Towards Quantum-Enhanced Machine Learning for Network Intrusion Detection

Arnaldo Gouveia[1,2] Miguel Correia[2]
[1]IBM Belgium, Brussels [2]INESC-ID, Instituto Superior Técnico, Universidade de Lisboa – Lisboa, Portugal

***Abstract*—Network Intrusion Detection Systems (NIDSs) are commonly used today to detect malicious activities. Quantum computers, despite not being practical yet, are becoming available for experimental purposes. We present the first approach for applying unsupervised Quantum Machine Learning (QML) in the context of network intrusion detection from the perspective of quantum information, based on the concept of quantum-assisted ML. We evaluate it using IBM QX in simulation mode and show that the accuracy of a Quantum-Assisted NIDS, based on our approach, can be high, rivaling with the the best conventional SVM results, with a dependence on the characteristics of the dataset.**

I. Introduction

The detection of security violations using *machine learning* (ML) has been extensively investigated [1]–[8]. *Intrusion Detection Systems* (IDSs) are tools used to detect such malicious activity. Network IDSs (NIDSs) in particular detect malicious activity in networks and are one of the best known contexts of ML application in the field [3]. IDSs and NIDSs can be classified as signature-based or anomaly-based [1]. A signature-based (N)IDS detects attacks by comparing the data flow under analysis to patterns stored in a signature database of known attacks. An anomaly-based (N)IDS typically detects anomalies using a model of normal behaviour of the monitored system and flagging behavior lying outside of the model as anomalous or suspicious. Signature-based IDSs can detect well-known attacks with high accuracy but fail to detect or find unknown attacks, whereas anomaly-based IDSs have that capacity. In this paper we focus on *anomaly-based NIDSs*.

Quantum computers, and among them *Noisy Intermediate-Scale Quantum* (NISQ) *computers*, aim to leverage quantum physics to perform computational tasks beyond the capabilities of the most powerful classical computers, potentially achieving *quantum supremacy* [9]–[11]. As the amount of Qbits and accuracy of quantum computers increases, the problem of when they exceed the state-of-the-art classical computation draws a great deal of attention. A definitive proof of quantum dominance is predicted in the near future [9]–[11], and one argument has already been made that it has been accomplished [12], although it has been disputed by competitors [13]. A consequence of quantum supremacy is that quantum computers may outperform their classical counterparts quadratically in terms of learning efficiency, or even exponentially in performance [10]. This is a motivation towards investigating the potential of *Quantum-Assisted Machine Learning* (QAML) in the context of network intrusion detection.

978-1-7281-8326-8/20/$31.00

On May 2016, IBM announced the first Near Term Quantum Computer, a NISQ, to be publicly available on the cloud [14]. The product is known as *IBM Q Experience* or simply IBM QX. Customers can programme a universal quantum computer, simulate quantum operations via an interface, and perform the same quantum operations on a quantum computer. IBM QX has been used in many experiences, e.g., quantum oscillator synchronization [15], variational quantum algorithms [16], and quantum search [17], and optimization approaches to convert common quantum circuits descriptions into the IBM QX have been proposed [18]. A 5-Qbit quantum computer was initially offered and, since June 2017, also a 16-Qbit quantum computer, respectively IBM QX2 and IBM QX3. Revised versions of these quantum computers, QX4 and IBM QX5, are available since September 2017. In 2019 and 2020 systems with 27 and 65 qubits were released. In order to use them, the desired quantum functionality, e.g., a quantum circuit, has to be properly mapped so that the underlying physical constraints are satisfied. Recently the plan to reach 100 Qbits by 2023 was announced.

We aim to use *an hybrid classical/quantum approach* where the data and feature learning are classical, whereas the classification algorithm is quantum. In this approach, classical data has to be converted into quantum data. This approach allows the implementation of quantum algorithms on the quantum computers available today, e.g., NISQs like IBM QX.

The IBM Quantum Experience (IBM Q) is accessible online giving users in the general public access to a set of IBM's prototype quantum processors. It is an example instance of cloud based quantum computing. As of May 2018, there are three processors on the IBM Quantum Experience: two 5-qubit processors and a 16-qubit processor. This service can be used to run algorithms and experiments, and explore tutorials and simulations around what might be possible with quantum computing.

The *first part* of the processing relates to the concept of feature learning. Principal component analysis (PCA) is arguably the most used technique for dimensionality reduction, so it has already been used in the context of intrusion detection [19], [20]. *Autoencoders* are a more general technique that, however, can also be used for dimensionality reduction, with several benefits:

Autoencoders can be trained using a variety of stochastic optimization methods that have been developed for training deep neural networks, becoming highly efficient for large datasets, in particular with statistical signatures that change

over time [21]. Autoencoders can handle high-dimensional training data, such as images, being suitable for online learning contexts in which new data arrives continuously, without the need of computationally expensive matrix operations as PCA. In summary, the advantages of using an autoencoder over PCA can be summarized as: autoencoders can process high-dimensional data; autoencoders can process large datasets; autoencoders are capable of online processing, processing new data as it arrives.

We use a quantum version of SVM, Quantum Support Vector Machine (QSVM), that uses a quantum kernel estimator and optimizer [22], [23].

In this work we aim at answering the following questions:

1) How effectively can autoencoders, with known NIDS datasets, produce an encoding with probability distribution functions (PDFs) – in other terms a latent space – related to the input variables so that they can processed by other ML algorithms, in an unsupervised fashion?
2) Can this result be used to interface with quantum computation systems in line with a quantum-enhanced approach for network intrusion detection?

