Future Generation Computer Systems 166 (2025) 107632

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Review article

Quantum machine learning algorithms for anomaly detection: A review

Sebastiano Corli [a,b], Lorenzo Moro [b,c], Daniele Dragoni [d], Massimiliano Dispenza [e], Enrico Prati [b,f,*]

[a] Department of Physics, Politecnico di Milano, Milano, Italy
[b] Istituto di Fotonica e Nanotecnologie, Consiglio Nazionale delle Ricerche, Piazza Leonardo da Vinci, 32, Milano, Italy
[c] Department of Electronics, Information and Bioengineering, Politecnico di Milano, Milano, Italy
[d] High Performance Computing Laboratory, Leonardo s.p.a., Via R. Pieragostini, 80, Genova, Italy
[e] Quantum Technologies Leonardo Lab, Leonardo S.p.A., Via Tiburtina, KM 12, 400, Roma, Italy
[f] Department of Physics, Università degli studi di Milano, Via G. Celoria, 16, Milano, Italy

ARTICLE INFO

Keywords:
Quantum computing
Quantum machine learning
Neural networks
Anomaly detection
Cybersecurity

ABSTRACT

The advent of quantum computers has justified the development of quantum machine learning algorithms, based on the adaptation of the principles of machine learning to the formalism of qubits. Among such quantum algorithms, anomaly detection represents an important problem crossing several disciplines from cybersecurity, to fraud detection to particle physics. We summarize the key concepts involved in quantum computing, introducing the formal concept of quantum speed up. The survey provides a structured map of anomaly detection based on quantum machine learning. We have grouped existing algorithms according to the different learning methods, namely quantum supervised, quantum unsupervised and quantum reinforcement learning, respectively. We provide an estimate of the hardware resources to provide sufficient computational power in the future. The survey provides a systematic and compact understanding of the techniques belonging to each category. We eventually provide a discussion on the computational complexity of the learning methods in real application domains.

Contents

* Corresponding author at: Department of Physics, Università degli studi di Milano, Via G. Celoria, 16, Milano, Italy.
E-mail addresses: sebastiano.corli@polimi.it (S. Corli), lorenzo.moro@polimi.it (L. Moro), daniele.dragoni@ext.leonardo.it (D. Dragoni), massimiliano.dispenza@leonardo.it (M. Dispenza), enrico.prati@unimi.it (E. Prati).

https://doi.org/10.1016/j.future.2024.107632
Received 20 August 2024; Received in revised form 11 November 2024; Accepted 22 November 2024
Available online 9 December 2024

1. Introduction

Anomaly detection takes advantage from a wide range of artificial intelligence algorithms, which – combined with human supervision – may raise the degree of protection and of integrity of systems and data. On the other hand, the advent of quantum computing has made possible to implement quantum algorithms on real prototypical – but already commercial – quantum hardware. Such algorithms include machine learning-related algorithms which may inherit the quantum speed-up typical of quantum algorithms. The differentiation among quantum computing architectures (gate based [1,2], adiabatic [3,4], measurement based [5–7]), encoding (digital [8] versus continuous variables [9–12]), hardware (several substrates from solid state to trapped ions [13,14] to photons [15]), the diversity of the quantum algorithms and the unclear advantage carried by some of them in the field of quantum machine learning, and also the range of potential application domains and the different methods to achieve the same goal, call for a systematic review of what has been done and what is known in the field, in order to address more efficiently the investigation towards meaningful, feasible and relevant applications. The quantum machine learning algorithms proposed in literature for anomaly detection purposes are updated to Q1 of 2024, and clustered by applying the criteria of training method. Indeed, the latter represents the criterion which drives the choice among a families of algorithms. Therefore, we classify the quantum algorithms for machine learning according to the same classification of classical algorithms, namely among supervised learning, unsupervised learning and reinforcement learning, respectively. Despite its recent birth, the topic of quantum machine learning has been systematically reviewed in the time span 2015–2023 [2,16–20]. We privileged the literature which includes a practical implementation on either an actual quantum computer or the simulator of some existing hardware. The literature reviewed by such sources is integrated by more recent articles not included there. In the next sections we describe the summary of aims and AI algorithms used in anomaly detection, the basics of quantum computing and quantum advantage, the key quantum algorithms developed in the field which are relevant for anomaly detection purposes, and we systematically analyze the most recent advancements in the field of quantum machine learning applied to anomaly detection. In the second section, we summarize the application for quantum machine learning in the field of anomaly detection. In the third section, we introduce the concept of quantum advantage, along with a classification for the possible quantum speedups. In the fourth section, the groundings of quantum computing are introduced: from the definition of qubits, qudits and qumodes to the encoding of classical information into these units of computation. In the fifth section, different architectures of quantum computation (adiabatic and circuital models) are introduced, along with the support of classical computers. The sixth section is dedicated to focus on the role of HPC and classical computation to interface with quantum hardware and algorithms. Quantum neural networks and variational circuits, to translate on a quantum device the classical neural networks, are explained in section seven. In the remaining sections, a survey on specific quantum algorithms for anomaly detection is provided, classified with respect to the categories of supervised, unsupervised and reinforcement learning. A summary of all the exposed algorithms can be found at Table 3, while the available datasets for training quantum circuits are summed up by Table 4. In order to cover all of the classes of quantum algorithms and computational architectures in Fig. 1, we select and summarize a paramount paper for each of these classes. For instance, Killoran's work [10] in 2019 defined how to build continuous variables neural networks for quantum computers, along with their employment for anomaly detection, while Tacchino in 2019 [21] proposed a model of fully quantum neural perceptron. The other works we report are from Useche [22] (2022) for performing classification tasks with qudits, Herr [23] (2021) for introducing the QGAN in the field of anomaly detection, Moro [24] (2023) to boost the performances on the Restricted Boltzmann Machine via an annealer, the Harrow, Hassidim, Lloyd paper [25] (2009), which introduced the namesake HHL algorithm employed for the support vector machines (and beyond), and the paper by Albarràn-Arriagada [26] (2018) for the quantum Reinforcement Learning.

2. Application domains for quantum anomaly detection

As this review elaborates on the intersection of quantum machine learning methods with applications in the anomaly detection, we first briefly assess which classes of algorithms are related to such topic, from image recognition and data classification to clustering analysis. Moreover, another topic of interest is provided by the domain of applications: cybersecurity is a major focus for quantum machine learning [10,27–30] (and for all–around machine learning), but a considerable interest arises on disparate topics such as big data for science, i.e. detecting Higgs–boson decay at LHC collider [31,32], geophysical analysis [33], detection of new particles at LHC [34,35] or even audio recognition [36,37].

Typically, cybersecurity is characterized as a collection of technologies and processes designed to protect computers, networks, programs, and data against malicious activities, attacks, harm, or unauthorized access. In the field of cybersecurity, anomaly detection is of paramount importance. Many datasets exist, including intrusion analysis, malware analysis, and spam analysis, which are used for different purposes [38].

All cybersecurity matter, while becoming increasingly crucial to all modern industrial and institutional activities, has also grown in the past decades in terms of complexity of the multiple bodies and structure which have been created to implement cyber defence functionalities. SOCs (Security Operation Centers) are for example hugely complex cybersecurity systems, exploited to monitor infrastructures, to supervise networks, to detect threats also able to guarantee early warning and security awareness. Once security incidents have been detected, they have also to be managed. The so called Computer Emergency Response Teams (CERTs) come into play. Such complexity of cyber monitoring and countermeasure systems is strongly related to an equivalent rearrangement of the threat side, where more conventional private cyber crime groups and cyber terrorists or hacker sources were joined by government linked teams. Such groups carry out a so called cyber warfare by systematically implementing cyber attacks to national or institutional IT services [39]. Complexity grows together with continuous and rapid adaptations and modification of threats themselves. Ransomware groups and other malicious players, e.g., are changing their initial access vectors while the digital attack surface and vulnerabilities shift, also exploiting commercial tools to disguise their breaches and deploying new ransomware schemes. Anomaly detection [40] is a fundamental tool for this task and such continuously evolving cyber threat landscape ultimately calls for actions by SOCs and CERTs to be largely become automated only asking for man-in-the-loop in few very critical steps; which on its turn naturally links to the use of machine learning [41–44], as a means of automated response.

The main purpose of such algorithms is to provide early warning of attack, possibly even before the attack is launched [45]. Cyber intelligence deals with amount of data, their heterogeneity and their high production rate. AI is believed [46] capable to enhance cyberspace security more effectively than conventional methods for three reasons, namely:

- Better adaptation to detect anomalous, faster and more accurate operations.
- Naturally handling high volumes of data.
- Learning over time to respond better to threats.

Moreover, between different AI methods such as neural networks, fuzzy logic, expert system, machine learning, and deep learning, the two latter bring the most achievements. In the field of cybersecurity, the applications span mainly on malware detection, intrusion detection (ID), endpoint detection (ED), phishing detection and advanced persistent threat (APT). All of them take advantage of different methods based on a number of possible algorithms (or combinations of them): Naive Bayes method, Support Vector Machines (SVM), Decision Trees and, more recently deep Neural Networks [47–50]. One should keep in mind that since spurious transactions are far fewer than the normal ones, the highly imbalanced data makes fraud detection very challenging and calls for ways to address it beyond the traditional machine learning approach [28]. Furthermore, the development for instance of new fraud detection methods is made more difficult due to the severe limitation of the exchange of ideas in fraud detection [51]. More recently, Reinforcement Learning proved to be a robust but flexible method to prevent cyber attacks [52–54], also thanks to a vast range of available algorithms, such as Deep Deterministic Policy Gradient (DDPG) [55], Trust Region Policy Optimization (TRPO) [56], Proximal Policy Optimization (PPO) [57], Generalized Advantage Estimation (GAE) [58].

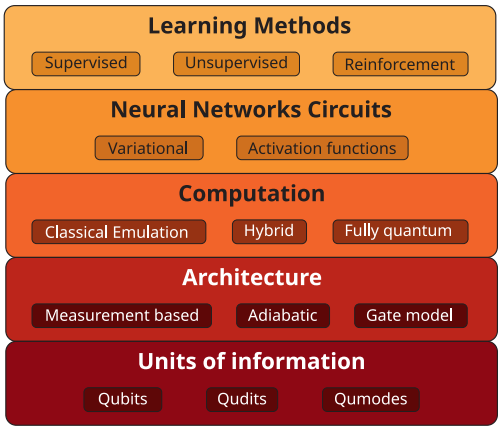

Fig. 1. A list of topics to categorize the field of quantum machine learning and its algorithms.

In the broader field of anomaly detection, Neural Networks have also been successfully employed for medical and public health domain [59], fault detection for mechanical components and structural damage detection [60–62]. As for image and pattern recognition or data text analysis, along with detection of spurious elements in datasets from the corresponding domains, techniques such as Support Vector Machines [36,63], Neural Networks [64,65] and clustering based algorithms [66,67] have been deployed. Support Vector Machines have also been addressed to foil phishing attacks, achieving a rationale performance with 99.6% of True Positive Rate and 0.44% of False Positive Rate [68]. Finally, also *advanced persistent threat* can be performed by deep learning algorithms, such as dilated convolutional auto-encoders (DCAEs) algorithm [69].

Anomaly detection represents a major method in the field across these different purposes, and it turns out to be promisingly explored also in the field of quantum machine learning. In the next sections we therefore anticipate the key concepts of quantum information and quantum algorithms, so to consistently review anomaly detection from the perspective of the solution of security-related tasks, as outlined above.

3. Quantum algorithms and quantum advantage

Quantum algorithms belong to computational classes defined by quantum Turing machines [70] instead of the conventional Turing machines. In the complexity theory, for both classical and quantum computation, the runtime of an algorithm is measured in terms of number of elementary operations N involved [71]. In the circuit models for quantum computing, such operations match the application of native gates on the hardware for the gate-based architecture. Therefore, the same problem can be mapped as NP but not P for classical machines while it can be of class BQP if defined on the Hilbert spaces on which quantum Turing machines rely [72]. Jager and Krems demonstrated that there exists a feature map and a quantum kernel that make

Table 1
Speed-up quantification for given quantum machine learning subroutines. The table is taken from [17]. Some of the reported algorithms will be further discussed in detail in the next sections.

Methods	Speed-up	Amplitude amplification	HHL	Adiabatic	qRAM
Bayesian inference	$O(\sqrt{N})$	Yes	Yes	No	No
Online perceptron	$O(\sqrt{N})$	Yes	No	No	Optional
Least-squares fitting	$O(\log(N))$	Yes	Yes	No	Yes
Classical Boltzmann machine	$O(\sqrt{N})$	Yes/No	Optional/No	No/Yes	Optional
Quantum Boltzmann machine	$O(\log(N))$	Optional/No	No	No/Yes	No
Quantum PCA	$O(\log(N))$	No	Yes	No	Optional
Quantum support vector machine	$O(\log(N))$	No	Yes	No	Yes
Quantum reinforcement learning	$O(\sqrt{N})$	Yes	No	No	No

variational quantum classifiers and quantum kernel support vector machines efficient solvers for any BQP problem. Therefore, some problems which are classically NP but BQP from a quantum approach, can be solved exponentially faster by using the appropriate quantum algorithm instead of a classical algorithm. Such property is called quantum speed-up. Different degrees of speed-up have been defined by a panel of experts in 2014 [73], which can be summarized as follows:

Provable quantum speed-up: there is a proof that there can be no classical algorithm that performs as well or better than the quantum algorithm. Example: Grover's algorithm scales quadratically better than classical, provided there exist an oracle to mark the desired state;

Strong quantum speed-up: the quantum algorithm performs better than the best possible, not necessarily known explicitly, classical algorithm (i.e. lower bound to classical algorithm is not known) Example: Shor's quantum algorithm to factorize prime numbers (grows polynomially instead of exponentially with the number of digits of the prime number);

Common quantum speed-up: the quantum algorithm performs better than the *best available* classical algorithm, as often the best available classical algorithm for strong quantum speed-up is not known;

Potential quantum speed-up: if there is no consensus about which is the best available classical algorithm, it refers to the comparison with an arbitrary classical algorithm;

Limited quantum speed-up it refers to the benchmark of two corresponding algorithms. Example: classical and quantum annealing.

