ISSN 0146-4116, Automatic Control and Computer Sciences, 2021, Vol. 55, No. 8, pp. 1165–1174. © Allerton Press, Inc., 2021.
Russian Text © The Author(s), 2020, published in Problemy Informatsionnoi Bezopasnosti, Komp'yuternye Sistemy.

Analysis of a Huge Amount of Network Traffic Based on Quantum Machine Learning

M. O. Kalinin[a], * and V. M. Krundyshev[a], **

[a] *Peter the Great St. Petersburg Polytechnic University, St. Petersburg, 195251 Russia*

**e-mail: max@ibks.spbstu.ru*

***e-mail: vmk@ibks.spbstu.ru*

Received January 15, 2021; revised January 15, 2021; accepted February 10, 2021

Abstract—A method for analyzing network traffic based on quantum machine learning is presented. A method for network traffic encoding into quantum computer terms is developed. The experimental results show the superiority of the proposed approach over traditional machine learning methods for detecting network attacks.

Keywords: network traffic analysis, quantum computer, quantum machine learning, qubit, network attacks, intrusion detection system

DOI: 10.3103/S014641162108040X

Since 1969, when the first computer network was created by the U.S. Defense Advanced Research Projects Agency, a steady increase in the amount of network traffic has been observed. Throughout this history, the amount of Internet traffic has been increasing unevenly always depending on technological advance: availability of desktop computers, prevalence of broadband Internet, advent of laptops and smartphones, and improvement of wireless communications. The emergence of Internet of Things devices, digitalization of production, implementation of next generation 5G/5G+ wireless networks, and development of streaming services have resulted in extremely high amounts of traffic recorded by network equipment vendors. For example, in March 2020, Frankfurt DE-CIX, one of the busiest network hubs of the world, recorded the all-time traffic peak, over 9.1 Tbps [1].

Leading IT companies unanimously predict a further increase in the load on network infrastructure around the world. According to a forecast made by Cisco, global IP traffic is to increase more than threefold by 2022. The number of stationary and mobile devices will reach 28.5 billion (3.6 per person), and more than half of all connections will be made by smart devices, sensors, appliances, and so on [2]. Nokia Bell Labs analysts believe that in 2022, the total amount of traffic will reach the level of 330 EB per month [3].

At the same time, the obvious impact of the COVID-19 pandemic on people's minds is worth mentioning. Even when the virus is defeated, such changes in our lifestyle as distance learning at schools and universities, employee transfer to remote work, and online shopping will undoubtedly remain more or less in our life, thereby considerably increasing network loading.

In these circumstances, the issues of information security, protection of personal data, and support of the sustainable functioning of digital infrastructure are even more urgent. The software of many endpoints does not have built-in security features and needs regular updates, so they can be compromised by attackers, combined into a botnet, and used to generate malicious traffic [4, 5].

Traditional intrusion detection systems (IDSs) based on signature analysis have been used successfully for a long time as protection against network attacks. However, the method for the incoming traffic mapping to attack patterns has several disadvantages when analyzing big data. Firstly, the diversity of attacks requires creating huge constantly updated databases of malicious signatures, whose search slows down significantly the IDS operation [6, 7]. Secondly, certain attacks, such as coercive energy consumption and coercive topology modification, have divided sequences of operations, time intervals in the chain of actions, and some attacks may have nonlinear sequences of actions. Thirdly, there is a class of polymorphic attacks that have mutations (namely, local differences, omissions, and delays) of the action sequences

during the attack implementation. Such attacks adapt to a wide range of conditions, operating systems, and circumstances; they strive to avoid being scanned by security tools and aim to infect endpoints [8].

In the recent decade, information security researchers and IDS developers have focused on artificial intelligence methods [9–12]. The main advantages of the AI systems are their self-learning capacity, capability to detect unknown attacks, high speed of operation, and the absence of necessity to create large databases of malicious signatures. However, when processing not just big data, but huge data $>10^6$, these systems display a decrease in the accuracy of network traffic classification, and an increase in the processing time of incoming packets.