The contributions of the this work are: (1) a study of autoencoders in the scope of unsupervised learning; (2) a study on quantum unsupervised ML by interfacing a dimensionality reduction technique based on autoencoders and quantum processors. We evaluate our approach using IBM QX and two well-established network intrusion detections datasets (UNB NSL KDD and UNSW NB15).

II. Background

A. Support Vector Machines

The SVM classification algorithm can classify data into two sub-groups. Let us assume there are M training data points $\vec{x_i}, i = 1, \ldots, M$, and each data has a label $+1$ or -1. Let us also assume that the number of dimensions of the feature space is N. In this case, the training data can be written as $\{(\vec{x_i}, y_i) : \vec{x_i} \in \mathbb{R}^N, y_i = \pm 1\}$.

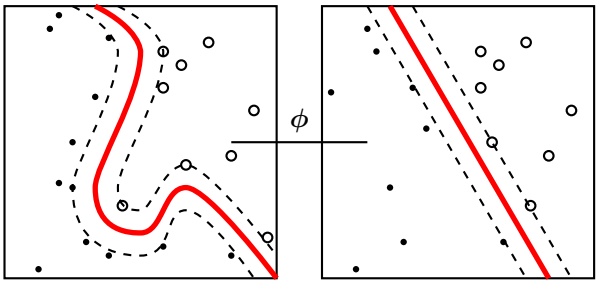


Fig. 1. SVM kernel trick illustrated with samples from two classes here represented by black and white dots.

The frontier between two classes can be represented by the distance between two support hyperplanes $\vec{w} \cdot \vec{x} + b = 1$ and $\vec{w} \cdot \vec{x} + b = -1$. With the expressions of the support hyperplanes defined, the distance between them can be represented by $\frac{2}{\|\vec{w}\|}$. Thus, SVM aims at maximizing the margin $\frac{2}{\|\vec{w}\|}$, which is the same as minimizing $\frac{\|\vec{w}\|^2}{2}$. Originally SVM learning algorithms could only solve linearly separable problems. To address this limitation the kernel trick was introduced. The mapping, represented in the figure, is characterized by the choice of a class of functions known as kernels. Kernels use specific functions $\phi(x)$ to map the data into a higher-dimensional space, see Figure 1. This procedure is justified by Cover's theorem [24], which guarantees that any data set becomes (linearly) separable as the data dimension grows. In this situation SVM finds the hyperplane in a new space that maximizes the margin and minimizes misclassifications. A classifier with a large margin is prone to be more accurate.

B. Quantum-Assisted Machine Learning

Quantum computing can help speed up some types of problems. It may be able to find new, easier ways to achieve economic optimisation. Nevertheless, as stated above, the existing quantum computers are very small. We therefore plan to use QAML, a hybrid classical / quantum solution where data and function learning are classical, while the classification algorithm is quantum.

An example of a classical / quantum hybrid algorithm that relies on quantum circuits considered to be inefficiently scaled by classical methods was provided in [23]. In this work, the authors define and apply two binary classification approaches utilizing supervised learning. Such classification algorithms are related to regular SVMs. The aim in this research is to introduce a non-linear function map that takes the data to be graded into a space where it can be linearly segregated.

The method used in this work is the second method explained in [23] directly exploits this connection to classical SVMs. Here only the overlap of the feature map circuits is estimated for each pair in the training set. That is, the quantum computer is only used once to compute the kernel matrix for the training set. The optimal separating hyperplane and the support vectors can be obtained efficiently in the training set size by solving the conventional dual optimization problem on a classical computer. This problem is concave and therefore efficiently solvable on a classical computer. Once the support vectors have been determined the quantum computer needs to be queried again for classification, when the kernel for a new datum is estimated. The quantum processor is thus used twice: once during training and then again in the classification phase, in both cases to estimate the quantum kernel.

III. Quantum Support Vector Machines

A. Quantum Encoding Preliminary Concepts

Suppose we have a dataset D of N instances:

$$D = \{x_1, ..., x_i, ..., x_N\} \tag{1}$$

where each x_i is a real number. In *basis encoding*, each instance x_i is encoded as $|b_i\rangle$ where b_i is the binary repre-

sentation of x_i. The dataset D can then be represented as a superposition of all computational basis states:

$$|\psi\rangle = \frac{1}{\sqrt{N}} \sum_{i=1}^{N} |b_i\rangle \tag{2}$$

In *amplitude encoding*, data is encoded in the amplitudes of quantum states. The above dataset D can be encoded as:

$$|\psi\rangle = \frac{\sum_{i=1}^{N} x_i |i\rangle}{||D||} \tag{3}$$

The encoding scheme proposed consists of a layer of Hadamard gates on each qubit, followed by an operator, $U_\Phi(\vec{x})$, then another layer of Hadamards, and $U_\Phi(\vec{x})$ again.

A feature map on n-Qbits is generated by the unitary where H denotes the conventional Hadamard gate and $\mathcal{U}_\Phi(\vec{x}) = U_{\Phi(\vec{x})} H^{\otimes n} U_{\Phi(\vec{x})} H^{\otimes n}$.

The $U_{\Phi(\vec{x})}$ gate consists of $\Phi()$ gates, which can operate on one or two qubits. The single-qubit Φ operation is just a Z rotation by x^i on the i^{th} qubit.