The landscape of quantum algorithms shows a range of possible speed-ups, which is even more difficult to systematize for the domain of quantum machine learning methods, where the (possible) speed-up is sometimes not quantified. The evaluation increases in difficulty as more architectures and more encoding methods are possible (see Section 5 and Fig. 1).

From the point of view of the implementation of quantum machine learning proposed in literature, three approaches are common. First, by the direct speed-up of machine learning techniques, by using algebra-related algorithms like those in the Table 1, of which the HHL algorithm is paramount [17]. Secondly, by implementing variational quantum circuits and finally multilayer perceptron-based quantum neural networks. Recently, the competitiveness of quantum models based on variational circuits compared to classical models has been raised by the demonstration [20] that explicit models [74,75] outperform implicit models, and data re-uploading models exponentially outperforms simple explicit models. Explicit models rely on a parametric definition of the unitary operators $\hat{U}(\theta)$, θ collecting the family of parameters. Such models therefore can be easily encoded into any variational quantum circuit. The QAOA algorithm is an example of explicit model.

4. Encoding data with quantum systems

4.1. From bits to qubits

The qubit is the fundamental unit of information encoded by a quantum computer. The qubit is a quantum state, defined by a vector $|\psi\rangle$ in a Hilbert space $\mathcal{H} = \mathbb{C}^2$. It is the transposition of the classical bit, but instead of assuming two discrete values, formally $\{0, 1\}$, such values are transposed in a vector state [8]:

$$0 \to |0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad 1 \to |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \quad \Rightarrow \quad |\psi\rangle = \cos\left(\frac{\theta}{2}\right)|0\rangle + e^{i\phi}\sin\left(\frac{\theta}{2}\right)|1\rangle \tag{1}$$

The $\{|0\rangle, |1\rangle\}$ vectors form a orthonormal basis in the $\mathbb{C}^2$ space, therefore any qubit can be set in a linear combination as in the rightmost expression for $|\psi\rangle$ [2, pag.94]. It follows immediately that $\langle\psi|\psi\rangle = 1$, i.e. the state vector $|\psi\rangle$ is normalized. The a and b coefficients represent the probability for the state to be found either in the $|0\rangle$ or $|1\rangle$ state. All the logic operations on a single qubit are implemented by operators $\hat{U}$ which transform such state from $\mathbb{C}^2$ to $\mathbb{C}^2$: $\hat{U}|\psi\rangle = |\psi'\rangle$. To preserve the normalization of the vector $|\psi\rangle$, the single-qubit operators $\hat{U}$ are given by the unitary group $SU(2)$. The most known single-qubits operators are the NOT gate $\hat{X}$, the $\hat{Z}$ gate and the Hadamard gate $\hat{H}$:

$$\hat{X} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad \hat{Z} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, \quad \hat{H} = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{2}$$

The $\hat{X}$ gate flips the $|0\rangle$ state into the $|1\rangle$ and vice versa, acting in fact as the classical NOT gate. The Hadamard gate, instead, is an isomorphism on $\mathbb{C}^2$ mapping the computational basis $\{|0\rangle, |1\rangle\}$ into the conjugated one $\{|+\rangle, |-\rangle\}$ one, where $|\pm\rangle = (|0\rangle \pm |1\rangle)/\sqrt{2}$. For a comparison, see Table 2.

4.1.1. Encoding bits into qubits

It is possible to encode bits into qubits in several ways. The most vanilla method consists of a 1-to -1 encoding, making one bit i to be encoded by one quantum state $|i\rangle$. In literature, such encoding is referred to as multi-register encoding [76], where N is the number of available qubits, and $|i\rangle$ can assume the values of $|0\rangle$ or $|1\rangle$. When all the qubits are initialized in the $|0\rangle$ state, to flip a single qubit it suffices to apply a $\hat{X}$ NOT gate. Such operation can be performed simultaneously, yielding a circuit depth of $O(1)$ operations to be performed. However, an enhancement can be given by the superposition principle: in a register of N qubits, it is possible to encode $n = 2^N$ bits, each permutation being given by the superposition of the different states. In fact, the analog encoding makes usage of all the possible permutations of 0 and 1. In the following line, the multi-register and the analog encoding are

respectively shown [77]:

$$\bigotimes_{i=0}^{N-1}|i\rangle\,, \qquad \sum_{i=0}^{2^N-1} c_i|i\rangle \tag{3}$$

Therefore, within the analog encoding (rightmost expression) via N qubits it is possible to store $n = 2^N$ units of memory, which means N bits require $\log_2(N)$ available qubits. At this point, one should also notice that in the latter case the encoding process requires an exponential number $O(N)$ of steps [76], calling for methods or approximations to circumvent the issue. In the past, quantum RAM (qRAM) have been proposed to feed directly the register of qubits asked to load the data [78], but hitherto no example does exists [77,79,80].

One last family of data encoding is given by the Hamiltonian encoding. Within this frame, the data need to be formatted into a Hamiltonian operator, for instance the Ising Hamiltonian. Such a technique turns to be useful mainly in order to perform quantum annealing, see Section 5.2, and it will be explored in details in Section 4.4.

4.2. Qudits

The qudit is a d-dimension generalization of the qubit. Instead of a basis spanned by 2 vectors, the space of qudits is generated by D vectors. It is possible to choose a computational basis [81] $\{|0\rangle, |1\rangle, \dots, |D-1\rangle\}$, therefore a vector in this $\mathbb{C}^d$ space will given by

$$|\psi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle + \cdots + \alpha_{D-1}|D-1\rangle \tag{4}$$

The state in Eq. (4) must be normalized [82], so that $|\alpha_0|^2 + |\alpha_1|^2 + \cdots + |\alpha_{D-1}|^2 = 1$. Referring to Section 4.1.1, given M qudits it is possible to store D^M units of binary memory. A feature comparison between qubits and qudits is provided in Table 2. The qubits operators from Eq. (2) can be adapted in the qudits formalism as well. The phase Hadamard gate should be able to put any $|k\rangle$ element from the computational basis $\{|i\rangle\}_{i=0}^{D-1}$ into a superposition over all the generators of such basis [83], along with a generalization for the NOT and the $\hat{Z}$ operators [22,84]:

$$\hat{H}_D = \frac{1}{\sqrt{D}}\sum_{i,j=0}^{D-1}\exp\left\{i\theta_{i,j}^D\right\}|i\rangle\langle j|, \qquad \hat{X}^m = \sum_{k=0}^{D-1}|k\oplus m\rangle\langle k|,$$
$$\hat{Z} = \sum_{k=0}^{D-1} e^{\frac{2\pi i}{k}}|k\rangle\langle k| \tag{5}$$

where the angle $\theta_{n,m}^D$ is defined as $\theta_{n,m}^D = \frac{2\pi}{D}nm$. The qudits in their conjugated basis, after a Hadamard transformation, are reported in Table 2. The $\oplus$ is the sum modulo D (for the qubits, $D = 2$). In [22], it is introduced the control gates $C\hat{U}$ as a two-qudits operators. Here a target and a control qudits need to be specified: the operation $\hat{U}$ holds on the target qudit only if the control one is set in the $|1\rangle$ state, otherwise it does not. The generalized control gate operator is $C\hat{U}^k$: the $\hat{U}$ unitary operator is applied on the target qudit when the control one is set in the $|k\rangle$ state.

4.3. Qumodes (continuous variables)

In the frame of Continuous Variables (CV) for quantum computing, quantum information is encoded in continuous degrees of freedom such as the amplitudes of the electromagnetic field [10]. Specifically, the unit of information we described before as a qubit is substituted by the so-called qumode. In the phase space representation, the state of a single qumode is described by two real and conjugated variables such as $(x, p) \in \mathbb{R}^2$, whether N qumodes are depicted by $(\mathbf{x}, \mathbf{p}) \in \mathbb{R}^{2N}$. Qumode states also have a representation as vectors or density matrices in the countably infinite Hilbert space spanned by the Fock states $\{|n\rangle\}_{n=0}^{+\infty}$, which are the eigenstates of the photon number operator $\hat{n} = (\hat{x}^2 + \hat{p}^2 - 1)/2$, where $\hat{x}$ and $\hat{p}$ are the position and momentum operators. A comparison [9] between qubit, qudits and qumode architectures is provided in Table 2.

Table 2
Comparing features from qubits, qudits and qumodes, see Fig. 1.

Units of information		
Qubits	Qudits	Qumodes
Computational basis		
$\{\|0\rangle, \|1\rangle\}$	$\{\|i\rangle\}_{i=0}^{D-1}$	$\{\|q\rangle\}_{q\in\mathbb{R}}$
Scalar product		
$\langle k\|l\rangle = \delta_{k,l},\ k,l \in \{0,1\}$	$\langle k\|l\rangle = \delta_{k,l},\ k,l \in \{0,\dots,D-1\}$	$\langle q\|q'\rangle = \delta(q-q'),\ q,q' \in \mathbb{R}$
Superposition		
$\|\psi\rangle = a\|0\rangle + b\|1\rangle$	$\|\psi\rangle = \sum_{i=0}^{D-1}\alpha_i\|i\rangle$	$\|\psi\rangle = \int \mathrm{d}q\,\psi(q)\|q\rangle$
Conjugated basis		
$\|\pm\rangle = \frac{1}{\sqrt{2}}(\|0\rangle \pm \|1\rangle)$	$\|\theta_k\rangle = \frac{1}{\sqrt{D}}\sum_{i=0}^{D-1}\theta_{i,k}^D\|i\rangle$	$\|p\rangle = \int \mathrm{d}q\, e^{ipq}\|q\rangle$

4.3.1. Operations in the frame of continuous variables

Killoran et al. [10] report some possible operations to implement in the CV frame. First, the position and momentum operators are introduced:

$$\hat{X} = \int_{\mathbb{R}} x|x\rangle\langle x|\mathrm{d}x, \quad \hat{P} = \int_{\mathbb{R}} p|p\rangle\langle p|\mathrm{d}p \tag{6}$$

Being $\hat{X}$ and $\hat{P}$ defined on the entire real line, the orthonormality relations hold:

$$\langle p|p'\rangle = \delta(p-p'), \quad \langle x|x'\rangle = \delta(x-x') \tag{7}$$

The so-called Gaussian operators implement the linear transformations. On a set of a single (x, p) qumode, the rotation operator $\hat{R}$ acts between positions and momenta, while the displacement operator $\hat{D}$ performs the translations over the qumodes:

$$\hat{R}(\phi) : \begin{bmatrix}x\\p\end{bmatrix} \to \begin{bmatrix}\cos(\phi) & \sin(\phi)\\ -\sin(\phi) & \cos(\phi)\end{bmatrix}\begin{bmatrix}x\\p\end{bmatrix}, \quad \hat{D}(\alpha) : \begin{bmatrix}x\\p\end{bmatrix} \to \begin{bmatrix}x+\sqrt{2}\,\mathrm{Re}(\alpha)\\ p+\sqrt{2}\,\mathrm{Im}(\alpha)\end{bmatrix} \tag{8}$$

Together $\hat{R}$ and $\hat{D}$ are able to implement affine transformations on a single qumode. Another Gaussian transformation is given by the beamsplitter $\hat{BS}$, which is a 2-qumodes operator:

$$\hat{BS}(\theta) : \begin{bmatrix}x_1\\x_2\\p_1\\p_2\end{bmatrix} \to \begin{bmatrix}\cos(\phi) & \sin(\phi) & 0 & 0\\ -\sin(\phi) & \cos(\phi) & 0 & 0\\ 0 & 0 & \cos(\phi) & \sin(\phi)\\ 0 & 0 & -\sin(\phi) & \cos(\phi)\end{bmatrix}\begin{bmatrix}x_1\\x_2\\p_1\\p_2\end{bmatrix} \tag{9}$$

The last of the Gaussian operator is given by the squeezing one, $\hat{S}$:

$$\hat{S}(\tau) : \begin{bmatrix}x\\p\end{bmatrix} \to \begin{bmatrix}e^{-\tau} & 0\\ 0 & e^{\tau}\end{bmatrix}\begin{bmatrix}x\\p\end{bmatrix} \tag{10}$$

Defining $\hat{\mathbf{r}} = (\hat{\mathbf{X}}, \hat{\mathbf{P}})$, it is straightforward to state the uncertainty relation in the CV frame:

$$[\hat{r}_i; \hat{r}_j] = i\Omega_{ij}, \quad \Omega = \begin{pmatrix}\mathbb{O} & \mathbb{I}\\ \mathbb{I} & \mathbb{O}\end{pmatrix} \tag{11}$$

4.4. Embedding into QUBO problems

Adiabatic quantum computers can find the optimal solution to a specific class of optimization problems: Quadratic Unconstrained Binary Optimization (QUBO) problems. A QUBO problem is mathematically described as:

$$\hat{H}_P \equiv -\sum_{i=1}^{N} h_i\hat{\sigma}_i^z - \sum_{i<j} J_{ij}\sigma_i^z\hat{\sigma}_j^z\,, \tag{12}$$

where $\hat{\sigma}_i^z$ are the Pauli matrices that acting along the z-direction, and J_{ij} and h_i represent the parameters to the problem to be solved. We call J_{ij} couplings or weights and h_i biases. Since the eigenvalues of

the Hamiltonian H_P represent the possible solution to the problem, the goal is to set the couplings and the biases so that the ground state of H_P represents the optimal solution to the optimization problem.