It is proposed to use quantum machine learning methods to analyze huge amounts of network traffic. The following problems are solved: development of a dataset for training and testing, development of a method for encoding network traffic in quantum computer terms, and assessment of the accuracy and performance of the developed quantum classifier.

QUANTUM MACHINE LEARNING

The basic idea of quantum machine learning is to merge machine learning and quantum computing techniques. Quantum computers use quantum mechanical phenomena, such as superposition and entanglement, to perform computations. The quality of decision-making during the data driven learning directly depends on the availability of large amounts of data and reliable processing. Quantum machine learning uses hybrid methods involving both classical and quantum processing, where computationally complex routines are outsourced to a quantum device [13, 14]. Since these routines can have more complex nature, they can be executed faster by quantum devices.

Quantum computers DWave and IBM Q are the most elaborated. DWave has a status of the "analog quantum computer," because it can solve only a narrow range of quantum annealing problems. At the same time, its claimed power is 2000 qubits. IBM Q is a project for the development of universal quantum computers that can execute arbitrary quantum algorithms. Systems of 20 qubits (commercial use) and open source Q Experience systems of 16 and 5 qubits are currently in operation. The main platforms for quantum computing implementation are presented in Table 1.

The Qiskit platform that allows managing resource overhead and adapting applications for specific devices appears to be the most promising. In addition, the platform contains the implemented algorithms: QSVM, VQC, and QGAN.

Quantum computing is successfully applied in various fields. Quantum distribution of cryptographic keys successfully solves the problem of distributing keys between users through open communication channels secured at a level of fundamental laws of nature. In 2014, Chinese researchers first implemented handwriting recognition using quantum computing [15]. The instantaneous processing of huge amounts of data and the solution of optimization problems make quantum technologies one of the most promising tools in the field of artificial intelligence and machine learning. For this reason, in 2013, Google and NASA created a joint laboratory for research in this area.

Researchers consider quantum machine learning to be one of the most promising areas in the field of quantum computing [16]. When processing large amounts of data, the use of quantum computing can achieve quadratic and even exponential acceleration compared to their classical counterparts. The existing libraries of quantum machine learning are presented in Table 2.

Tensorflow Quantum library developed by Google is chosen for further use based on the analysis of the available frameworks. Its main advantages are flexibility, availability of ready-to-use machine learning models and application packages, scalability by using hardware and software, large online community, and compatibility with the Keras library. Its disadvantage is supporting only NVIDIA GPUs.

STREAM DATASET GENERATION

The existing datasets, such as IEEEDataPort IoT Network Intrusion Dataset [17], Stratosphere Lab Malware on IoT Dataset [18], and NSW Canberra The BoT-IoT Dataset [19], contain network packets of two categories: "with attack" and "without attack." However, in practice, the approach to classifying individual packets has certain significant drawbacks. Firstly, the content of many packets from the available datasets is not malicious. For example, packets of the Denial of Service category do not contain malicious signatures; the classifier categorizes such packets as malicious solely based on the sender's IP address. Secondly, there are attacks that can distribute the payload over different packets. Thirdly, many attacks are carried out in stages, so an IDS based on packet classification is ineffective in such cases. Modern IDSs such as Snort and Suricata use packet-grouping techniques to analyze the network traffic. In this paper,

Table 1. Comparison of quantum computing platforms

Name	QDK	Qiskit	ProjectQ	Forest
Developer	Microsoft	IBM	ETH Zurich	Rigetti
1st release	January 2018	March 2017	January 2017	January 2017
Open source codes	+	+	+	+
Supported OS	Mac, Windows, Linux	Mac, Windows, Linux	Mac, Windows, Linux	Mac, Windows, Linux
Requirements	Visual Studio Code	Python 3.5+, Jupyter Notebooks Anaconda 3	Python 2 or 3	Python 3, Anaconda
Quantum programming language	Q#	Qiskit	ProjectQ	pyQuil
Quantum language	–	OpenQASM	–	Quil
Quantum hardware	–	IBMQX2 (5 qubits), IBMQX4 (5 qubits), IBMQX5 (16 qubits), QS1 1 (20 qubits)	Can be connected to the IBM backend	8 qubits
Simulation size	30 qubits locally, 40 qubits via Azure cloud	~25 qubits locally, 30 qubits via the cloud	~28 qubits locally	~20 qubits locally, 26 qubits with most API keys for QVM, 30+ with private access
Features	Built-in algorithms and examples	QASM code generation, topology-specific compiler, Slack community channel, circuit panel, Aqua library	Circuit building, connection to IBM server modules, availability of several library plug-ins	Quil code generation, algorithm examples in Grove, compiler for specific topology, noise features in simulator, Slack community channel