Quantum Support Vector Machines (QSVMs) use a quantum kernel estimator to estimate the kernel function and optimize the classifier directly. QSVMs are an interesting application for NISQ computers [22], [23]. The scheme we are proposing deals with the original problem of supervised learning: the creation of a SVM classifier and is fully explained in [23]. For this question, we provide data from the training set T and the test set S from the subset $\Omega \subset \mathbf{R}^d$. Both are assumed to be labeled as $m : T \cup S \to \{+1, -1\}$ unknown to the algorithm. The algorithm just gets test data labels T. The aim is to predict an estimated map on the test set $\tilde{m} : s \to \{+1, -1\}$ in such a way that it fits the true map $m(\vec{s}) = \tilde{m}(\vec{s})$ on the test data $\vec{s} \in S$ with a high probability.

By means of SVMs, the data is projected non-linearly to a high dimensional space, *a function area* , where a hyperplane is configured to differentiate the named samples. The quantum variant of this method, QSVM, has already been proposed by Rebentrost et al. [25]. Their scheme makes it possible to accomplish incremental progress if data is given in a coherent superposition.

A feature map on n-Qbits is generated by the unitary $\mathcal{U}_\Phi(\vec{x}) = U_{\Phi(\vec{x})} H^{\otimes n} U_{\Phi(\vec{x})} H^{\otimes n}$, where H denotes the conventional Hadamard gate and

$$U_{\Phi(\vec{x})} = \exp\left(i \sum_{S \subseteq [n]} \phi_S(\vec{x}) \prod_{i \in S} Z_i \right), \tag{4}$$

is a diagonal gate in the Pauli Z basis (Figure 2(b)). This circuit will act as an initial state on $|0\rangle^n$. The $\phi S(\vec{x}) \in \mathbf{R}$ coefficients encode the $\vec{x} \in \Omega$ data. In general, any diagonal $U\Phi(\vec{x})$ can be used if it can be implemented efficiently. This is the case, for example, when only $\leq$ 2interactions are considered. The exact evaluation of the inner-product between two states created from a similar circuit with only one diagonal layer $U\Phi(\vec{x})$ is ¶-hard [26]. Nevertheless, in the experimentally relevant context of the estimate of the additive

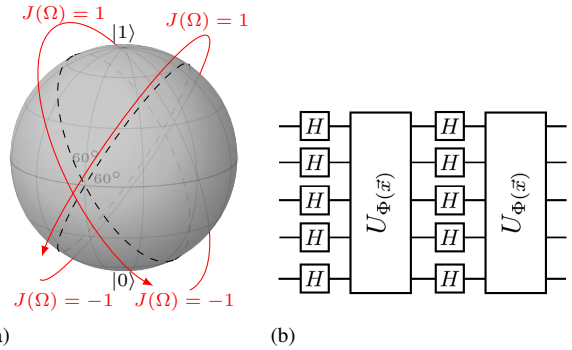


Fig. 2. **Quantum Kernel Functions**: On the left; A classical dataset mapped into the Bloch sphere. $J_\Omega()$ returns binary labels $(-1, 1)$ and can be mapped onto the Bloch sphere by using a non-linear feature map: for a single Qubit $U_{\Phi(x)} = Z_x$ is a phase-gate of angle $x \in \Omega$. The mapped data can be separated by the hyperplane given by normal $\vec{w}$. On the right: For the general circuit $U_{\Phi(\vec{x})}$ is formed by products of single- and two-Qubit unitaries that are diagonal in the computational basis. The circuit family depends non-linearly on the data through the coefficients $\phi_S(\vec{x})$ with $|S| \leq 2$. The experimental implementation of the parameterized diagonal two-Qbit operations is done using CNOTs and $Z-$gates [23].

defect, the simulation of a single layer preparation circuit can be effectively performed. by uniform sampling [27].

Every n-Qbit device operation can be approximated to an appropriate level of accuracy by the gates of the universal gate collection. The nearly universal set contains only two types of gates: the Toffoli Gate and the Hadamard Gate [28] [29].

The Toffoli Gate is common for (reversible) classical computing and has a totally classical character in quantum circuits as it preserves its computational basis. This seems to suggest that the quantum benefit is provided by gates which do not maintain the theoretical base, i.e. the Hadamard gates. Since circuits such as those used in this simulation are likely to result in the successful testing of a purely classical computer.

B. Quantum kernel estimation

Quantum kernel estimation: The protocol chosen implements the SVM directly. A classical SVM is used for classification. The quantum computer is used twice in this protocol:

- First, the kernel $K(\vec{x}_i, \vec{x}_j)$ is estimated on a quantum computer for all pairs of training data $\vec{x}_i, \vec{x}_j \in T$. Here it will be convenient to write $T = \{\vec{x}_1, \ldots, \vec{x}_t\}$ with $t = |T|$; also let $y_i = m(\vec{x}_i)$ be the corresponding label. The optimization problem for the optimal SVM can be formulated in terms of a dual quadratic program that only uses access to the kernel.

$$L_D(\alpha) = \sum_{i=1}^{t} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{t} y_i y_j \alpha_i \alpha_j K(\vec{x}_i, \vec{x}_j), \tag{5}$$

  subject to $\sum_{i=1}^{t} \alpha_i y_i = 0$ and $\alpha_i \geq 0$ for each i. This problem is concave whenever $K(\vec{x}_i, \vec{x}_j)$ is a positive definite matrix. The solution to this problem will be given by a nonnegative vector $\vec{\alpha} = (\alpha_1, \ldots, \alpha_t)$.
- The quantum computer is used a second time to estimate the kernel for a new datum $\vec{s} \in S$ with all the support

vectors. The optimal solution $\vec{\alpha}^*$ is used to construct the classifier

$$\tilde{m}(\vec{s}) = \text{sign}\left(\sum_{i=1}^{t} y_i \alpha_i^* K(\vec{x}_i, \vec{s}) + b\right). \quad (6)$$