The QUBO problems can be represented as graphs, where nodes are associated with biases and edges with couplings. Ideally, we want to map the optimization problem graph directly into the quantum annealer QPU. However, in a quantum annealing system, the hardware graph topology, which represents the pattern of physical connections for qubits and the couplers between them, is fixed. Since we cannot modify the qubit connectivity of a specific quantum annealer, we must map the model parameters into the hardware topology by a suitable embedding to solve an optimization problem. The basic idea of embedding is to identify groups of qubits (chains) so that they form the topology of the QUBO problem under investigation by behaving as individual units. The connectivity of each group can be enhanced by creating strong ferromagnetic couplings between the qubits, which forces coupled qubits to stay in the same state.

5. Processing quantum information with quantum computers

Three available architectures are provided in the field of quantum computing, see Fig. 1. Two out of three, the MBQC and the gate models, reproduce on a quantum device the classical Von Neumann-Zuse paradigm, i.e. processing the inputs into outputs via a sequence of commands. Such architectures can be labeled as circuital model, as both of them aim to install a circuit of logical, controlled and sequential operations on the qubits. On the contrary, the adiabatic computation provides a scheme of computation which is unedited for any classical device.

5.1. Circuital model

In the gate model, the logic gates are given by certain physical operations on the qubits. Such gates, apart from being described by unitary matrices in an algebraic fashion, can just be taught as the classical logic gates (OR, AND, XOR and so on) transposed in the quantum frame. Such logic transformations can involve just one single qubit, or rather two as well as n, see Fig. 2. Any logic gate can be built by composing a set of universal gates, generally given by two single-qubit and one two-qubit operators [1].

In the measurement-based model instead, such logic gates are built relying on quantum phenomena such as entanglement and measurement [7]. Nevertheless, the gate model can be mapped into the MBQC one [6,85], proving that the two models are able to yield the same output.

5.2. Adiabatic quantum model

Quantum annealers are quantum computers capable of finding the optimal solution to a QUBO problem by measuring the ground state of the QPU, i.e., the qubit configuration corresponding to the minimum energy of the system. The basic idea of quantum annealing is to prepare the qubits in the ground state, an easy-to-build configuration described by a Hamiltonian H_T, and then let the system evolve until it becomes equal to H_P, as in Eq. (12). If the evolution is sufficiently slow, the adiabatic theorem [3,86] guarantees that the systems stay in the ground state. Therefore, it is possible to find the solution to the optimization problem by simply measuring the system.

Quantum annealers realize quantum annealing by introducing a time-dependant transverse field resulting in a total Hamiltonian:

$$H(t) = -F(t)\underbrace{\left(\sum_{i<j} J_{ij}\sigma_i^z\sigma_j^z + \sum_i h_i\sigma_i^z\right)}_{H_P} - G(t)\underbrace{\sum_i \sigma_i^x}_{H_T} \tag{13}$$

where t represents the time, and the functions $F(t)$ and $G(t)$ control the annealing evolution and are referred to as the annealing schedule. At the beginning of the annealing, the system is prepared in the ground state of H_T ($G(t) \gg F(t)$). At the end of the annealing, the system should be in the ground state of H_P ($G(t) \ll F(t)$).

6. Emulation of quantum computing resources by high-performance computing

Quantum computing is a potentially disruptive computational paradigm that will enable efficient solutions to problems that are inherently difficult for classical digital devices. Although large-scale, error-corrected quantum computers are not yet available, hardware technology is evolving at a rapid pace, and demonstrations of quantum supremacy have already been achieved on current noisy intermediate-scale quantum (NISQ) devices for selected problems of academic interest [87–90]. From the practical applications standpoint, however, NISQ devices still need to operate alongside classical digital hardware. Real-world applications are in fact characterized by complex computational workflows and large problem instances in which most of the computational burden is necessarily carried by traditional resources. For these reasons, NISQ computers are currently utilized as accelerators or co-processors embedded in hybrid quantum–classical computation.

The orchestration of hybrid hardware resources is currently implemented by means of a loose-integration paradigm, in which the classical and quantum processing is typically performed via local machines with consumer capacities and via remote quantum devices respectively. From an end-user perspective, this paradigm is considered the most effective as it allows the evaluation of alternative vendor solutions while limiting the risks associated with the experimentation of highly prototypical technologies which might suffer from rapid obsolescence. The main drawbacks of such an approach are instead related to the latency associated with the continuous data transfer, along with the additional concerns regarding the exchange with third-parties servers of sensitive or restricted data.

To mitigate these issues, the scientific community is also starting to investigate on-premises scenarios with the co-location of quantum and digital classical hardware. This is commonly considered as a first step that, in the long term, should yield to a tight-integration paradigm in which quantum and classical processors are both co-located and interconnected via dedicated high-speed, high-capacity links [91]. The first experimentations in this direction are being conducted in various high-performance computing (HPC) centers, see at euroHPCJU and HPCQS.

Beyond providing the means for an effective exploration of hybrid quantum–classical integration paradigms,

HPC resources enable the emulation of quantum computers with up to the equivalent of 40 to 50 qubits, which is more than what most NISQ devices deliver today. Given that the current quantum hardware is still difficult and expensive to access, HPC emulators provide unique opportunities for conducting impactful R&D that would not be possible otherwise. Typical activities enabled by HPC emulators are test/development of new algorithms for real-world applications, evaluation of the solution quality and time-to-solution behavior when scaling up the size of the problem, and the investigation of co-design of quantum algorithms and hardware.

A variety of emulators are currently available that can be used to implement a range of quantum algorithms, including those that are presented in this review. Most of them target a qubit architecture that implements the gate-based computational setting, either with an exact quantum state representation (state-vector, density matrix) or with approximate/compressed state representation (tensor networks). Such emulating libraries implement standard linear algebra operations that emulate the behavior of physical gates operated on the qubits register. Some of the most known software development kits (SDKs) that provide emulator backends are Qiskit (IBM), Cirq (Google), and Pennylane [92] (Xanadu). The majority of such libraries, which have been initially written to run on local machines, are now capable to exploit distributed

memory protocols and GPU acceleration, which enable the simulation of intermediate-size quantum states (30+ qubits) and the execution of deeper circuits with acceptable running times.

In addition, as most of the hybrid quantum machine learning algorithms currently under investigation rely on variational circuits (VQE [93], QAOA [94], Quantum NN), which are parameterized circuits trained by a classical computer through the optimization of a differentiable loss function, many of these SDKs have also been designed or integrated with state-of-the-art machine learning software packages such as PyTorch [95], TensorFlow [96], Paddle-Paddle to leverage automatic differentiation techniques such as backpropagation. These can take advantage of GPU acceleration to reduce the overall execution time but incur additional memory overhead due to the need to store partial derivatives of the forward pass. The utilization of premium, large-memory GPUs that are typically available in HPC centers can boost performances.

Within the qubit architecture, a few emulators have been developed to deal with the measurement-based computational setting. Examples are Parceval (Quandela) [97] and Paddle-Quantum (Baidu), the latter taking also advantage of backpropagation methods. As for architectures based on qudits, Jet (Xanadu) [98] and Cirq (Google) libraries are available. Emulators based on continuous variable architectures are also available.

7. Generalization of neural networks in quantum circuits

The classes of Machine Learning and Deep Learning are often juxtaposed in literature and applications, but indeed the first includes the second as a special case based on neural networks. Deep learning exploits a deep hierarchy of layers of artificial rate neurons, resulting in a non-Von Neumann-Zuse architecture that is virtualized on standard digital CMOS hardware. A software tunes a set of hyperparameters of the NNs, called synaptic weights, to re-elaborate the inputs to outputs.

A ML approach based on traditional and classical computing is straightforward to be translated on a quantum hardware, as it suffices to encode the inputs and the prompts into a quantum circuit. The main bottleneck is the constraint on the current size of the hardware, but from a theoretical perspective the problem can be treated as any classical-to-quantum algorithm. In such a way, it is quite standard to benchmark the performances between any classical algorithm and its quantum counterpart. A comparison for some of the most known classical machine learning methods and routines is given in Table 1. Instead, neural network may require a paradigm shift towards new architectures. In the following two subsections, we present the two main approaches to neural networks, namely the variational and the quantum perceptron approaches, respectively.

7.1. Variational approach

As said, several algorithms rely on an Artificial Neural Network (ANN, or more simply NN). NNs allow to transform input data to outputs (labels, actions) encoding the inputs through various layers of artificial synapses. According to the Hornik's theorem [99], a sufficiently complex NN can always approximate the label output given an input.

In quantum computing, many models have been proposed to replace the classical architecture of multiperceptron-based neural networks. The main feature consists of introducing a specific circuit model, able to process the input states of the system through a series of iterations. Of course, shallow circuits belong to this architecture, as well as one-layer classical neural networks. In order to process the information, instead of layers of neurons, quantum circuits display a block of unitary operations to be performed. The angles, implemented by unitary operators such as rotations, substitute the synaptic weights. In this frame, a quantum neural network (QNN) can be implemented via a variational quantum circuit (VQC) [100,101].

However, a broad range of quantum neural networks models have been proposed [102–106]: hybrid quantum circuit-classical neural networks approaches [107], quantum neuron models [108] as an alternative for the classical activation functions and so on. A classical activation function f is defined as

$$f(\mathbf{x}) = \sigma\left(\sum_j W_{ij} x_j + b_i\right) \tag{14}$$

where W is called the weight function, and $\mathbf{b}$ the bias vector. There are several classes of σ activation functions, among them the perceptron, the sigmoid, the ReLu and others. The perceptron, in its classical description, given a set of N inputs acts as a step activation function [109][2, pag.49], i.e. σ is substituted by the Heaviside step function θ, or rather by a sign function. In Section 8.1 a model for a quantum perceptron is presented.

It must be noted that variational circuits are known for suffering the barren plateau effect [110]. As random circuits are often proposed as initial guesses for exploring the space of quantum states during optimization of the parameters, one discovers that the exponential dimension of Hilbert space and the gradient estimation complexity make such choice unsuitable on more than a few qubits. In general the probability that the gradient along reasonable directions is non-zero to some fixed precision is exponentially small as a function of the number of qubits. Mitigations to this issue have been proposed thanks to smart initialization [111], also avoiding it thanks to quantum convolutional neural networks [112].

7.2. Neurons into qubits

There are more options to encode neurons into a qubit. Nevertheless, given the definition of a qubit, few encoding options can be defined, bounded by the maximum encoding capability of a register of N qubits. The first option consists of the 1-*to*-1 encoding, where each and every input neuron of the network corresponds to one qubit [113–117]. The information is provided as a string of bits assigned to classical base states of the quantum state space. Similarly, a 1-*to*-1 method consists of storing a superposition of binary data as a series of bit strings in a multi-qubit state. Such quantum neural networks refer to the concept of the quantum associative memory [118,119]. A different 1-*to*-1 option is given by the quron (quantum neuron) [120]. A quron is a qubit whose 0 and 1 states stand for the resting and active neural firing state, respectively [120].

Alternatively, a radically different encoding option consists of storing the information as coefficients of a superposition of quantum states [21,109,121–124]. The encoding efficiency becomes exponential as a n-qubit state is an element of a 2^n-dimensional vector space, but one has to remember that also operations required to store the state increase exponentially. From one hand, loading a real image classification problem of few megabits in a quantum neural network makes the 1-*to*-1 option currently not viable [125], while the choice n-*to*-2^n allows to encode a megabit image in a state by using ~ 20 qubits only. In the latter case one should anyway deal with the difficulty of preparing such a state, unless it is for instance generated by a circuit which approximates some aimed distribution, or alternatively it comes directly from the physical conversion of flying qubits containing quantum data acquired by some quantum sensing system.