Table 2. Comparing quantum machine learning frameworks

Name	Tensorflow Quantum	PyTorch	Cirq	Strawberry Fields	PennyLane
Tutorials and examples	+++	+	+	++	++
Capability of modeling convolutional neural networks	+++	+++	++	++	++
Capability of modeling recurrent neural networks	++	++	+	++	+
Architecture: easy-to-use modular interface	+++	++	+	++	+
Operation speed	+++	+++	++	++	++
Multiple GPU support	++	++	+	++	+
Strengths	Fast processing of large data sets	Academic use and production	Studying the machine learning	Image processing	High-speed computing

we propose transforming the IoT Network Intrusion Dataset that contains six types of attacks and normal traffic into a stream dataset.

Such fields as 'tcp.srcport', 'tcp.dstport', 'udp.srcport', 'udp.dstport', 'tcp.checksum.status', and 'udp.checksum.status' were merged into 'srcport', 'dstport', and 'l4.checksum.status' to get rid of the dependency on the forth layer protocol type. The 'tcp.stream' and 'udp.stream' fields have been merged into the 'stream' field that is used to group packets into streams. The fields in the stream represent the transformed fields of the packets included in that stream. For example, for the 'ip.ttl', 'tcp.seq_raw', 'tcp.ack_raw' and 'tcp.window_size_value' fields, the average, minimum, and maximum values, and the standard deviation are calculated. The intervals between packets in the stream are analyzed separately, which allows one to judge the frequency of packet sending. For example, a flooding attack stream with equal intervals between packets will give a low average deviation of interval values. Table 3 provides a description of the stream fields used.

Table 3. Description of the stream fields

Field name	Description
ip.flags.rb.mean	Mean flag values
ip.flags.df.mean	
ip.flags.mf.mean	
tcp.flags.res.mean	
tcp.flags.ns.mean	
tcp.flags.cwr.mean	
tcp.flags.ecn.mean	
tcp.flags.urg.mean	
tcp.flags.ack.mean	
tcp.flags.push.mean	
tcp.flags.reset.mean	
tcp.flags.syn.mean	
tcp.flags.fin.mean	
frame.len.std	Minimum, maximum, and mean packet length and standard deviation
frame.len.min	
frame.len.max	
frame.len.mean	
frame.len.rate	Bps throughput
payload.std.mean	Minimum, maximum, and mean values of packet data bytes and standard deviation
payload.min.mean	
payload.max.mean	
payload.mean.mean	
payload.print.mean	Mean number of printable characters in packet data
srcport.std	Standard deviation of the source port value
dstport.std	Standard deviation of the destination port value
ip.checksum.status.std	Checksum analysis in Layer 3 and Layer 4 headers
ip.checksum.status.min	
ip.checksum.status.max	
ip.checksum.status.mean	
l4.checksum.status.std	
l4.checksum.status.min	
l4.checksum.status.max	
l4.checksum.status.mean	
ip.ttl.std	Minimum, maximum, and mean values of the packet lifetime and standard deviation
ip.ttl.min	
ip.ttl.max	
ip.ttl.mean	
tcp.seq_raw.std	Minimum, maximum, and mean values of the packet sequence number and standard deviation
tcp.seq_raw.min	
tcp.seq_raw.max	
tcp.seq_raw.mean	

Table 3. (Contd.)