IV. Description of the Experiment

This section presents the full quantum-enhanced intrusion detection scheme. The scheme can be divided in the following phases:

1) *Obtain the input data*, i.e., network flows corresponding to a period of time. The NIDS can be configured to detect intrusions by inspecting from the last T period of time. For example, T can be set to 10 minutes, 30 minutes, or 1 hour.
2) *Preprocess the data*, e.g., do feature normalization.
3) *Encode the data*. This phase does dimensionality reduction, i.e., summarizes the input data to a size that is appropriate in terms of dimension to be inputted to the QSVM() function. As explained before, this phase is processing the data with an Autoencoder and obtaining its latent space.
4) *Normalize the data* returned by the autoencoder for preparing it for the next phase.
5) *Quantum processing*, done by a quantum computer, in order to detect intrusions in the input data. As explained, this phase involves running a QSVM() function algorithm to classify flows as normal or anomalous.
6) *Repeat* for the next period of time or data sample in parallel. Notice that the period may overlap the previous if that is considered useful for some reason.

The detection process has to be preceded by a *training* process. Training involves the same phases 1 to 5 above, except that the QSVM in phase 5 is executed in training mode.

V. Experimental Evaluation

An important component of a NIDS evaluation is a good benchmarking dataset. We use two recent datasets that have been designed specifically and carefully to evaluate NIDSs. They contain a large number of attacks and are labelled with information about which network flows are normal traffic and which are malicious. We designate them NSL-KDD [30] and NB15 [5], [6].

A. Data encoding and preparation

The phases presented in Section IV were executed the following way:

1) Obtaining the input data: the flows were extracted from the KDD-NSL and NB15 datasets:
   - NSL-KDD: a sample of 100 data samples for training and 50 data samples for testing has been extracted from the KDDTrain+.arff file.
   - NB15: a sample of 150 data samples for training and 50 data samples for testing has been extracted from the NB15 train file.
2) Preprocessing of the data: the non-numeric features were encoded into numeric values, then normalized into the interval $[0, 1]$.
3) Encoding of the data: we considered an implementation based on the *autoencode()* function provided by the Ruta package [31], [32] for the R statistical computing package [33].
   - NSL-KDD: For the NSL-KDD dataset *autoencode()* has been tested with four activation functions – Linear, Hyperbolic Tangent, Relu and Sigmoid –, in order to understand the performance in the four cases.
   - NB15: For the NB15 dataset the *autoencode()* has been tested with a single activation function, *sigmoid*, as there was no point in repeating the previous evaluation.
4) Normalization: the data returned by *autoencode()* was normalized to the range $[-1, 1]$.
5) Quantum processing: the resulting dataset has been processed by the QSVM() function[1] of the Qiskit quantum computing framework. This function runs the binary classification at the quantum simulation level.

For comparison purposes, a classical SVM algorithm was run over the entire datasets after being processed by the autoencoder. Specifically, we used the liquidSVM, a package that implements SVMs and configures hyper-parameters automatically using k-fold cross validation.[2]

B. Experimental process for the Quantum component

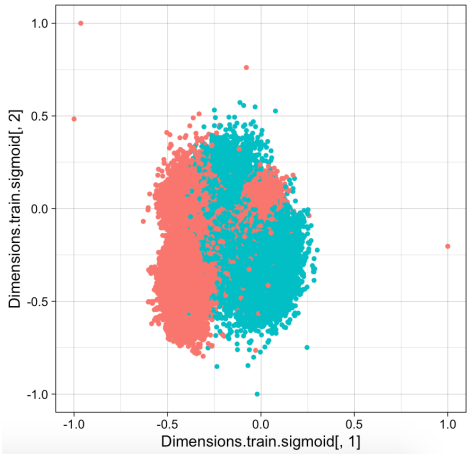
1) For each data point, the process of encoding data involves a series of gates that will transform the initial state, $|0\rangle$, into some state that depends on $\vec{x}|\Phi(\vec{x})\rangle$. One such method is amplitude encoding, in which the resulting state would be. In this scheme, the data is encoded into the basis state amplitudes of the state:

$$\begin{pmatrix} a_1 \\ \vdots \\ a_{2^n} \end{pmatrix} \leftrightarrow \begin{pmatrix} x_1 \\ \vdots \\ x_{2^n} \end{pmatrix}, \sum_i |x_i|^2 = 1, x_i \in \mathbb{R}$$

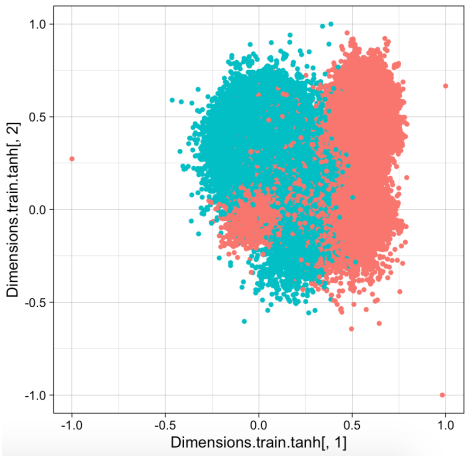
2) According to [23], [34] , classifiers based on quantum circuits should only provide a quantum advantage if the kernel used in the quantum space is difficult to estimate classically: $K(\vec{x}, \vec{z}) = |\langle \Phi(\vec{x}) | \Phi(\vec{z}) \rangle|^2$.
3) Inference: Inference in quantum machine learning amounts to the unitary operation, U, such that the loss is minimized when measuring $U\langle\Phi(\vec{x})|$. Mathematically, U is a matrix with complex elements which is unitary, where n is the number of qubits; However, in practice, such a general matrix cannot be directly trained, since a real quantum computer has to employ combinations of a limited set of gates rather than arbitrary unitary operations. What is used are layers of rotation gates separated by entangling layers composed of controlled Pauli gates [35].