8. Quantum supervised learning

Supervised Learning is the branch of Machine Learning which has been more transposed in a quantum formulation. Here we present the most significant quantum algorithms relevant for cybersecurity tasks, along with a description of their classical counterparts: activation functions for binary decisions [21,123,126], Support Vector Machine (SVM) [33,127–132] and kernel methods in general [74]. One should notice that while current cybersecurity data are fundamentally classical

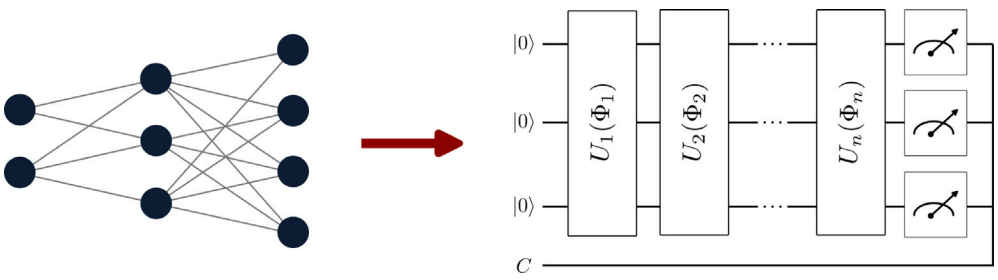

Fig. 2. In classical deep learning, neural networks are employed for data processing. In quantum computing, circuits replace the typical neural architecture. The blocks of unitary operators $\hat{U}_i(\boldsymbol{\Phi}_i)$ are the counterparts for the classical layers, the $\boldsymbol{\Phi}_i$ variables are the hyperparameters, which have to be tuned in order to minimize a certain loss function.

in nature, in the future incoming quantum data from either quantum sensors or quantum communication networks may carry quantum (entangled) data, which in turn can be classified for instance by quantum tensor networks, as demonstrated by one of the authors [133]. In the broader field of classical anomaly detection, a paramount role is played by image classification, e.g. to spot medical diseases [59,134,135] or mechanical defects in industrial processes [60,61,136], therefore a specific Section of our analysis is dedicated to their quantum counterpart. Classification by quantum tensor network on reduced MNIST with 4 categories has shown to return the same performances as best supervised learning algorithms, but more interestingly, it was able to discriminate quantum ground states carrying entanglement. In the following Section, we introduce a range of techniques which vary from the implementation of natively quantum perceptron to the employment of adiabatic computation to improve performances of the Restricted Boltzmann Machine. Moreover, we show how to encode quantum neural networks on the continuous variables and classification tasks on qudits. All of the aforementioned techniques performs well when dealing with sampling from probabilistic distributions. Eventually, we show how to achieve quantum advantage on the Support Vector Machines via the HHL algorithm.

8.1. Quantum feed-forward neural network for binary decisions

Recently [123,126], a model of perceptron implemented by a quantum circuit has been proposed. This model features an input vector $\vec{i} = (i_0, \dots, i_{m-1})$ and a weight vector $\vec{w} = (w_0, \dots, w_{m-1})$, such that the activation response depends on their $\vec{i} \cdot \vec{w}$ scalar product. In such scenario, the components of the vectors $i_k, w_k = \pm 1 \, \forall k$. The vectors can be encoded into a quantum register by the following states:

$$|\psi_i\rangle = \frac{1}{\sqrt{m}} \sum_{j=0}^{m-1} i_j |j\rangle, \qquad |\psi_w\rangle = \frac{1}{\sqrt{m}} \sum_{j=0}^{m-1} w_j |j\rangle \tag{15}$$

where the state $|j\rangle$ belongs to $\{|0 \dots 00\rangle, |0 \dots 01\rangle, \dots, |1 \dots 11\rangle\}$. Being i_k, w_k all ± 1, and m being the dimension of the input and weight vectors, $|\psi_i\rangle$ and $|\psi_w\rangle$ are real equally-weighted (REW) superpositions of all the computational basis states $|j\rangle$. The states $|j\rangle$ live in a $\mathcal{H}^{\otimes N}$ Hilbert space, where $N = \log_2(m)$.

The inner product between $|\psi_i\rangle$ and $|\psi_w\rangle$ returns $\vec{i} \cdot \vec{w}$ times m. To prepare the input state, the following transformation can be implemented as

$$\hat{U}_i |0\rangle^{\otimes N} = |\psi_i\rangle \tag{16}$$

where $\hat{U}_i$ can be composed by any $m \times m$ matrix with $\vec{i}$ on the first column [21]. To perform the $\vec{i} \cdot \vec{w}$ inner product, it is possible to define a unitary operator $\hat{U}_w$ such that $\hat{U}_w |\psi_w\rangle = |1\rangle^{\otimes N} = |m-1\rangle$. To do so, choose any unitary $m \times m$ with $\vec{w}$ on the last row. The inner product between $|\psi_i\rangle$ and $|\psi_w\rangle$ can thus be performed by

$$\langle \psi_w | \psi_i \rangle = \langle \psi_w | \hat{U}_w^\dagger \hat{U}_w | \psi_i \rangle = \langle m-1 | \psi_{i,w} \rangle = c_{m-1} \tag{17}$$

where $|\psi_{i,w}\rangle = 1/\sqrt{m} \sum_{j=0}^{m-1} c_j |j\rangle$ is simply $\hat{U}_w |\psi_i\rangle$. Therefore, c_{m-1} yields the scalar product $(\vec{i} \cdot \vec{w})m$. The coefficient c_{m-1} can be obtained, in a circuital computation, by entangling $|\psi_{i,w}\rangle$ with an ancilla $|0\rangle$, through a multi CNOT gate (i.e. a CNOT whose control qubits are given by the $|j\rangle$ state). The sole state on which such multi-CNOT operator $\hat{C}_{|j\rangle,m}$ acts on is the $|1\rangle^{\otimes(m-1)}$ state, i.e. the last one ($|m-1\rangle$):

$$\hat{CX}_{|m-1\rangle,|m\rangle} |\psi_{i,w}\rangle |0\rangle = \frac{1}{\sqrt{m}} \left[\sum_{j=0}^{m-2} c_j |j\rangle |0\rangle + c_{m-1} |m-1\rangle |1\rangle \right] \tag{18}$$

where $|m-1\rangle$ is the multi-qubits control state and $|m\rangle$ the target one. Measuring the ancilla qubit on the $|1\rangle$ basis, it is possible to activate the perceptron with probability $|c_{m-1}|^2$. Such achievement reproduces the perceptron in a quantum circuit. One should notice that such activation function ends the circuit with the measurement process, so the quantum information cannot travel further to other nodes. The issue has been addressed by one of the Authors [137] by replacing the measurement process with a quantum circuit performing the Taylor series of the aimed activation function. Such method enables to program a multilayered perceptron.

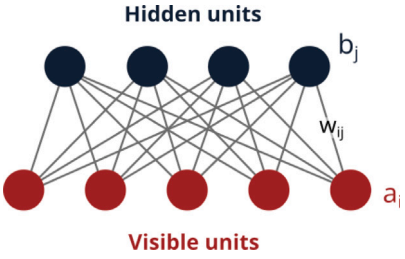

Fig. 3. A graphical representation of a Restricted Boltzmann Machine. Each undirected edge represents a weighted dependency between two nodes, while each node is associated with a bias. The network has four hidden units (blue) and six visible units (red).

8.2. Quantum restricted Boltzmann machine

Restricted Boltzmann Machines (RBMs) are neural network generative models first introduced by Hinton et al. in 1983 to improve upon the Hebbian learning method used in Hopfield networks. These models are designed to learn the underlying probability distributions of a dataset by using the Boltzmann distribution in their sampling function. A RBM consists of two layers: a layer of visible binary units (representing the input/output) and a layer of hidden binary units (which help the model mimic the dataset's structure). The units are connected by real-weighted connections, as illustrated in Fig. 3. RBMs do not allow connections between units within the same layer, resulting in a bipartite system.

RBMs are flexible neural network models that can be used for various tasks, such as generating samples, making recommendations, or extracting features. They can also be used as classifiers by using different techniques, such as using them as feature extractors and

appending a separate classifier or training them supervised with the label appended to the input data. These supervised RBMs are called discriminative restricted Boltzmann machines (DRBMs), which combine descriptive power with classification ability. The idea is to train the DRBM with a dataset where the label is appended to the input, then remove it from unseen inputs and reconstruct it using the RBM.

The RBM is an energy-based model where every specific configuration of visible and hidden units is associated with an energy $E(\mathbf{v},\mathbf{h}) = -\sum_i a_i v_i - \sum_j b_j h_j - \sum_{i,j} v_i W_{ij} h_j$, where $\mathbf{a}$, $\mathbf{b}$ are biases and $\mathbf{W}$ are the weights that represent the connection strength between units. Specifically, the joint probability of a configuration is given by the Boltzmann distribution

$$P(\mathbf{v},\mathbf{h}) = \frac{\exp(-E(\mathbf{v},\mathbf{h}))}{\sum_{\mathbf{v},\mathbf{h}} \exp(-E(\mathbf{v},\mathbf{h}))}. \tag{19}$$

The objective of training an RBM is to adjust the model's weights to increase the energy of states in the training dataset and lower the energy of all other configurations, allowing the model to learn how to generate and reconstruct the critical information encoded in the dataset. However, training an RBM can be challenging due to the large number of states that increases exponentially with the number of visible and hidden units, making it impractical to compute the partition function. Although an exact computation is not possible, several classical methods can be used to train the model, such as Contrastive Divergence [138] (CD), Persistent Contrastive Divergence [139] (PCD), and Lean Contrastive Divergence [140] (LCD).

Although these methods are effective in practice, RBMs can be more difficult and costly to train than other models that rely on backpropagation techniques, such as neural networks. Their training is often unstable and requires significant computational resources, and the approximations made during training can affect overall performance. Quantum computers provide an alternative approach to training RBMs, allowing for faster computation and better gradient estimates by querying the quantum processing unit. D-Wave quantum annealers, which are commonly used to sample the ground state of a QUBO problem, can also be used to train RBMs, resulting in faster computation and a better gradient estimate. These RBMs trained on a quantum annealer are called Quantum Restricted Boltzmann Machines (QRBMs).

The basic idea to train a RBM on a quantum computer [141–143] is to extract a batch of samples from the quantum machine, which are dispersed according to the Boltzmann distribution associated to the RBM. If the computational cost of initializing the quantum computer is neglected, the quantum algorithm computational complexity to obtain a single sample scales as $O(1)$. The advantage of employing the D-Wave adiabatic quantum machine to exploit RBMs could emerge as an increase of performance metrics, such as the accuracy and the likelihood, or as a reduction in the computational complexity or computational times depending on the specific problem under consideration. The quantum RBM has been used to address anomaly detection of IP traffic data, performing 64x faster than classic hardware in the inference [24]. More general machines called Boltzmann machines, based on a complete (not bipartite) graph, have also been addressed on an adiabatic quantum computer [144].

8.3. Neural networks in continuous variables

An architecture to set up a Neural Network (NN) by continuous variables (CV) has been provided by Killoran et al. [10]. It is shown that through the gates of CV encoding it is possible to reproduce the classical layer for a NN:

$$\mathcal{L}(\mathbf{x}) = \varphi(W\mathbf{x} + \mathbf{b}) \tag{20}$$

where φ is the activation function, W is the weight matrix and $\mathbf{b}$ is the bias vector. Such layer can be embedded in the CV formalism via the following sequence of operators/logic gates:

$$\hat{\mathcal{L}} = \hat{\Phi}\circ\hat{D}\circ\hat{\mathcal{U}}_2\circ\hat{S}\circ\hat{\mathcal{U}}_1 \tag{21}$$

Here $\hat{D} = \bigotimes_{i=1}^N \hat{D}_i(\alpha_i)$, $\hat{S} = \bigotimes_{i=1}^N \hat{S}_i(\tau_i)$, where $\hat{D}$ and $\hat{S}$ are the Gaussian operators in Section 4.3.1, while the $\hat{\mathcal{U}}_i$ operators are given by a composition of beamsplitter $\hat{BS}(\theta)$. Instead, $\hat{\Phi}$ is a new non-Gaussian operation we are going to define in this Section. A depiction of such quantum circuit is provided in Fig. 4. To perform Machine Learning tasks in CV, is therefore possible to implement a variational circuit built by a set of layers such as in Eq. (21). In the following, it is shown how a quantum neural network has been built in Ref. [10]. The first three operations can be decomposed into a direct sum of two blocks:

$$\begin{aligned}\hat{\mathcal{U}}_2\circ\hat{S}\circ\hat{\mathcal{U}}_1 &= \begin{bmatrix} M_2 & \mathbb{O} \\ \mathbb{O} & M_2 \end{bmatrix}\begin{bmatrix} \Sigma & \mathbb{O} \\ \mathbb{O} & \Sigma^{-1} \end{bmatrix}\begin{bmatrix} M_1 & \mathbb{O} \\ \mathbb{O} & M_1 \end{bmatrix} \\ &= \begin{bmatrix} M_2\Sigma M_1 & \mathbb{O} \\ \mathbb{O} & M_2\Sigma^{-1}M_1 \end{bmatrix}\end{aligned} \tag{22}$$

where the first block on the diagonal acts over the $\hat{\mathbf{x}}$ variables, the second one over $\hat{\mathbf{p}}$, in a similar fashion as the beamsplitter operator in Eq. (9). Afterwards, it is possible to apply the shifting by the displacement operator, so that the initial state $|\mathbf{x}\rangle$, up to this point, is morphed into

$$|\mathbf{x}\rangle \to \hat{\mathcal{L}}|\mathbf{x}\rangle = |M_2\Sigma M_1\mathbf{x} + \sqrt{2}\boldsymbol{\alpha}_r\rangle \tag{23}$$

where $\boldsymbol{\alpha}_r = \mathrm{Re}(\boldsymbol{\alpha})$. The next step is to implement a non-linear transformation, which is given by the non-Gaussian operations $\hat{\Phi}$. To build up such single-qumode gate, define a non-linear transformation $\phi(x)$, which can be written in a Taylor expansion of $\hat{X}$ for a certain degree of approximation, thereafter implement the operation in the form

$$\exp\left(-\frac{i}{2}\phi(\hat{X}_1)\otimes\hat{P}_2\right)|x\rangle_1|0\rangle_2 = \exp\left(-\frac{i}{2}\phi(x)\hat{P}_2\right)|x\rangle_1|0\rangle_2 \tag{24}$$

Now the unitary operation is nothing but a translation over the second qubit:

$$\exp\left(-\frac{i}{2}\phi(x)\hat{P}_2\right)|x\rangle_1|0\rangle_2 = |x\rangle_1|\phi(x)\rangle_2 \tag{25}$$

Eventually, a similar operation as from Eqs. (14) and (20) has been implemented in the framework of Continuous Variables, as stated in Eq. (20):

$$\hat{\mathcal{L}}(\mathbf{r}) = \phi(M\mathbf{r} + \sqrt{2}\boldsymbol{\alpha}) \tag{26}$$

where $M = M_2\Sigma M_1 \oplus M_2\Sigma^{-1}M_1$ (see Fig. 4).