Field name	Description
tcp.ack_raw.std	Minimum, maximum, and mean values of the packet confirmation number and standard deviation
tcp.ack_raw.min	
tcp.ack_raw.max	
tcp.ack_raw.mean	
tcp.window_size_value.std	Minimum, maximum, and mean values of the packet window size and standard deviation
tcp.window_size_value.min	
tcp.window_size_value.max	
tcp.window_size_value.mean	
count	Number of packets
duration	Stream duration
int.std	Minimum, maximum, and mean values of the inter-packet interval and standard deviation
int.min	
int.max	
int.mean	
prate	Number of packets per second
proto	Layer 4 protocol
category	Stream category

In the larger-scale experiments, the traffic was additionally generated with Nmap and Hping programs, and Wireshark program was used to capture packets. Tshark, the console version of Wireshark, received a .pcap file and a set of parameters at input; then, text data was read from the standard output stream and written in the CSV format. As a result, a streaming data set with more than 10 million records consisting of 58 fields was developed.

THE CODING METHOD DEVELOPED

A quantum model training based on real data requires the capability to translate data from a bit to a qubit representation. To solve this problem, a network stream coding method is developed.

First, the Cirq library is used to create a qubit that is to be placed in the circuit.

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

Next, the IP address is transformed into a number that is the basis for setting the rotation angle, and then a Pauli gate is added to the circuit. The number, to which the IP address is transformed, is used as the rotation angle.

```
>>> C=cirq.Circuit()
>>> a =make_num_from_ip(i[0])
>>> a =make_angle(a)
>>> C.append(cirq.rx(a)(qubit))
```

Then the algorithm starts working with the fields that contain numerical values. In total, there are 58 such fields. The rotation angle is generated for each of them, and the next Pauli gate is added to the circuit.

```
>>> for j in range(1,58):
>>> angle=make_angle(i[j])
>>> C.append(cirq.rx(angle)(qubit))
```

Therefore, each stream field is transformed into a number from 0 to π, after which a qubit is created for each stream. Then Pauli valves are successively applied to the qubit. The value obtained at the previous step is used as the rotation angle, and, then, the resulting quantum circuit is added to the list, which serves to train the classifier. In Fig. 1, the flowchart of the coding algorithm is shown.

An example of a quantum circuit is shown in Fig. 2.

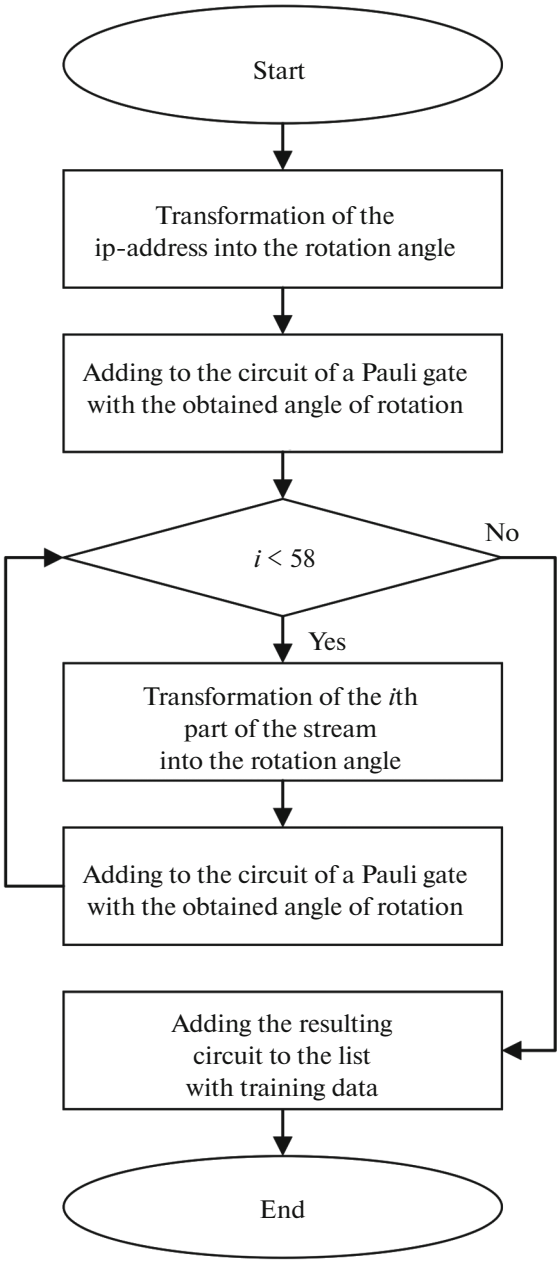

Fig. 1. Flowchart of the developed coding algorithm.