[1] Qiskit Aqua algorithms on Qiskit
[2] liquidSVM on GitHub

4) Measurement: After all the operations in the circuit have been performed, some observable is measured $\mathbf{f} = Z_1 Z_2$ as the parity function and a random unitary $V \in SU(4)$. Each measurement provides one of a number of discrete values. For calculating the expected value for an observable, so measurements are made repeatedly in order to obtain the mean. The end result is that the expected value of a measurement from the entire circuit is $\langle \Phi(\vec{x}) | V^{\dagger} \mathbf{f} V | \Phi(\vec{x}) \rangle$), where $\mathbf{f}$ represents the observable measured. The measurement is then used to produce a label and a value for the loss which is used to optimize the parameters used in the inference step.



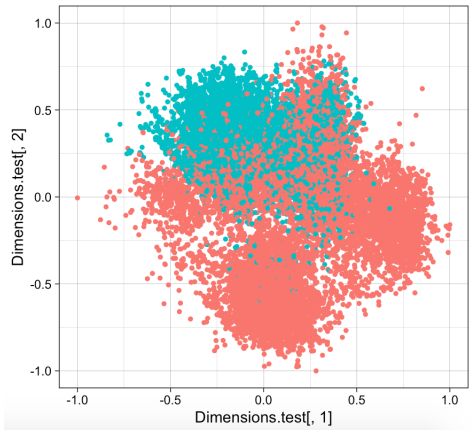
(a)



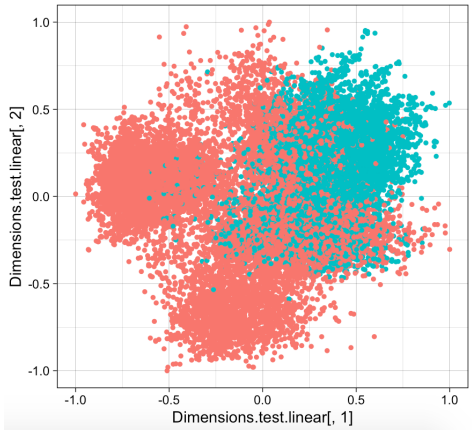
(b)

Fig. 3. Distribution at the encoding layer (Linear activation and Elu function case). This diagram shows the location of the different encoding for normal and malicious samples. Weak superposition can be seen with a continuous distribution between the two types of samples. Data obtained from the *autoencode()* Ruta package function applied to the NSL-KDD dataset. Two cases are shown: (a) NSL-KDD test dataset processed with sigmoid activation.Training loss: 9.4375; (b) NSL-KDD test dataset processed with Hyperbolic activation. Training loss: 9.3450.

*Evaluation:*The values of accuracy of our *quantum classification algorithm*, based on QSVM, are shown in Table I and Table II. Complementing this table, a depiction of the accuracy versus circuit depth can be seen in Figure 8. It shows



(a)



(b)

Fig. 4. Distribution at the encoding layer for the NSL KDD test dataset. As the statistical distribution (SD) does not match the SD of the training set the data from the test datasets were not used for testing. Data obtained from the *autoencode()* Ruta package function applied to the NSL-KDD dataset. Two cases: (a) NSL-KDD test dataset processed with sigmoid activation: training loss: 9.3377; (b) NSL-KDD test dataset processed with linear activation: training loss: 9.4114.

that the accuracy increases with the quantum circuit depth as theoretically foreseen.

When these performance figures are compared with classical SVM classification (Table II) we can see that they compare favourably. In the NSL-KDD case both figures are very close, 0.92 acuuracy for QSVM and 0.93 accuracy for SVM. Regarding the NB15 dataset case we can see that the accuracy values provided by the QSVM classifier are somewhat lower than the ones provided by the SVM classifier $Acc = 0.75$.

In Figures 9 and 10 the quantum kernel matrices obtained in the simulations are shown. More defined matrices are observable for the best performing cases, in line with what was expected. A clear visualization of the quality of the kernel obtained is linked to the capacity to observe clear spots for the upper left and lower right of the figure denoting a more distinctive density and as a consequence a better capacity for SVM to separate data points.

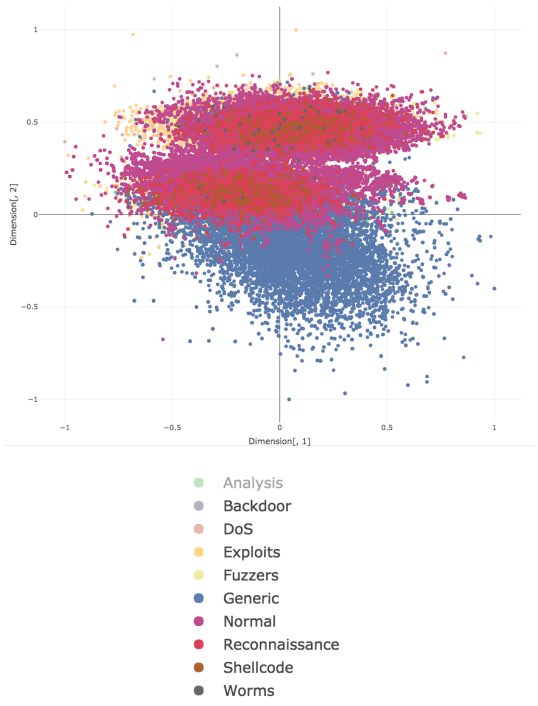


Fig. 5. Distribution at the encoding layer (Linear activation function case). This diagram shows the location of the different encoding for normal and malicious samples. Strong superposition can be seen with a continuous distribution between the several types of samples. Data obtained from the *autoencode()* Ruta package function applied to the NB15 dataset.