8.3.1. CV neural networks for fraud detection

From the formalism in Section 8.3, it is possible to develop a hybrid algorithm, where it is possible to alternate classical Neural Network with quantum circuits for Supervised Learning tasks. As both the hyperparameters from the classical NNs and the variational circuit need to be tuned, a backpropagation can be performed by training the data on a classical device. In Ref. [10], a mean square error (MSE) was introduced as loss function:

$$L = \frac{1}{N}\sum_{i=1}^{N}[f(x_i) - \langle\psi(x_i)|\hat{X}|\psi(x_i)\rangle]^2 \tag{27}$$

The model, when correctly trained, should return $\langle\psi(x)|\hat{X}|\psi(x)\rangle = f(x) \quad \forall x$. In [10], such model of hybrid Supervised Learning was employed to detect fraudulent transactions. The performance of the training outputted a ROC curve with area 0.945, whereas the ideal curve should return a unitary area.

8.3.2. Potential advantages of CV neural networks

In the implementation of Killoran et al.. [10], three potential advantages are proposed for employing the CV quantum neural networks. In the first place, CV neural networks can be performed on any photonic device, as the operation to implement them are universal in the photonic technology platform.

Secondly, Hornik's theorem, as explained in Section 7.1, guarantees that any Lebesgue measurable function can be reproduced by a neural

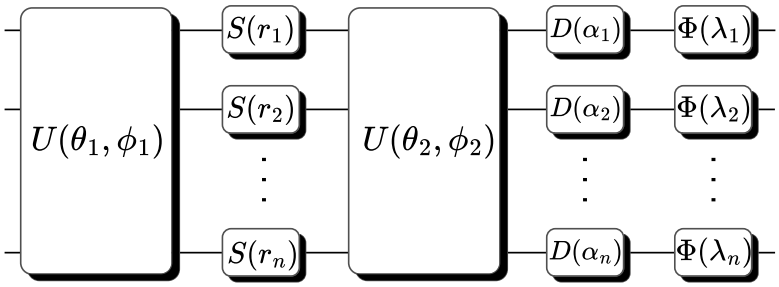

Fig. 4. The picture represents how to build up a layer $\mathcal{L}$ in the frame of Continuous Variables, following the scheme from Eq. (21).

network. Quantum neural networks embed this property along with effects such as superposition and entanglement, which are intrinsic of the quantum realm. Furthermore, dealing with qumodes $(\mathbf{x}, \mathbf{p})$ it is possible to rely on both the positions and momenta representations, $\mathbf{x}$ and $\mathbf{p}$ being the Fourier transform of each other.

As the last point, quantum neural network in the CV framework can be employed for nonlinear transformations over distributions of probability. For instance, given a single-mode state $|\psi\rangle$, it is possible to encode its amplitude and transform it via a unitary transformation $\mathcal{W}$, due to the transformations acting inside the layers in Eq. (26):

$$|\psi\rangle = \int \psi(x)\mathrm{d}x \Rightarrow \tilde{\psi}(x) = \int \mathcal{W}(x, x')\psi(x')\mathrm{d}x' \tag{28}$$

8.4. Classification with qudits

Given a finite set of vectors x in $\mathbb{R}^m$, partitioned between D classes, the Density Matrix Kernel Density Classification method (DMKDC) is an algorithm developed by Useche et al. [22] which aims to reproduce the probability functions P_j for an element x to belong to a certain class $j \in \{0, \dots, D-1\}$.

Given a system of qudits in a $\mathbb{C}^d$ space, suppose to have a number of classes $D \leq d$. Thereafter, a collection of training data $\{x_i\}_{i=1}^N \subseteq \mathcal{X}$ is provided, along with a feature map $\psi : x_i \to |\psi(x_i)\rangle$. Such a map can be set by a softmax encoding, see Ref. [145], rather than via random Fourier features (RFF), see Ref. [145,146]. Both of these encodings would provide a normalized vector such that $\langle\psi(x)|\psi(x)\rangle = 1$, $|\psi(x)\rangle$ being encoded in the fashion of a qudit vector, as in Eq. (4), whose coefficients are computed by the chosen encoding methods. In second place, the density matrix $\hat{\rho}$, associated to such states, is constructed as a maximally mixed state over all the samples, see the below Equation. At the same time, it is possible to define a specific density matrix ρ_j corresponding to each jth class:

$$\hat{\rho} = \frac{1}{N}\sum_{i=1}^{N}|\psi(x_i)\rangle\langle\psi(x_i)|, \qquad \hat{\rho}_j = \frac{1}{N_j}\sum_{i=1}^{N_j}|\psi(x_i)\rangle\langle\psi(x_i)| \tag{29}$$

N being the cardinality of the entire $\{x_i\}_{i=1}^N$ dataset. The frequency $\pi_j = N_j/N$ accounts how many times a data x_j belongs to the jth class, N_j counting the number of samples into the jth class. The posterior probability for a generic x sample to belong to the jth class reads as [145]

$$P_j(x) = \frac{\pi_j\langle\psi(x)|\hat{\rho}_j|\psi(x)\rangle}{\sum_{k=0}^{D-1}\pi_k\langle\psi(x)|\hat{\rho}_k|\psi(x)\rangle} \tag{30}$$

The aim of the algorithm is to sample $\langle\psi(x)|\hat{\rho}_k|\psi(x)\rangle$ $\forall k$ in order to get the probability P_j. As the $|\psi(x_k)\rangle$ are known from the data, and $\hat{\rho}_j$ being a Hermitian operator, it is possible to diagonalize it via a $\hat{U}_j$ transformation:

$$\hat{\rho}_j = \hat{U}_j\hat{\Lambda}_j\hat{U}_j^\dagger = \hat{U}_j\left(\sum_{i=0}^{D-1}\lambda_{ji}|i\rangle\langle i|\right)\hat{U}_j^\dagger \tag{31}$$

The λ_{ji} are the ith eigenvalues for the $\hat{\rho}_j$ operators, i and j ranking from 0 to $D-1$. For each operator $\hat{\rho}_j$, there exists a specific $\hat{U}_j$ unitary transformation capable to diagonalize the density matrix $\hat{\rho}$ into the computational basis $|i\rangle$, where $i = 0, \dots, D-1$. The expectation $\langle\psi(x)|\hat{\rho}_j|\psi(x)\rangle$ can therefore be written as

$$\begin{aligned}\langle\psi(x)|\hat{\rho}_j|\psi(x)\rangle &= \langle\psi(x)|\hat{U}_j\left(\sum_{i=0}^{D-1}\lambda_{ji}|i\rangle\langle i|\right)\hat{U}_j^\dagger|\psi(x)\rangle \\ &= \sum_{i=0}^{D-1}\lambda_{ji}\left|\langle i|\hat{U}_j^\dagger|\psi(x)\rangle\right|^2\end{aligned} \tag{32}$$

Afterwards, it is possible to introduce the $\hat{U}_{\lambda_j}$ operator:

$$\hat{U}_{\lambda_j}|0\rangle = \sum_{i=0}^{D-1}\sqrt{\lambda_{ij}}|i\rangle \tag{33}$$

Before to show the qudit implementation of the DMKDC circuit, we sum up the pipeline for the training of the training process based on the density matrix estimation:

1. map the data $x_i \in \mathcal{X}$ into a qudit vector, thanks to a RFF or a softmax encoding;
2. sample the π_j frequencies, thus estimating the $\hat{\rho}_j$ probability densities;
3. introduce the $\hat{U}_j$ operator able to diagonalize the $\hat{\rho}_j$ observables.

It is worthy to notice that such training procedure does not involve iterative operations. The training samples are solely employed to prepare the $\hat{\rho}$ matrices, the time complexity of the algorithm scaling linearly on the size of training dataset. In fact, as remarked by Gonzalez [145], the complexity of the algorithm is $O(N D^2)$ for the estimation of the $\hat{\rho}_j$ elements, N being again the cardinality of the dataset, and $O(D^3)$ for the diagonalization of the same probability densities (see Fig. 5).

8.4.1. The DMKDC circuit

To prepare the DMKDC register, in the first place a qudit with all the frequencies π_j is prepared as follows:

$$|\pi\rangle = \sum_{i=0}^{D-1}\sqrt{\pi_i}|i\rangle \tag{34}$$

In the second place, a qudit $|\psi(x)\rangle$ encodes the classical data to be classified, at last a $|0\rangle$ ancilla. The overall state, before to compute, results in

$$|\pi\rangle \otimes |\psi(x)\rangle \otimes |0\rangle = \left(\sum_{i=0}^{D-1}\sqrt{\pi_i}|i\rangle\right) \otimes |\psi(x)\rangle \otimes |0\rangle \tag{35}$$

As a first step, apply a $\hat{X}$ gate on the $|\pi\rangle$ qudit, which consists of a sum modulo D over the $|i\rangle$ generators, so that the new state reads

$$\left(\sum_{i=0}^{D-1}\sqrt{\pi_i}|i \oplus 1\rangle\right) \otimes |\psi(x)\rangle \otimes |0\rangle \tag{36}$$

where $\oplus$ is the sum modulo D. Afterwards, apply a $C\hat{U}_0^\dagger$ and a $C\hat{U}_{\lambda_0}$ gates, with the first qudit as control and the second and third

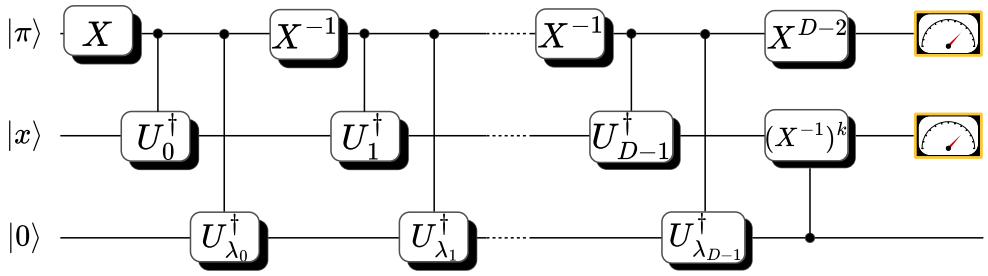

Fig. 5. A scheme for the DMKDC circuit.

respectively as targets:

$$\sqrt{\pi_0}|1\rangle \otimes \hat{U}_0^\dagger|\psi(x)\rangle \otimes |\lambda_0\rangle + \left(\sum_{i=1}^{D-1} \sqrt{\pi_i}|i \oplus 1\rangle\right) \otimes |\psi(x)\rangle \otimes |0\rangle \tag{37}$$

where the first controlled gate apply a $\hat{U}_0$ on the second qudit, while the second c-gate morphs the $|0\rangle$ qudit into $|\lambda_0\rangle$. Applying back a $\hat{X}^{-1}$ gate, the state turns to be

$$\sqrt{\pi_0}|0\rangle \otimes \hat{U}_0^\dagger|\psi(x)\rangle \otimes |\lambda_0\rangle + \left(\sum_{i=1}^{D-1} \sqrt{\pi_i}|i\rangle\right) \otimes |\psi(x)\rangle \otimes |0\rangle \tag{38}$$

As a second step, from the above state apply the $C\hat{U}_1^\dagger$ and the $C\hat{U}_{\lambda_1}$ gates, which outputs

$$\sqrt{\pi_0}|0\rangle \otimes \hat{U}_0^\dagger|\psi(x)\rangle \otimes |\lambda_0\rangle + \sqrt{\pi_1}|1\rangle \otimes \hat{U}_1^\dagger|\psi(x)\rangle \otimes |\lambda_1\rangle + \left(\sum_{i=2}^{D-1} \sqrt{\pi_i}|i\rangle\right) \otimes |\psi(x)\rangle \otimes |0\rangle \tag{39}$$

To iterate the process, apply the $\hat{X}^m$ gate on the first qudit, thereafter the $C\hat{U}_m^\dagger$ and $C\hat{U}_{\lambda_m}$ gates, for $m = 2, \dots, D-1$. Eventually the circuit returns the following state:

$$\sum_{i=0}^{D-1} \left(\sqrt{\pi_i}|i\rangle \otimes \hat{U}_i^\dagger|\psi(x)\rangle \otimes |\lambda_i\rangle\right) \tag{40}$$

