Lett. 113(3), 130503 (2014)

31. Cong, I., Choi, S., Lukin, M.D.: Quantum convolutional neural networks. Nat. Phys. 15(12), 1273–1278 (2019)

32. Wan, K.H., Dahlsten, O., Kristjánsson, H., Gardner, R., Kim, M.S.: Quantum generalisation of feedforward neural networks. npj Quantum Inf. 3(1) (2017)

33. Zhou, R., Qin, L., Jiang, N.: Quantum Perceptron Network. vol. 4131 LNCS-I, pp. 651–657 (2006)

34. Wiebe, N., Kapoor, A., Svore, K.M.: Quantum Perceptron Models. In: Advances in Neural Information Processing Systems, pp. 4006–4014 (2016)

35. Johnson, M.W., Amin, M.H.S., Gildert, S., Lanting, T., Hamze, F., Dickson, N., Harris, R., Berkley, A.J., Johansson, J., Bunyk, P., Chapple, E.M., Enderud, C., Hilton, J.P., Karimi, K., Ladizinsky, E., Ladizinsky, N., Oh, T., Perminov, I., Rich, C., Thom, M.C., Tolkacheva, E., Truncik, C.J.S., Uchaikin, S., Wang, J., Wilson, B., Rose, G.: Quantum annealing with manufactured spins. Nature 473(7346), 194–198 (2011)

36. Amin, M.H., Andriyash, E., Rolfe, J., Kulchytskyy, B., Melko, R.: Quantum Boltzmann machine. Phys. Rev. X 8(2), 021050 (2018)

37. Da Silva, A.J., De Oliveira, W.R., Ludermir, T.B.: Classical and superposed learning for quantum weightless neural networks. Neurocomputing 75(1), 52–60 (2012)

38. Schuld, M., Sinayskiy, I., Petruccione, F.: Quantum walks on graphs representing the firing patterns of a quantum neural network. Phys. Rev. A Atomic Mol. Opt. Phys. 89(3), 032333 (2014)

39. Cortes, C., Vapnik, V.: Support-vector networks. Mach. Learn. 20(3), 273–297 (1995)

40. Havlicek, V., Corcoles, A.D., Temme, K., Harrow, A.W., Kandala, A., Chow, J.M., Gambetta, J.M.: Supervised learning with quantum-enhanced feature spaces. Nature 567(7747), 209–212 (2019)

41. Kerenidis, I., Landman, J., Prakash, A.: Quantum algorithms for deep convolutional neural networks. (2020). arXiv:1911.01117

42. Boyd, S., Vandenberghe, L.: Convex Optimization (2004)

43. Kak, S.: On quantum neural computing. Inf. Sci. 83(3–4), 143–160 (1995)

44. Nitta, T.: An extension of the back-propagation algorithm to complex numbers. Neural Netw. 10(8), 1391–1415 (1997)

45. Sharir, I., Friesem, A.A.: All-optical neural network with inhibitory neurons. Opt. Lett. 14(10), 485 (1989)

46. Inagaki, T., Inaba, K., Hamerly, R., Inoue, K., Yamamoto, Y., Takeo, H.: Large-scale ising spin network based on degenerate optical parametric oscillators. Nat. Photonics 10(6), 415–419 (2016)

47. McMahon, P.L., Marandi, A., Haribara, Y., Hamerly, R., Langrock, C., Tamate, S., Inagaki, T., Takesue, H., Utsunomiya, S., Aihara, K., Byer, R.L., Fejer, M.M., Mabuchi, H., Yamamoto, Y.: A fully programmable 100-spin coherent ising machine with all-to-all connections. Science 354(6312), 614–617 (2016)

48. Hagan, S., Hameroff, S.R., Teszynska, J.A.: Quantum computation in brain microtubules: decoherence and biological feasibility. Phys. Rev. E Stat. Phys. Plasmas Fluids Related Interdiscipl. Top. 65(6), 061901 (2002)

49. LeCun, Y., Bengio, Y.: Convolutional Networks for Images, Speech, and Time Series, pp. 276–279 (2003)

50. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. Commun. ACM 60(6), 84–90 (2017)

51. Phung, S.L., Bouzerdoum, A.: MATLAB Library for Convolutional Neural Networks. Technical report. http://documents.uow.edu.au/~phung/docs/cnn-matlab/cnn-matlab.pdf documents.uow.edu.au/~phung/docs/cnn-matlab/cnn-matlab.pdf

52. Ding, Y., Lamata, L., Martín-Guerrero, J., Lizaso, M.S., Chen, X., Orus, R., Solano, E., M., S.: Towards Prediction of Financial Crashes with a D-Wave Quantum Computer. (2019). arXiv:1904.05808

53. Santos, A.C.: The IBM Quantum Computer and the IBM Quantum Experience. (2016). arXiv:1610.06980

54. Kang, H., Ahn, D.H., G.M., L., Yoo, J.D., K.H., P., H.K., K.: IoT network intrusion dataset. http://ieeexplore.ieee.org/open-access/iot-network-intrusion-dataset (2019)

55. Garcia, S., Parmisano, A., Erquiaga, M.J.: IoT-23: A labeled dataset with malicious and benign IoT network traffic (Version 1.0.0). www.stratosphereips.org/datasets-iot23 (2020)

56. Koroniotis, N., Moustafa, N., Stinikova, E., Tummbull, B.: Towards the Development of Realistic Botnet Dataset in the Internet of Things for Network Forensic Analytics: Bot-IoT Dataset. (2018). arXiv:1811.00701

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.