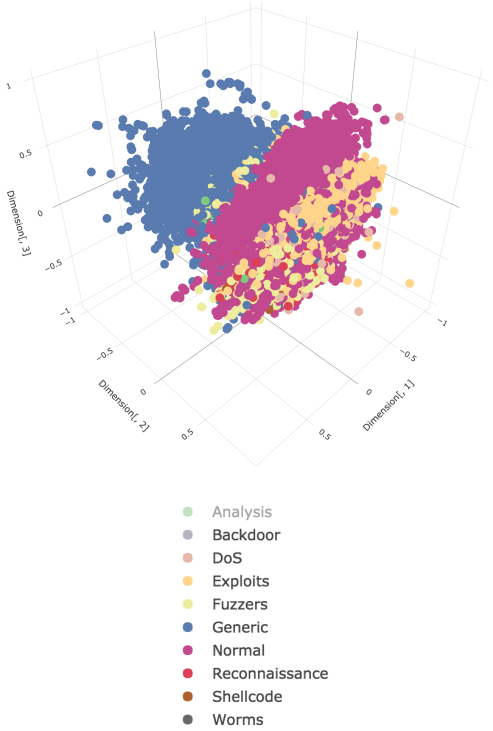


Fig. 6. NB15 train dataset distribution at the encoding layer (Linear activation function case). This diagram shows the location of the different encoding for normal and malicious samples. Strong superposition can be seen with a continuous distribution between the several types of samples, each one corresponding to a different type of flow. Data obtained from the *autoencode()* Ruta package function applied to the NB15 dataset.

Accuracy values				
Depth	2	3	4	5
NSL KDD	0.8	0.92	0.92	0.92
NB15	0.6	0.64	0.64	0.64

TABLE I
ACCURACY TABLE VERSUS CIRCUIT DEPTH FOR THE QUANTUM SVM SIMULATION FOR THE NSL KDD DATASET CASE.

The values of accuracy obtained for the NSL-KDD dataset – $Acc = 0.92$ for the Quantum SVM and $Acc = 0.93$ for classical (Tables I and II) are very close. The kernel matrices show how the SVM algorithm can draw the margin to differentiate the two classes of traffic (normal/malicious). In this experiment, we can see kernel matrices can label the classes properly most of the time. The kernel matrices are depicted in Figures 10 and 9. One can observe that the distinctiveness of the matrices are much more evident for the NSL-KDD case, where at the same time better separability was obtained (with better accuracy values).

	Accuracy	**Pos Val Error**	**Neg Val Error**
NB15	0.75	0.132	0.383
NSL-KDD	0.93	0.136	0.019

TABLE II
ACCURACY TABLE FOR THE CLASSICAL SVM (LIQUIDSVM, NON-QUANTUM) EXECUTION OVER THE ENTIRE DATASET FOR BOTH NSL-KDD AND NB15.

VI. CONCLUSIONS

An hybrid solution to submit the classification of network flows to a quantum computer has been demonstrated. The performance values obtained have shown that the approach is feasible and may in the future allow NISDs to benefit from quantum assisted computing. One particular but fundamental issue relates to the need for statistical similarity between the testing and training dataset for the test dataset to be used with the models trained. Hence results also show that some work has to be done to optimize the datasets for submission to the quantum algorithms, here used in simulation mode. Judging from the accuracy values obtained for the classical SVM and Quantum SVM trials, encouraging data seems to hint at showing advantage for the quantum processing. This should be object of further study.

Acknowledgements. This research was supported by by the European Commission under grant agreement number 830892 (SPARTA) and by national funds through Fundação para a Ciência e a Tecnologia (FCT) with reference UIDB/50021/2020 (INESC-ID).

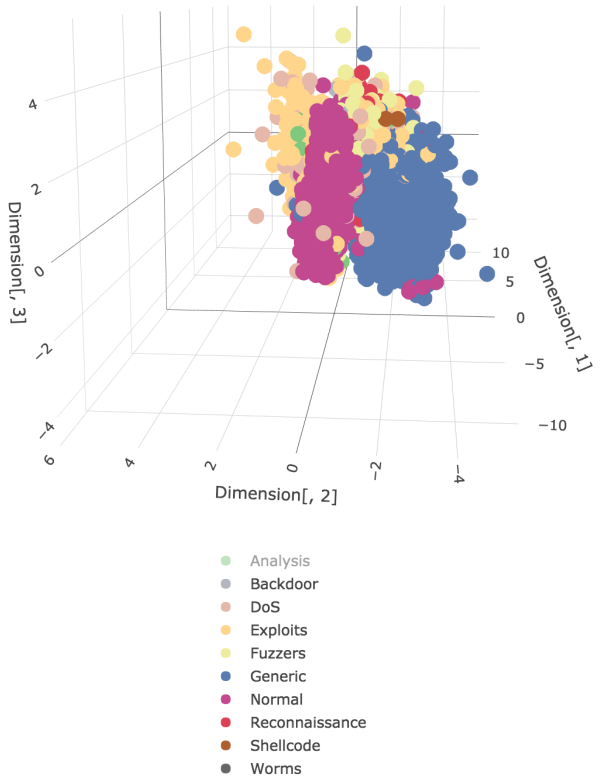


Fig. 7. NB15 Test dataset distribution at the encoding layer (Sigmoid activation function case). As the statistical distribution (SD) does not match the SD of the training set the data from the test datasets were not used for testing. This diagram shows the location of the different encoding for normal and malicious samples. Weak superposition can be seen with a continuous distribution between the two types of samples. Data obtained from the *autoencode()* Ruta package function applied to the NB15 dataset. Training loss: 8.2805