The second and the third qudits can be rewritten as

$$\begin{aligned}\hat{U}_i^\dagger|\psi(x)\rangle \otimes |\lambda_i\rangle &= \sum_{l=0}^{D-1} |l\rangle\langle l|\hat{U}_i^\dagger|\psi(x)\rangle \otimes \sum_{m=0}^{D-1} \sqrt{\lambda_{mi}}|m\rangle \\ &= \sum_{l=0}^{D-1} a_{li}|l\rangle \otimes \sum_{m=0}^{D-1} \sqrt{\lambda_{mi}}|m\rangle\end{aligned} \tag{41}$$

where the a_{li} coefficients are $\langle l|\hat{U}_i^\dagger|\psi(x)\rangle$. It is possible to recombine the tensor product coupling the diagonal terms and the off-diagonal apart:

$$\sum_{l=0}^{D-1} a_{li}\sqrt{\lambda_{li}}|ll\rangle + \sum_{\substack{m,l=0 \\ m\neq l}}^{D-1} a_{li}\sqrt{\lambda_{mi}}|lm\rangle \tag{42}$$

Applying the $C(\hat{X}^{-1})^k$ gate using the second qudit as target and the third as control, it leads to

$$\sum_{i=0}^{D-1} \sqrt{\pi_i}|i\rangle \otimes \left(\sum_{l=0}^{D-1} a_{li}\sqrt{\lambda_{li}}|0l\rangle + \sum_{\substack{m,l=0 \\ m\neq l}}^{D-1} a_{li}\sqrt{\lambda_{mi}}|(l-m)m\rangle\right) \tag{43}$$

The overall scheme of the circuit is reported in Fig. 5. The probability P_j in Eq. (30) can be achieved by measuring the first qudit in the jth element and the second one in $|0\rangle$:

$$P_{j0} = \pi_j \sum_{l=0}^{D-1} \left|a_{lj}\right|^2 \lambda_{lj} = \pi_j \langle\psi(x)|\hat{\rho}_j|\psi(x)\rangle \tag{44}$$

The second passage has made usage of Eq. (32). At the end of the process, in the phase of testing, it is possible to point out which class the data x belongs to by maximizing the probability:

$$\max_j [\pi_j \langle\psi(x)|\hat{\rho}_j|\psi(x)\rangle] \tag{45}$$

8.5. Classical and quantum support vector machines

In the field of quantum machine learning, support vector machines (SVM) have been deployed for instance to distinguish anomalies from normal activities. More specifically, such algorithms has been employed to spot fraudulent credit card transactions or spurious bank loan [28], to address malware detection [147] or rather to prevent cyber attacks, such as DDoS attacks [27]. Support Vector Machines are a classical supervised learning algorithm which aims to learn from the training samples $\mathbf{x}_i$ in order to classify a new data sample into positive or negative class [148]. The data samples are given in the form $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_N, y_N)\}$, with say two possible classes A and B and a relation to satisfy given by [149]

$$\begin{cases} y_k = +1 & \text{if } x_k \in A \\ y_k = -1 & \text{if } x_k \in B \end{cases} \tag{46}$$

The space where the data $\mathbf{x}_i$ are set is $\mathbb{R}^d$. In such a formulation, it is possible to define a new set of coordinates $\mathbf{z}$ such that $\phi(\mathbf{x}) = \mathbf{z}$. The ϕ are called feature maps, mapping the $\mathbf{x}_i$ to the space of the $\mathbf{z}$, i.e. $\phi : \mathbb{R}^d \to \mathbb{R}^M$, with $M \geq d$. The purpose is to set a hyper-plane, given by the $\mathbf{w}\cdot\mathbf{z} + b = 0$ equation, where $\mathbf{z}$ are the generic coordinates in a $\mathbb{R}^M$ space and $\{\mathbf{w}, b\}$ define the parameters for the hyperplane. The vector $\mathbf{w}$ of parameters is defined as

$$\mathbf{w} = \sum_{i=1}^{d} \alpha_i \phi(\mathbf{x}_i) \tag{47}$$

where $\mathbf{x}_i$ are the data for the training, and α_i their corresponding weights.

For the classification to succeed, at the end of the training $\mathbf{w}$ and $\mathbf{b}$ should be set such that $\mathbf{w}\cdot\mathbf{z} + b \geq 1$ for a training point $\mathbf{x}_i$ in the positive class, and $\mathbf{w}\cdot\mathbf{z} + b \leq -1$ for a training point $\mathbf{x}_i$ in the negative class. Via the formulation in Eq. (47), the hyperplane $D(\mathbf{x}) = 0$ can be defined as [130]

$$\begin{aligned} D(\mathbf{x}) &= \sum_{i=1}^{d} \alpha_i \phi(\mathbf{x}_i)\cdot\mathbf{z} + b \\ &= \sum_{i=1}^{d} \alpha_i \phi(\mathbf{x}_i)\cdot\phi(\mathbf{x}) + b = \sum_{i=1}^{d} \alpha_i k(\mathbf{x}_i, \mathbf{x}) + b \end{aligned} \tag{48}$$

where $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$ is called kernel function.

8.5.1. Kernel models

In Eq. (48), the kernel function is defined as the inner product between feature maps, but many other definitions may arise. Linear kernels are defined as [34,132,150]

$$k(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)\cdot\phi(\mathbf{x}_j) \tag{49}$$

In such case, d and M, the dimensions of the data and the feature space, are equal. Nevertheless, many models of different kernels may arise. Depending on the nature of the problem to be tackled, different kernels may induce different metrics for the classification tasks. The polynomial kernel is defined as [33,34,131] $k_s(\mathbf{x}_i, \mathbf{x}_j) = (\lambda\phi(\mathbf{x}_i)\cdot\phi(\mathbf{x}_j) + r)^s$, where s is the polynomial degree and λ, r are constants to be tuned.

Another class is given by the gaussian kernel [33,131,150], $k_g(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\phi(\mathbf{x}_i)-\phi(\mathbf{x}_j)\|^2}{2\sigma^2}\right)$, where σ is again a constant to be tuned. The last kernel model frequently cited in literature is the Radial Basis Function kernel (RBF) [34], $k_{RBF}(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\gamma\|\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j)\|^2\right)$. Again, γ is a parameter to tune for best fitting the real model.

In any of these formulations, K_{ij} still remains a symmetric matrix.

8.5.2. From hard to soft margins

Regardless of the choice for the kernel, the final goal of the algorithm should be to reproduce the function in Eq. (46) and Ref. [127]:

$$y(\mathbf{x}) = \text{sgn}\,(D(\mathbf{x})) = \text{sgn}\left(\sum_{i=1}^{m} \alpha_i k(\mathbf{x}_i, \mathbf{x}) + b\right) \tag{50}$$

and therefore, the hyperparameters which need to be trained are now $\{b, \vec{\alpha}\}$. A way to express the affiliation of a $\mathbf{x}_i$ data to one of the two classes, for $\phi(\mathbf{x}) = \mathbf{x}$ (ϕ thus being the identity map) is the following [18]:

$$(\mathbf{w} \cdot \mathbf{x}_i + b) y_i \geq 1 \tag{51}$$

With such a formulation, the closest data $\mathbf{x}_i$ to the hyperplane yield an equation $\mathbf{w} \cdot \mathbf{x}_i + b = \pm 1$. In Appendix B, we prove $2/\|\mathbf{w}\|$ to be the distance between such points, thus calling $\|\mathbf{w}\|$ to be minimized for the classification to succeed at best. However, some data $\mathbf{x}_i$ may fall into a so-called grey region, with distance ε_i with respect to the corresponding hyperplane. The ε_i can be thought as errors, or soft-variables (because the margins of the hyperplanes are now "soft"). The condition in Eq. (51) can be translated into an equality:

$$(\mathbf{w} \cdot \mathbf{x}_i + b) y_i = 1 - \varepsilon_i \;\Rightarrow\; (\mathbf{w} \cdot \mathbf{x}_i + b) = y_i(1 - \varepsilon_i) \tag{52}$$

y_i being ± 1. We will refer to this condition in the next steps. Being $\mathbf{w}$ the normal vector to the hyperplane with coordinates $\mathbf{x}$, the distance between the two separation hyperplanes for the two classes is given by $\mathbf{w}^T \cdot \mathbf{w}$, i.e. by the norm of $\mathbf{w}$. Choosing the direction of $\mathbf{w}$, it is possible to minimize the distance between the two regions in the phase space which define the two classes, reducing therefore the probability to find an error ε_i. The purpose is now to minimize such distance, so that any object falling in the between of the two planes can be classified with no ambiguity. Such geometrical deduction can be pursued in Fig. 6(a). Nevertheless, when some outliers inevitably occur, as in Fig. 6(b), we are even interested in reducing the ε_i total distance. Summing these conditions with the constraint in Eq. (52), the following system is provided:

$$\begin{cases} \min_{\mathbf{w}} \langle \mathbf{w}, \mathbf{w} \rangle = \|\mathbf{w}\|^2 \\ \min_{\varepsilon_i} \frac{\gamma}{2} \sum_i \varepsilon_i^2 \\ (\mathbf{w} \cdot \mathbf{x}_i + b) = y_i(1 - \varepsilon_i) \end{cases} \tag{53}$$

where γ is the sensitivity to the total amount of errors ε_i. This optimization problem can be formulated via the Lagrangian multipliers $L(x) = f(x) - \lambda g(x)$, where the condition $f(x)$ is given by the inner product of $\mathbf{w}$ and the sum over the ε_i, while the constraint $g(x)$ by the last equation of the system [129]:

$$L(\mathbf{w}, b, \vec{\alpha}) = \frac{\|\mathbf{w}\|^2}{2} + \frac{\gamma}{2} \sum_i \varepsilon_i^2 - \sum_i \alpha_i [\langle \mathbf{w}, \mathbf{x}_i \rangle + b - y_i(1 - \varepsilon_i)] \tag{54}$$

The α_i coefficients play the role for the Lagrangian multipliers. To get the best parameters, we derive the Lagrangian $L(\mathbf{w}, b, \vec{\alpha})$ with respect to $\mathbf{w}$, b, ε_i and α_i:

$$\begin{cases} \vec{\nabla}_{\mathbf{w}} L = \mathbf{w} - \sum_i \alpha_i \mathbf{x}_i \overset{!}{=} 0 \Rightarrow \mathbf{w} = \sum_i \alpha_i \mathbf{x}_i \\ \frac{\partial L}{\partial b} = -\sum_i \alpha_i \overset{!}{=} 0 \Rightarrow \sum_i \alpha_i = 0 \\ \frac{\partial L}{\partial \varepsilon_j} = \gamma \varepsilon_j - \alpha_j y_j \overset{!}{=} 0 \Rightarrow \varepsilon_j = \frac{\alpha_j y_j}{\gamma} \\ \frac{\partial L}{\partial \alpha_j} \overset{!}{=} 0 \Rightarrow \mathbf{w} \cdot \mathbf{x}_j + b = y_j - \varepsilon_j y_j \end{cases} \tag{55}$$

Afterwards, substitute the first and the third equation in the fourth one, which yields

$$\sum_i \alpha_i \mathbf{x}_i \cdot \mathbf{x}_j + b = y_j - \frac{\alpha_j}{\gamma} \tag{56}$$

as $y_i^2 = 1$. The same expression can be rewritten in terms of the kernel K_{ij} and with all the parameters $\{b, \vec{\alpha}\}$ on the left member:

$$\sum_i \alpha_i K_{ij} + \frac{\alpha_j}{\gamma} + b = y_j \tag{57}$$

8.5.3. Quantum advantage due to HHL algorithm

Taking into account the constraint $\sum_i \alpha_i = 0$ from Eq. (55), the same expression in Eq. (57) can be reformulated into a matrix fashion as

$$\begin{pmatrix} 0 & \vec{1}^T \\ \vec{1} & K + \gamma^{-1} \hat{I} \end{pmatrix} \begin{pmatrix} b \\ \vec{\alpha} \end{pmatrix} = F \begin{pmatrix} b \\ \vec{\alpha} \end{pmatrix} = \begin{pmatrix} 0 \\ \vec{y} \end{pmatrix} \tag{58}$$

The first row holds because of the third equation in the system from Eq. (55). The dimension of the kernel matrix K and the identity $\mathbb{I}$ which multiplies γ^{-1} is M, i.e. the number of data from the training. Therefore, it is possible to obtain the parameters $\{b, \vec{\alpha}\}$ by just inverting the F matrix:

$$\begin{pmatrix} b \\ \vec{\alpha} \end{pmatrix} = F^{-1} \begin{pmatrix} 0 \\ \vec{y} \end{pmatrix} \tag{59}$$

The F matrix can be expressed as $J + K_\gamma$, where

$$J = \begin{pmatrix} 0 & \vec{1}^T \\ \vec{1} & \mathbb{O} \end{pmatrix}, \qquad K_\gamma = \begin{pmatrix} 0 & \vec{0}^T \\ \vec{0} & K + \gamma^{-1}\mathbb{I} \end{pmatrix} \tag{60}$$

Back to Eq. (58), it is possible to encode the training parameters $\{b, \vec{\alpha}\}$ as

$$|b\rangle|\vec{\alpha}\rangle = \frac{1}{\sqrt{C}} \left(b|0\rangle + \sum_{i=1}^{M} \alpha_k |k\rangle \right) \tag{61}$$

where the normalization constant is set to be $C = b^2 + \sum_{i=1}^{M} \alpha_k^2$. As in the classical case, there are many available definitions of kernels, the first one of which can be

$$K = \sum_{i,j} \langle \phi_i | \phi_j \rangle \, |i\rangle\langle j| \tag{62}$$