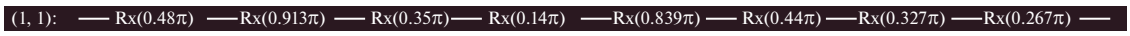

Fig. 2. Quantum circuit corresponding to one stream.

EXPERIMENTAL RESULTS

The test bench presented in Fig. 3 was designed for the experiments.

The Ubuntu operating system provides interaction between hardware and software. Python 3.7 programming language provides interaction between the user program and the operating system. The Cirq library is designed for creating and executing quantum circuits. Nvidia CUDA provides faster emulation for running quantum circuits. The Tensorflow platform is a tool for creating machine learning methods

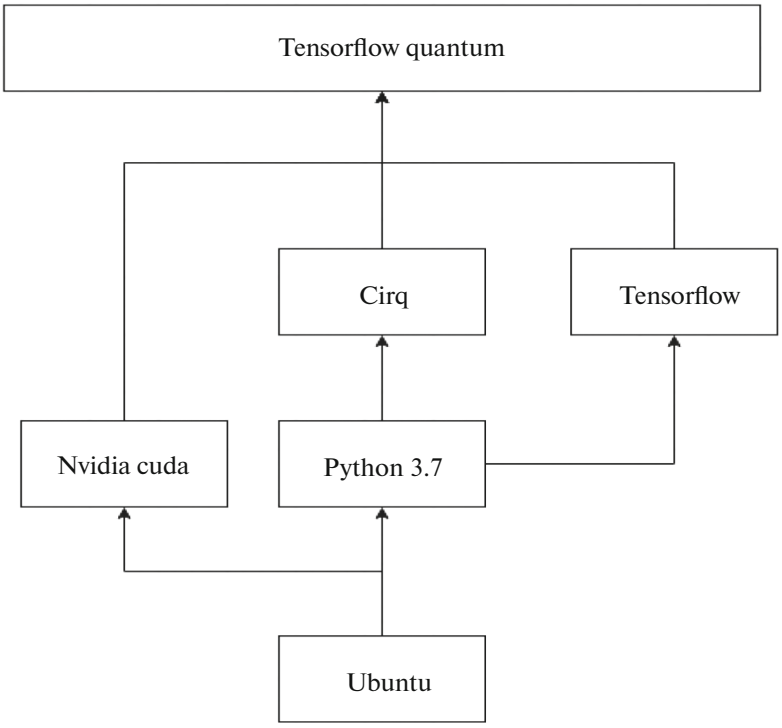

Fig. 3. Test bench.