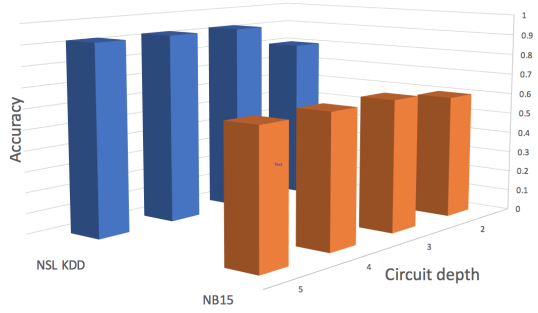


Fig. 8. Accuracy as a function of the quantum circuit depth for both datasets. These values can be compared with the accuracy obtained for the SVM in the classical domain executed over the entire datasets: Acc=0.75 for the NB15 dataset and Acc=0.93 for the NSL-KDD dataset.

References

[1] H. Debar, M. Dacier, and A. Wespi, "A revised taxonomy of intrusion detection systems," *Annales des Télécommunications*, vol. 55, no. 7, pp. 361–378, 2000.

[2] S. Dhaliwal, A.-A. Nahid, and R. Abbas, "Effective intrusion detection system using XGBoost," *Information*, vol. 9, no. 7, pp. 1–24, 2018.

[3] P. García-Teodoro, J. Díaz-Verdejo, G. Maciá-Fernández, and E. Vázquez, "Anomaly-based network intrusion detection: Techniques, systems and challenges," *Computers & Security*, vol. 28, no. 1-2, pp. 18–28, Feb. 2009.

[4] R. Mitchell and I.-R. Chen, "A survey of intrusion detection techniques for cyber-physical systems," *ACM Computing Surveys*, vol. 46, no. 4, pp. 55:1–55:29, Mar. 2014.

[5] N. Moustafa and J. Slay, "UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in *2015 Military Communications and Information Systems Conference*, Nov 2015, pp. 1–6.

[6] ——, "The evaluation of network anomaly detection systems: Statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set," *Information Security Journal: A Global Perspective*, vol. 25, no. 1-3, pp. 18–31, 2016.

[7] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *Proceedings of the 2nd IEEE International Conference on Computational Intelligence for Security and Defense Applications*, 2009, pp. 53–58.

[8] P. Wheeler and E. Fulp, "A taxonomy of parallel techniques for intrusion detection," in *Proceedings of the 45th Annual Southeast Regional Conference*, 2007, pp. 278–282.

[9] B. Villalonga, D. Lyakh, S. Boixo, H. Neven, T. S. Humble, R. Biswas, E. G. Rieffel, A. Ho, and S. Mandrà, "Establishing the quantum supremacy frontier with a 281 Pflop/s simulation," *ArXiv*, vol. abs/1905.00444, 2019.

[10] I. L. Markov, A. Fatima, S. V. Isakov, and S. Boixo, "Quantum supremacy is both closer and farther than it appears," *ArXiv*, vol. abs/1807.10749, 2018.

[11] D. Ristè, M. P. da Silva, C. A. Ryan, A. W. Cross, A. D. Córcoles, J. A. Smolin, J. M. Gambetta, J. M. Chow, and B. R. Johnson, "Demonstration of quantum advantage in machine learning," *npj Quantum Information*, vol. 3, no. 1, p. 16, 2017.

[12] F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin, R. Barends, R. Biswas, S. Boixo, F. G. Brandao, D. A. Buell *et al.*, "Quantum supremacy using a programmable superconducting processor," *Nature*, vol. 574, no. 7779, pp. 505–510, 2019.

[13] A. Khrennikov, "Echoing the recent Google success: Foundational roots of quantum supremacy," *arXiv preprint arXiv:1911.10337*, 2019.

[14] J. M. Gambetta, J. M. Chow, and M. Steffen, "Building logical qubits in a superconducting quantum computing system," *npj Quantum Information*, vol. 3, no. 1, p. 2, 2017.

[15] M. Koppenhöfer, C. Bruder, and A. Roulet, "Quantum synchronization on the ibm q system," 2019.

[16] M. Lubasch, J. Joo, P. Moinier, M. Kiffner, and D. Jaksch, "Variational quantum algorithms for nonlinear problems," 2019.

[17] K. Das and A. Sadhu, "Constant time quantum search algorithm over a datasets: An experimental study using ibm q experience," *ArXiv*, vol. abs/1810.03390, 2018.

[18] A. Zulehner, A. Paler, and R. Wille, "An efficient methodology for mapping quantum circuits to the IBM QX architectures," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 38, pp. 1226–1236, 2017.

[19] Z. Elkhadir, K. Chougdali, and M. Benattou, "Intrusion detection system using PCA and kernel PCA methods," in *Proceedings of the Mediterranean Conference on Information & Communication Technologies*, A. El Oualkadi, F. Choubani, and A. El Moussati, Eds., 2016, pp. 489–497.

[20] K. K. Vasan and B. Surendiran, "Dimensionality reduction using principal component analysis for network intrusion detection," *Perspectives in Science*, vol. 8, pp. 510–512, 2016.