Just for instance, in order to reproduce a polynomial kernel, it is possible to embed a state vector $|x_j\rangle$ in a higher dimensional space [127], $|\phi_j\rangle = |x_j\rangle^{\otimes s}$, and therefore the inner product $\langle \phi_j | \phi_i \rangle = \langle x_j | x_i \rangle^s$. To invert the matrix in Eq. (58), it is possible to apply the HHL algorithm, achieving an exponential speed-up. An overall explanation for the HHL algorithm is provided in Appendix A. In the first place, the linear system in Eq. (58) can be embedded into a quantum transformation as $\hat{F}|b, \vec{\alpha}\rangle = |\vec{y}\rangle$, where the matrix F (and consequently the $\hat{F}$ operator) is defined as $(J + K + \gamma^{-1}\mathbb{I})/\,\text{Tr}\{J + K + \gamma^{-1}\mathbb{I}\}$. The F matrix has a $(M + 1) \times (M + 1)$ dimension, along with a norm $\|F\| = \max_j \|F\mathbf{x}_j\|/\|\mathbf{x}_j\| \leq 1$ ($\mathbf{x}_j$ being its eigenvectors) because of the trace normalization. Thanks to the Lie-Trotter formula, it is possible to decompose the exponentiation of $\hat{F}$ as

$$e^{i\hat{F}\Delta t} = e^{-i\hat{\mathbb{I}}\gamma^{-1}\Delta t/C} e^{-i\hat{J}\Delta t/C} e^{i\hat{K}\Delta t/C} + O(\Delta t^2) \tag{63}$$

where $C = \text{Tr}\{J + K + \gamma^{-1}\mathbb{I}\}$ is the trace normalization. In order to apply the HHL algorithm, the state $1|\vec{y}\rangle$ must be endowed with an ancillary qubit, in order to store the eigenvalues of $e^{i\hat{F}\Delta t}$, and decomposed into a basis for $\hat{F}$:

$$|\vec{y}\rangle \rightarrow \sum_{j=1}^{M+1} \langle u_j | \vec{y} \rangle \, |u_j\rangle|0\rangle \tag{64}$$

Thus, applying the HHL algorithm, the $|\vec{y}\rangle|0\rangle$ state transforms into

$$\sum_{j=1}^{M+1} \langle u_j | \vec{y} \rangle \, |u_j\rangle/\lambda_j \tag{65}$$

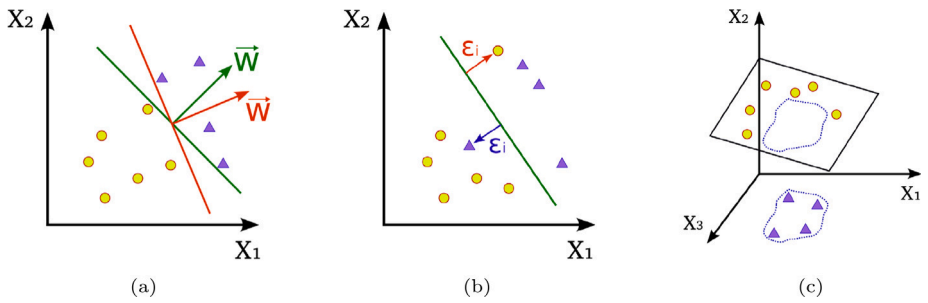

Fig. 6. The first two images represent a $2D$ region, split by two axes X_1 and X_2 (a) Tuning $\mathbf{w}$ changes the direction of the hyperplane, and therefore the distance between the data $\mathbf{x}_i$ and the hyperplane itself. The $\mathbf{w}$ vector is orthogonal to the hyperplane itself (b) Some data may follow into a disputed area, thereafter an error ε_i (soft variables) is introduced to correct these margins (c) Introducing feature maps, a linear classification is made feasible, where otherwise, in a lower dimension space, it would not be possible.

8.6. Natively quantum kernel methods

The kernel method consists of embedding a set of data into a higher-dimensional space (even infinite-dimensional) called feature space [74]. Given the space of data $\mathcal{X}$, the kernel k is defined as a map $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. Such a map can be considered as a metric in the $\mathcal{X}$ space of the data. In the previous section, the kernel function k has been alternatively defined through the feature maps $\boldsymbol{\Phi}$, where $k(x, x') = \langle \boldsymbol{\Phi}(x) | \boldsymbol{\Phi}(x') \rangle$, where $x, x' \in \mathcal{X}$. The feature map $\boldsymbol{\Phi}$ is a map between the space of data $\mathcal{X}$ and the feature space $\mathcal{H}$, $\boldsymbol{\Phi} : \mathcal{X} \to \mathcal{H}$. A visual representation for this encoding can be seen in Fig. 6(c). Such method turns to be useful in quantum machine learning, where classical data need to be encoded into a Hilbert space to perform some computation. It follows that the kernel function, encoding the data into the quantum circuit, induces a norm on the Hilbert space $\mathcal{H}$, and therefore a distance in the $\mathcal{X}$ space. Such feature accomplishes the purpose for a classification algorithm: given a target y, it is possible to compute the distance $k(x, y)$ thanks to the embedding in the feature space $\mathcal{H}$. By this approach, the data can be directly analyzed into a Hilbert space of features, where it is possible to deploy linear classifiers, relying on the inner products between quantum states. Increasing the size of the Hilbert space, such kernels turn to be classically intractable.

A way to encode the information into qubits (i.e. in the Hilbert space) could be formulated by introducing an operator $\hat{U}_{\boldsymbol{\Phi}}$, acting as $\hat{U}_{\boldsymbol{\Phi}}(x)|0\rangle = |x\rangle$ [34]. In such formulation, the $\hat{U}_{\boldsymbol{\Phi}}$ operator acts as a creator over the vacuum state $|0\rangle$. Thus the kernel can be estimated by confronting the $\hat{U}$ operators:

$$k(x, x') = \left| {}^{N\otimes}\langle 0 | \hat{U}_{\boldsymbol{\Phi}}(x)^{\dagger} \hat{U}_{\boldsymbol{\Phi}}(x) | 0 \rangle^{\otimes N} \right|^2 \tag{66}$$

By such formulation, the kernel entry can be evaluated on a quantum computer by measuring the $\hat{U}_{\boldsymbol{\Phi}}(x)^{\dagger}\hat{U}_{\boldsymbol{\Phi}}(x)|0\rangle^{\otimes N}$ state in the computational basis with repeated measurement shots and recording the probability of collapsing the output into the $|0\rangle^{\otimes N}$ state.

9. Quantum unsupervised learning

Recently, unsupervised learning gained wide success due to generative techniques, which allow to produce genuine new data mimicking the original dataset. Such techniques rely on sampling data from an unknown distribution, according to which the original ones are distributed. Quantum devices allow to generate samples from any distribution very efficiently, due to the intrinsic probabilistic nature of quantum mechanics. For instance, in [24] it has been proved that a Boltzmann Machine on an adiabatic quantum computer performs 64 times faster than its classical counterpart, involving tasks and data concerning the field of cybersecurity. In the next Section, we detail how classical generative-adversary techniques are designed and suited for quantum computers and anomaly detection purposes.

9.1. QGAN for anomaly detection

Differently from the approach based on SVM mentioned in the previous section, another approach has been proposed by Herr et al. [23] relying on hybrid quantum GANs for the anomaly detection task.

GAN (Generative Adversary Network) is an unsupervised algorithm of machine learning. This algorithm is based on two agents, the discriminative and generative one. Given a distribution of data $\mathbf{x} \in p_r(\mathbf{x})$ and a set of labels to pair $p_r(\mathbf{x})$ with, the former model tries to fit the best conditional probability $p(y|\mathbf{x})$, the latter how to generate the joint probability $p(y, \mathbf{x})$ for the data distribution. It is possible to introduce some latent variables, $\mathbf{z}$, to mimic the distribution for the $\mathbf{x}$ data. Given a distribution $p_g(\mathbf{z})$ for the latent variables $\mathbf{z}$ [151], the purpose for the generative model is to get the best parameters θ to build a function $G_\theta : \mathcal{Z} \to \mathcal{X}$. The data generated from G_θ will be distributed according to a probability distribution $\mathbb{P}_\theta$, while the true (unknown) distribution of data in the $\mathcal{X}$ space will be given by $\mathbb{P}_t$ (t standing for true). The purpose of the discriminative model, now that the hidden variables $\mathbf{z}$ are embedded in the $\mathcal{X}$ space, is to distinguish which variables are distributed according to $\mathbb{P}_t$ rather than $\mathbb{P}_\theta$: $D_\omega : \mathcal{X} \to [0; 1]$. The problem can be reformulated as a *minmax* one. In the WGAN (Wasserstein GAN) formulation [23], given a set of data $\mathbf{x}$ distributed accordingly to $\mathbb{P}_t$ (written as $\mathbf{x} \sim \mathbb{P}_t$), and a set of $\mathbf{z}$ with a $\mathbb{P}_z$ distribution, the purpose is to maximize the expectation function over D_ω, neutralizing at the same time the "fraudulent" action of G_θ:

$$\min_G \max_{D \in \mathcal{D}} \mathbb{E}_{\mathbf{x}\sim\mathbb{P}_t}[D(\mathbf{x})] - \mathbb{E}_{\mathbf{z}\sim\mathbb{P}_z}[D(G(\mathbf{z}))] \tag{67}$$

where $\mathcal{D}$ is the class of all the 1-Lipschitz functions. A α-Lipschitz function f is defined such that

$$\|f(x) - f(x_0)\|_{\mathbb{C}} \leq \alpha \|x - x_0\|_{\mathbb{D}} \tag{68}$$

where $0 < \alpha \leq 1$ (for $\alpha = 1$, f is defined 1-Lipschitz), $\mathbb{D}$ and $\mathbb{C}$ are respectively the domain and codomain of the function f and are metric spaces endowed with a distance and a norm (expressed by $\|.\|$ in Eq. (68)). Thereafter, it is possible to state the proposition by Gulrajani et al. [152]:

Proposition 1. *Let $\mathbb{P}_t$ and $\mathbb{P}_g$ two distributions in a $\mathcal{X}$ compact metric space. Then, there is a 1-Lipschitz function f^* which is the optimal solution of $\max_{\|f\|\leq L} \mathbb{E}_{y\sim\mathbb{P}_t}[f(y)] - \mathbb{E}_{x\sim\mathbb{P}_g}[f(x)]$.*

As a corollary to the proposition, it is stated that f^* has gradient norm 1 almost everywhere under $\mathbb{P}_t$ and $\mathbb{P}_g$. Without entering the mathematical details and proof for such proposition, which we recommend to Gulrajani's paper [152], it is possible to set the minmax problem in Eq. (67) with Lagrangian multipliers:

$$\mathcal{L}_c(\omega, \theta) = \mathbb{E}_{\mathbf{z}\sim\mathbb{P}_z}[D_\omega(G_\theta(\mathbf{z}))] - \mathbb{E}_{\mathbf{x}\sim\mathbb{P}_t}[D_\omega(\mathbf{x})] + \lambda \mathbb{E}_{\mathbb{P}_{\tilde{\mathbf{x}}}(\tilde{\mathbf{x}})}[(\|\nabla_{\tilde{\mathbf{x}}} D_\omega(\tilde{\mathbf{x}})\| - 1)^2] \tag{69}$$

where $\tilde{\mathbf{x}} = \mathbf{x} + \epsilon(G(\mathbf{z}) - \mathbf{x})$, ϵ being uniformly distributed in the range $(0, 1)$, i.e. $\epsilon \sim U(0, 1)$, while $\mathbf{x} \sim \mathbb{P}_t$ and $\mathbf{z} \sim \mathbb{P}_g$ still. The multiplier term is called gradient penalty term to the critic loss function $\mathcal{L}_c(\omega, \theta)$.

9.1.1. Training the NNs as competitors

The training for a WGAN consists of two steps: in the first one, given a set of generated $G_\theta(\mathbf{z})$ and true $\mathbf{x}$ inputs, the purpose is to find the global minimum for the Lagrangian loss function defined in Eq. (69), w.r.t. ω hyperparameters for the D_ω model via a stochastic gradient descent technique.

In the second step, the purpose is instead to enhance the pursuit of the generator. The way to achieve such goal is to maximize the first Lagrangian term, w.r.t. both the θ and ω hyperparameters:

$$\mathcal{L}_g(\theta) = -\mathbb{E}_{\mathbf{z}\sim\mathbb{P}_z}[D_\omega(G_\theta(\mathbf{z}))] \tag{70}$$

In its classical fashion, the generative model G_θ is built up by a series of L layers. In the first place, the latent variables $\mathbf{z} \sim U(0, 1)$ are reshaped through a series of maps $g_i : \mathbb{R}^N \to \mathbb{R}^N$, so that the overall NN model results in

$$g(\theta, \mathbf{z}) = (g_{\theta_L} \circ g_{\theta_{L-1}} \dots \circ g_{\theta_1})(\mathbf{z}) \tag{71}$$

where $\theta = (\theta_L, \theta_{L-1}, \dots, \theta_1)$. A set of activation functions can be applied for each g_{θ_i} layer: in [23] leaky ReLU were deployed. Secondly, the final form of the generator will be given by the $W : \mathbb{R}^N \to \mathbb{R}^M$ function:

$$G_\theta(\mathbf{z}, \phi) = W[g(\theta, \mathbf{z}), \phi] \tag{72}$$

W in Ref. [23] is set to be a sigmoid function on $w_{ij}z_j + b_i$, with ϕ collecting the corresponding hyperparameters of the weight matrix w and the bias vector $\mathbf{b}$. To update the hyperparameters, it is possible to adopt any gradient descent method. The updating proceeds by the usual chain rule in the derivation process:

$$\frac{\partial \mathcal{L}_g}{\partial \theta_m} = -\mathbb{E}_{\mathbb{P}(\mathbf{z})}\left[\frac{\partial D}{\partial G_\theta}\frac{\partial G_\theta}{\partial g_m}\frac{\partial g_m}{\partial \theta_m}\right] \tag{73}$$

The discriminative model $D_\omega : \mathbb{R}^M \to \mathbb{R}$ is a NN endowed with several hidden layers, mapping the data in a $\mathbb{R}^M$ space to the label space in $\mathbb{R}$.