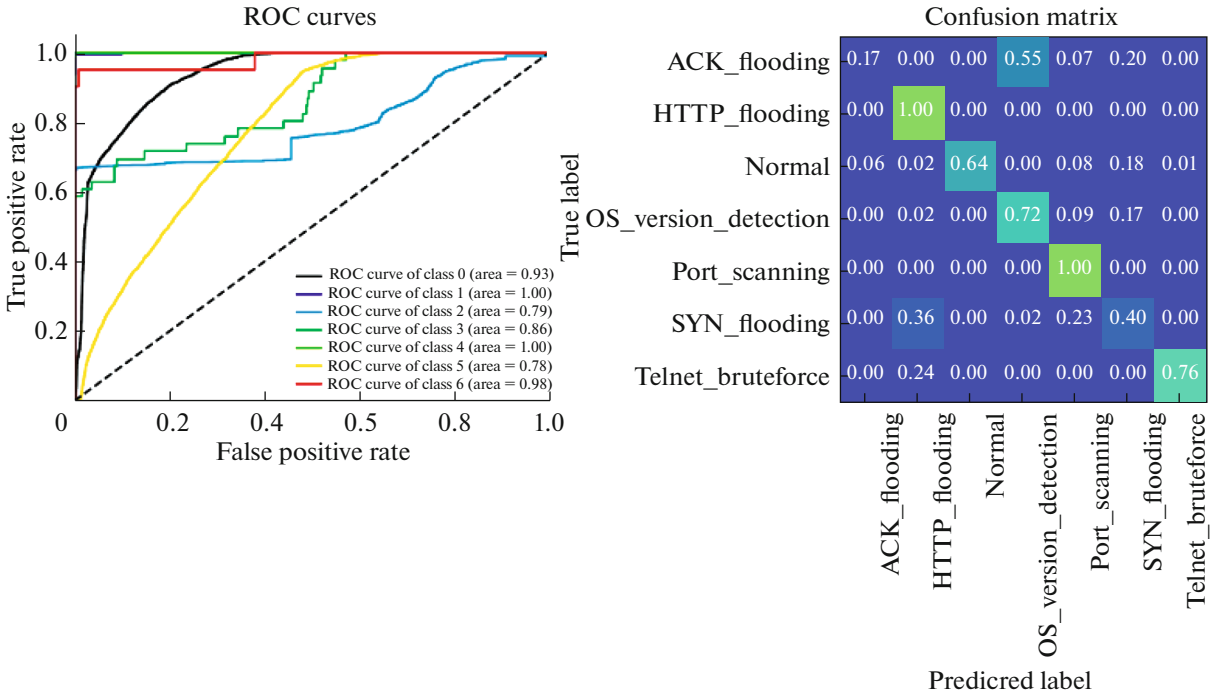

Fig. 4. Experimental results obtained using the classical SVM.

and neural networks. Tensorflow Quantum contains basic structures such as qubits, logic elements, circuits, and measurement operators.

The support vector machine (SVM) method was chosen to classify network streams. This method is effective when dealing with big data, has no tendency to overtraining, provides high accuracy when dealing

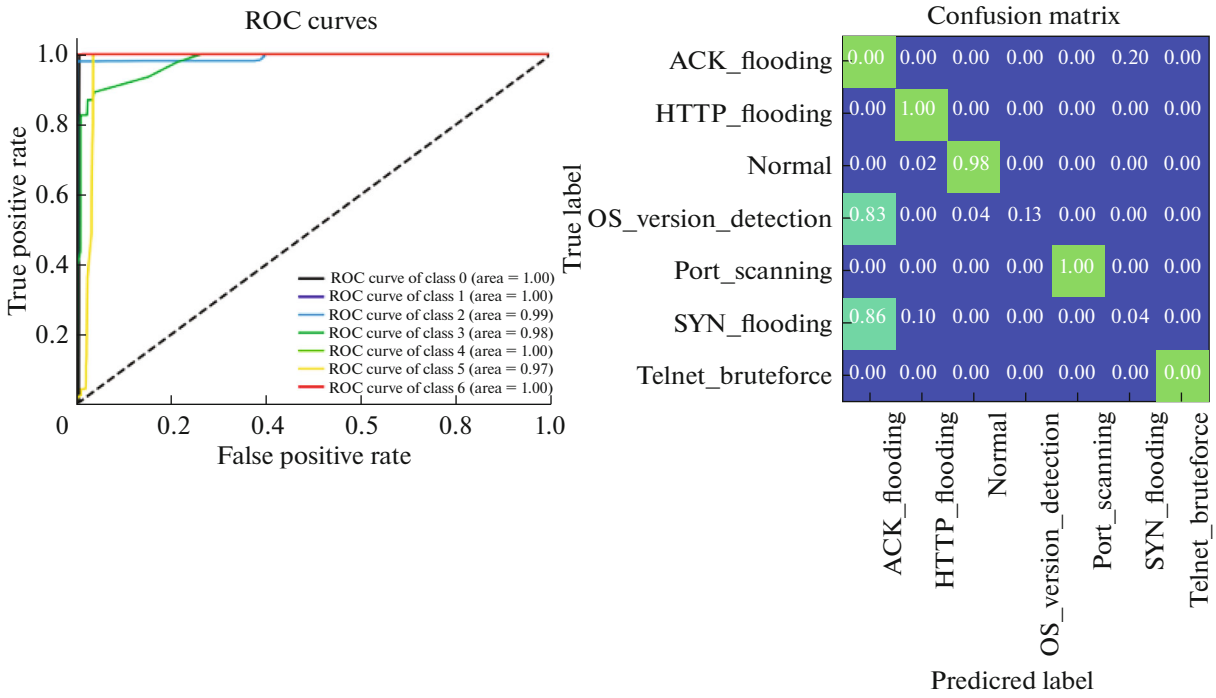

Fig. 5. Experimental results obtained using the quantum SVM.