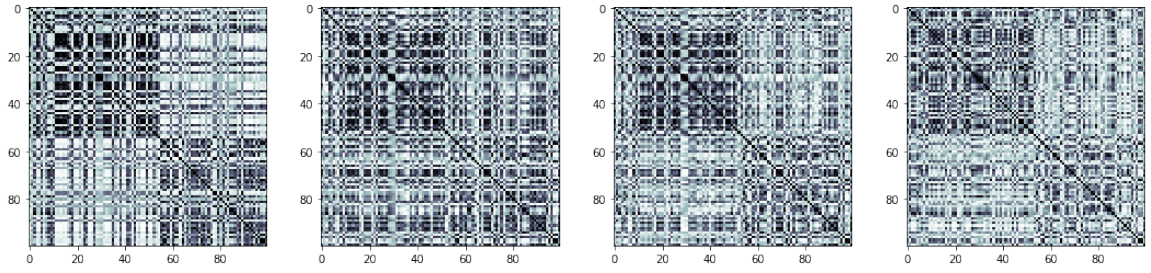
[21] E. Plaut, "From principal subspaces to principal components with linear autoencoders," *ArXiv*, vol. abs/1804.10253, 2018.

[22] J. Preskill, "Quantum Computing in the NISQ era and beyond," *Quantum*, vol. 2, p. 79, Aug. 2018.

[23] V. Havlíček, A. Córcoles, K. Temme, A. Harrow, A. Kandala, J. Chow, and J. Gambetta, "Supervised learning with quantum-enhanced feature spaces," *Nature*, vol. 567, pp. 209–212, 03 2019.

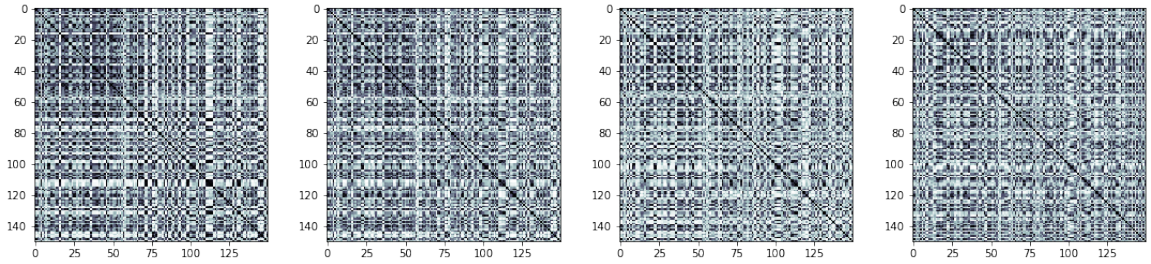
[24] T. M. Cover, "Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition," *IEEE Transactions on Electronic Computers*, vol. 14, no. 3, pp. 326–334, June 1965.

[25] P. Rebentrost, M. Mohseni, and S. Lloyd, "Quantum support vector machine for big data classification," *Physical Review Letters*, vol. 113, 07 2013.



(a) (b) (c) (d)

Fig. 9. Kernel matrices for several depth figures. Ideal kernel matrices containing the inner products of all data points used for training with the NSL-KDD dataset. Data obtained with a Qiskit simulation. (a) Kernel matrix for depth=2, Accuracy=0.8; (b) Kernel matrix for depth=3, Accuracy=0.92; (c) Kernel matrix for depth=4, Accuracy=0.92; (d) Kernel matrix for depth=5, Accuracy=0.92. Clearly visible the correlation between distinguishability of the kernel space and the accuracy values obtained.



(a) (b) (c) (d)

Fig. 10. Kernel matrices for several depth figures. Ideal kernel matrices containing the inner products of all data points used for training with the NB15 dataset. Data obtained with a Qiskit simulation. (a) Kernel matrix for depth=2. Accuracy=0.60; (b) Kernel matrix for depth=3. Accuracy=0.64; (c) Kernel matrix for depth=4. Accuracy=0.64; (d) Kernel matrix for depth=5.Accuracy=0.64. Clearly visible the correlation between distinguishability of the kernel space and the accuracy values obtained.

[26] L. A. Goldberg and H. Guo, "The complexity of approximating complex-valued ising and tutte partition functions," *Computational Complexity*, vol. 26, pp. 765–833, 2014.

[27] T. F. Demarie, Y. Ouyang, and J. F. Fitzsimons, "Classical verification of quantum circuits containing few basis changes," *Physical Review A*, vol. 97, no. 4, Apr 2018.

[28] Y. Shi, "Both Toffoli and controlled-not need little help to do universal quantum computing," *Quantum Info. Comput.*, vol. 3, no. 1, pp. 84–92, Jan. 2003.

[29] D. Aharonov, "A simple proof that Toffoli and Hadamard are quantum universal," 2003.

[30] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani, "Toward developing a systematic approach to generate benchmark datasets for intrusion detection," *Computers & Security*, vol. 31, no. 3, pp. 357–374, May 2012.

[31] D. Charte, F. Herrera, and F. Charte, "Ruta: Implementations of neural autoencoders in r," *Knowledge-Based Systems*, vol. 174, pp. 4 – 8, 2019.

[32] D. Charte, F. Charte, S. García, M. J. del Jesús, and F. Herrera, "A practical tutorial on autoencoders for nonlinear feature fusion: Taxonomy, models, software and guidelines," *Information Fusion*, vol. 44, pp. 78–96, 2018.

[33] R Core Team, *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria, 2019. [Online]. Available: https://www.R-project.org/

[34] M. Schuld and N. Killoran, "Quantum machine learning in feature Hilbert spaces," *Phys. Rev. Lett.*, vol. 122, p. 040504, Feb 2019.

[35] M. Schuld and F. Petruccione, "Supervised learning with quantum computers," 2018.