with a large attribute space, and allows using the kernel trick [19]. In Figs. 4 and 5, the experimental results for the huge data classification problem (>10^6 records]) using classical SVM and quantum SVM are presented.

When the classical SVM is used, only HTTP flooding and port scanning attacks are detected with high accuracy. The accuracy of most other classes is in the range from 0.4 to 0.8. Practically, network streams containing the ACK flooding attack have never been detected correctly. When using the quantum SVM, the binary classification problem was solved with a high accuracy of 98%.

Table 4 shows the training times of classical and quantum SVM (QSVM) when processing huge data.

QSVM was more than twice faster than the classical one. If a more productive computer or a real quantum computer were used, the difference would be even larger.

Table 4. Comparison of SVM and QSVM learning times

Size of a training sample	SVM learning time, hours	QSVM learning time, hours
100000	0.5	0.4
200000	1.4	0.80
300000	2.2	1.3
400000	3.1	1.7
500000	4.4	2.3
600000	5.9	3
700000	6.8	3.3
800000	7.7	3.8
900000	8.9	4.5
1000000	9.6	5.2
2500000	12.6	6.1
5000000	16.9	8
10000000	22.4	10.6

CONCLUSIONS

The possibility of applying quantum-learning methods to solve the problem of Huge Data analysis is considered. The analysis of existing platforms and libraries for the implementation of quantum computing shows that the Qiskit platform and Tensorflow Quantum library are the most promising. A stream dataset is formed, in which network packets are grouped into streams. This approach allows improving the effectiveness of detection of more complex attacks that are carried out in several stages, distribute malicious code over several packets, and change IP addresses. The method for encoding a bit representation of network streams into a qubit representation is developed for quantum information processing. The experimental results show the superiority of quantum machine learning over classical machine learning in solving the problem of classification of a huge amount of network traffic. Quantum SVM outperforms classical SVM in both accuracy and processing speed. The use of quantum machine learning reduce the learning time more than twice.

FUNDING

The study was carried out within the scholarship of the President of the Russian Federation supporting young scientists and postgraduate students (SP-2714.2021.5).

CONFLICT OF INTEREST

The authors declare that they have no conflicts of interest.

REFERENCES

1. DE-CIX. Highest jump ever: DE-CIX Frankfurt reaches 9.1 Tbps. https://www.de-cix.net/en/about-de-cix/news/de-cix-frankfurt-reaches-9-1-tbps. Cited December 25, 2020.
2. Cisco VNI: by 2022 the annual Internet traffic in Russia will be three times higher. https://www.cisco.com/c/ru_ru/about/press/press-releases/2018/12-14.html. Cited December 25, 2020.
3. Nokia catalyzes the next chapter of the Internet; innovations deliver massive performance and heightened security needed for cloud and machine era. https://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/. Cited December 25, 2020.
4. Zegzhda, D.P., Lavrova, D.S., and Pavlenko, E.Yu., Management of a dynamic infrastructure of complex systems under conditions of directed cyber attacks, *J. Comput. Syst. Sci. Int.*, 2020, vol. 59, no. 3, pp. 358–370. https://doi.org/10.1134/S1064230720020124
5. Lavrova, D., Poltavtseva, M., and Shtyrkina, A., Security analysis of cyber-physical systems network infrastructure, *IEEE Industrial Cyber-Physical Systems (ICPS),* St. Petersburg, 2018, IEEE, 2018, pp. 818–823. https://doi.org/10.1109/ICPHYS.2018.8390812
6. Belenko, V., Krundyshev, V., and Kalinin, M., Intrusion detection for internet of things applying metagenome fast analysis, *Third World Conf. on Smart Trends in Systems Security and Sustainability,* London, 2019, IEEE, 2019, pp. 129–135. https://doi.org/10.1109/WorldS4.2019.8904022
7. Kotenko, I.V., Saenko, I.B., Polubelova, O.V., and Chechulin, A.A., Application of security information and event management technology for information security in critical infrastructures, *Tr. SPIIRAN*, 2012, no. 20, pp. 27–56.
8. Kalinin, M.O., Krundyshev, V.M., and Sinyapkin, B.G., Development of the intrusion detection system for the Internet of Things based on sequence alignment algorithm, *Autom. Control Comput. Sci.*, 2020, vol. 54, no. 8, pp. 993–1000. https://doi.org/10.3103/S0146411620080155
9. Noskov, A.N., Chechulin, A.A., and Tarasova, D.A., Investigation of heuristic approach to attacks on the telecommunications network detection based on data mining techniques, *Tr. SPIIRAN*, 2014, no. 37, pp. 208–224. https://doi.org/10.15622/sp.37.13
10. Lavrova, D., Semyanov, P., Shtyrkina, A., and Zegzhda, P., Wavelet-analysis of network traffic time-series for detection of attacks on digital production infrastructure, *SHS Web Conf.*, 2018, vol. 44, p. 00052. https://doi.org/10.1051/shsconf/20184400052
11. Malyshev, E., Moskvin, D., and Zegzhda, D., Application of an artificial neural network for detection of attacks in VANETs, *Autom. Control Comput. Sci.*, 2019, vol. 53, no. 8, pp. 889–894. https://doi.org/10.3103/S0146411619080194
12. Zhukovsky, E.V. and Marshev, I.I., Detection of malicious executable files based on machine learning algorithms, *Probl. Inf. Bezop. Komp'yut. Sist.*, 2019, no. 1, pp. 89–99.

13. Wiebe, N., Braun, D., and Lloyd, S., Quantum algorithm for data fitting, *Phys. Rev. Lett.*, 2012, vol. 109, p. 050505. https://doi.org/10.1103/PhysRevLett.109.050505
14. Bisio, A., D'Ariano, G.M., Perinotti, P., and Sedlák, M., Quantum learning algorithms for quantum measurements, *Phys. Lett. A*, 2011, vol. 375, no. 39, pp. 3425–3434. https://doi.org/10.1016/j.physleta.2011.08.002
15. Zhaokai, L., Xiaomei, L., Nanyang, X., and Jiangfeng, D., Experimental realization of a quantum support vector machine, *Phys. Rev. Lett.*, 2015, vol. 114, p. 140504. https://doi.org/10.1103/PhysRevLett.114.140504
16. Biamonte, J., Wittek, P., Pancotti, N., Rebentrost, P., Wiebe, N., and Lloyd, S., Quantum machine learning, *Nature*, 2017, vol. 549, pp. 195–202. https://doi.org/10.1038/nature23474
17. Kang, H. Ahn, D.H., Lee, G.M., Yoo, J.D., Park, K.H., and Kim, H.K., IoT network intrusion dataset, *IEEE Dataport*, 2019. https://doi.org/10.21227/q70p-q449
18. Parmisano, S., Garcia, M., and Erquiaga, M., Stratosphere laboratory. A labeled dataset with malicious and benign IoT network traffic. January 22th. https://www.stratosphereips.org/datasets-iot23. Cited December 25, 2020.
19. Koroniotis, N., Moustafa, N., Sitnikova, E., and Turnbull, B., Towards the development of realistic botnet dataset in the Internet of Things for network forensic analytics: Bot-IoT dataset, *Future Gen. Comput. Syst.*, 2019, vol. 100, pp. 779–796. https://doi.org/10.1016/j.future.2019.05.041
20. Cortes, C. and Vapnik, V., Support vector networks, *Mach. Learn.*, 1995, vol. 20, pp. 273–297. https://doi.org/10.1007/BF00994018

Translated by N. Semenova