9.1.2. Quantum variational circuit for generative models

Quantum circuits support the generative procedure of the model. In fact, quantum computers are expected to sample efficiently from distributions which are hard in a classical way [76,87,153–155]. On the contrary, the critic model needs lots of classical data, which requires too much time to be loaded and makes such transposition unfeasible in the NISQ era [156].

When implementing the generative model on a quantum device, the $\mathbf{z}$ latent variables are given into a uniform distribution $\mathbf{z} \sim U(-\pi, \pi)$, whereas the encoded state $|\mathbf{z}\rangle$ is given by the preparation layer $\hat{S}(\mathbf{z})|0\rangle^{\otimes N}$. The operator $\hat{S}$, which implements such preparation layer, is composed as $\hat{S}(\mathbf{z}) = \bigotimes_{i=1}^N R_i^X(z_i)$, R_i^X being the X-rotation over the z_i angle. After the state has been encoded, it follows a layer of rotations $\hat{U}_\nu(\vec{\theta})$ in all the $\{\hat{X}, \hat{Y}, \hat{Z}\}$ basis, alternated to CNOT gates. Therefore, two hyperparameters are given: ν stores the basis on which to perform rotations, $\vec{\theta}$ the angles. While ν encodes the architecture of the circuit, the $\vec{\theta}$ angles are the variational parameters to optimize on. The multilayer generative function g in Eq. (70) is transposed in an expectation value over $\hat{\mathbf{Z}} = \bigotimes_{i=1}^N \hat{Z}_i$:

$$g(\theta, \mathbf{z}) = \langle \mathbf{z}|\hat{U}_\nu(\vec{\theta})\hat{\mathbf{Z}}\hat{U}_\nu(\vec{\theta})|\mathbf{z}\rangle \tag{74}$$

where instead of composing $\dots \circ g_{i+1} \circ g_i \circ \dots$ layers of activation functions, there is a sequence of $\hat{U}_\nu(\vec{\theta})$ operators. At the end, a classical activation function W is applied to compose the last layer for the generative model. Beside this difference, the gradient descent method is applied in the same manner as from Eq. (73), but instead the derivative of g_m is given by

$$\frac{\partial g}{\partial \theta_m} = \frac{\partial \langle \hat{\mathbf{Z}} \rangle_{\mathbf{z},\vec{\theta},\nu}}{\partial \theta_m} \tag{75}$$

9.2. Other approaches to quantum anomaly detection by unsupervised learning methods

Generally speaking, quantum anomaly detection has been intensively explored in the past few years [29]. Anomaly detection for cybersecurity can take advantage of its development in other fields. For instance, a field of application of anomaly detection is particle physics. There, a number of algorithms have been proposed. For instance, Alve et al. [157] have applied QAD to an analysis characterized by a low statistics dataset. They have explored anomaly detection task in the four-lepton final state at the Large Hadron Collider that is limited by a small dataset, by a semi-supervised mode, without finding any evidence of speed-up. On the contrary, other examples sharing the unsupervised approach provided quantum speed-up, as follows.

9.2.1. Quantum auto-encoders

Quantum auto-encoders have been assessed for unsupervised machine learning models based on artificial neural networks. The aim consists of learning background distributions by quantum auto-encoders based on variational quantum circuits, as problem of anomaly detection at the LHC collider. For representative signals, it turns out that a simple quantum auto-encoder outperforms classical auto-encoders [31]. There, a quantum auto-encoder has been developed, consisting of a circuit divided into three blocks, namely the state preparation that encodes classical inputs into quantum states, the unitary evolution circuit that evolves the input states, and the measurement and postprocessing part that measures the evolved state and processing the obtained observables.

9.2.2. Amplitude estimation-based method

An anomaly detection algorithm based on density estimation (ADDE) has been proposed by Liang et al. [158] to potentially express exponential speed-up, but it was later found not executing. Then, another group demonstrated such an exponential speed-up based on a modified version [159]. Such a new quantum ADDE algorithm is based on amplitude estimation. It is shown that such algorithm can achieve exponential speed-up on the number M of training data points compared with the classical counterpart.

9.2.3. Natively quantum kernels and hardware benchmarking

In 2022, anomaly detection for credit card fraud detection have been demonstrated by quantum kernels on 20 qubits by authors including HSBC Bank affiliation [160]. The benchmarks consist of kernel-based approaches, in particular unsupervised modeling on one-class support vector machines (OC-SVM). Quantum kernels are applied to different type of anomaly detection, leading to observe that quantum fraud detection challenges the equivalent classical protocols at increasing number of features, which are equal to the number of qubits for data embedding. The better precision has been achieved by combining quantum kernels with re-uploading, with the advantage increasing with the size of the system. The Authors claim that with 20 qubits the quantum–classical separation of average precision is equal to 15%. The Authors estimate the computational cost to estimate the Gram matrix representing the kernel is $O(N_s^2)$ where N_s is the number of samples, while the continuous retraining to update on-the-fly the kernel is $O(N_s)$. Instead, the time needed for inference (to assign a label fraud or not) for detecting N_d new-coming samples is of $O(N_d N_s)$ kernel evaluations. The report is of particular interest as an evaluation is made for what concerns such inference time for three different hardware platforms: (1) superconducting circuits, (2) trapped ions and (3) optical systems. One can evaluate the training time for a dataset of 500 elements. In su-

perconducting qubits, operation happen at MHz speed. A reproducible kernel measurements may require at least $N_{shots} = 10^{3-6}$ measurement shots. With 10^5 kernel evaluations, the training time is 100 s - 28 h training time at optimistic MHz rate same cost of 28 h. The inference time is significantly smaller down to 0.5 s in the case of reduced dataset. Instead, for large datasets (100000 samples), it may raise to 16 weeks, which could only be reduced with partial inference of Gram matrix. In trapped ions, for 10 kHz per shots, with 10^5 kernel evaluations the training time and $N_{shots} = 10^{3-6}$ is 3 h - 17 weeks. With photons on deterministic gates, which is currently still an open field of research, the expected time ranges between 10 ms and 10 s.

10. Quantum approximate optimization algorithm

Data clustering is the process of identifying natural groupings or clusters within multidimensional data based on some similarity measure. Clustering is a fundamental process in many different disciplines [161], for instance, it can be employed to divide the data set into a specified number of clusters, trying to minimize certain criteria (e.g. a square error function) falling into the class of optimization problems. Moreover, clustering algorithms are employed to perform network traffic identification [162] and for graph-based network security [163]. In literature, in fact, a common approach consists of representing the servers as nodes of a graph, and the flow of data between them as the edges of the graph itself [164]. By monitoring the topology of the graph, relying on a technique called graph similarity, any anomaly can be straightforwardly detected. In 2022, Li et al. proposed an algorithm of clustering in order to monitor the traffic flow on the web [162]. Nevertheless, despite the effectiveness of such approach, the graph encoding for anomaly detections turns the problem to an NP-hard one by scaling with the number of nodes – see [163,164]. Instead, quantum computation is able to tackle graph-based problems in a polynomial time. In the next Section, we provide a paramount example about how an NP-hard graph problem could be leveraged by a quantum machine learning approach.

10.1. The MaxCut problem

The MaxCut algorithm is a NP-hard combinatorial problem [165,166] which can be set as follows. Given a $G = (V, E)$ graph, $V = \{1, \dots, n\}$ being the vertices of the graph and E their connections, the weights of the $(i, j) \in E$ connections are given by the weight matrix w_{ij}. The purpose in the MaxCut problem is to find the best subset $S \subset V$ of vertices and its complement $\bar{S}$ to maximize the sum over the weights connecting the two subsets [165]: $w(S, \bar{S}) = \frac{1}{2}\sum_{i\in S, j\in\bar{S}} w_{ij}$. The MaxCut problem can be formulated as the following integer quadratic program [165,166] and therefore mapped into a Hamiltonian formulation, inspired by the Ising model:

$$\max \sum_{1\le i<j\le n} \frac{w_{ij}}{2}(1 - z_i z_j) \to \hat{H}_C = \sum_{ij} \frac{w_{ij}}{2}(\hat{I} - \hat{Z}_i\hat{Z}_j) \tag{76}$$

The constraint $z_i = \pm 1$ holds, allowing to replace such classical variables with the third Pauli matrix from $\mathfrak{su}(2)$ algebra, i.e. the $\hat{Z}$ operator. Nonetheless, it holds that $z_i = \langle \hat{Z}_i \rangle$. The cut is defined by the condition $S = \{i \in V | z_i = +1\}$, and it can be set by maximizing the $E_C = \langle \hat{H}_C \rangle$ observable [167]. When two vertices belong to the same subset S or $\bar{S}$, it follows that $z_i = z_j$ and the contribution to E_C is null. Thus, the set of S and $\bar{S}$ can be thought as a partition of the system in up and down spins.

The MaxCut problem can be employed in the field of data mining and machine learning [168], with special regards to unsupervised learning [169]: it is possible to recreate unsupervised learning clustering of data by mapping the problem to a graph optimization problem and finding the minimum energy for a MaxCut problem formulation.

10.2. QAOA formulation

The quantum approximate optimization algorithm (QAOA) has been many times applied to tackle the MaxCut problem [94,169–171]. Such algorithm consists of preparing a register of n-qubits in the eigenstate of a Hamiltonian $\hat{H}_B$:

$$|\Psi(t=0)\rangle = |+\rangle^{\otimes N}, \quad \hat{H}_B = \bigotimes_{i=1}^{N} \hat{X}_i \Rightarrow \hat{H}_B|\Psi(0)\rangle = |\Psi(0)\rangle \tag{77}$$

More specifically, the state $|+\rangle^{\otimes N}$ is the maximum for the Hamiltonian $\hat{H}_B$. The purpose is now making $|\Psi(t=0)\rangle$ evolve to the maximum eigenstate for the $-\hat{H}_C$ Hamiltonian from Eq. (76) via the adiabatic theorem, i.e. to the minimum eigenstate for $\hat{H}_C$. The next step is thus to encode a time-dependent Hamiltonian $\hat{H}(t)$ to make the state $|\Psi\rangle$ evolve [4]:

$$\hat{H}(t) = \left[1 - \frac{t}{T}\right]\hat{H}_B - \frac{t}{T}\hat{H}_C, \qquad 0 \le t \le T \tag{78}$$

The annealing schedules are set by the $1-t/T$ and t/T terms, the $\hat{H}_P$ Hamiltonian is shaped on the form of the $\hat{H}_C$ Hamiltonian in Eq. (76), while the transverse field Hamiltonians ($\hat{H}_T$ and $\hat{H}_B$) still keep the same form. Via the adiabatic theorem, it is possible to make $|\Psi(t)\rangle$ evolve from the higher energy state of $\hat{H}_B$ to the higher energy state of $-\hat{H}_C$ (and therefore to the ground state of $\hat{H}_C$). As the Hamiltonian in Eq. (78) depends on time, the corresponding time evolution is given by

$$|\Psi(T)\rangle = \exp\left(-\frac{i}{\hbar}\int_0^T \mathrm{d}t\, \hat{H}(t)\right)|\Psi(0)\rangle = \exp\left(-\frac{iT}{\hbar}\int_0^1 \mathrm{d}s\, \hat{H}(s)\right)|\Psi(0)\rangle \tag{79}$$

where $s = t/T$. It is possible to approximate the above evolution by splitting the continuous trajectory along t (or s) in a patchwork of p small, discrete steps of duration $\epsilon = 1/p$ [4,172]. By applying the Trotter formula [173], the operator in the above equation can be approximated as

$$\exp\left(-\frac{iT}{\hbar}\int_0^1 \mathrm{d}s\, \hat{H}(s)\right) \approx \prod_{k=0}^{p-1} \exp\left(-\frac{iT}{p\hbar}\hat{H}_k\right) \tag{80}$$

where $\hat{H}_k = \hat{H}(k/p)$. In the limit for $p \to \infty$, it is possible to involve again the Lie-Trotter formula [94,174] to split each $\hat{H}_k$ Hamiltonian into the $\hat{H}_C$ and $\hat{H}_B$ terms:

$$\prod_{k=1}^{p} \exp\left(-\frac{iT}{p\hbar}\left[1 - \frac{k}{p}\right]\hat{H}_C\right)\exp\left(-\frac{iT}{p\hbar}\frac{k}{p}\hat{H}_B\right)|\Psi(0)\rangle \tag{81}$$

The bigger is p, the better both the approximations in Eqs. (80) and (81) work. It is possible to treat the terms in the round brackets as a set of angles γ_j, β_j, to map the overall evolution into

$$|\Psi(\boldsymbol{\gamma}, \boldsymbol{\beta})\rangle = e^{-i\beta_p\hat{H}_B}e^{i\gamma_p\hat{H}_C} \dots e^{-i\beta_1\hat{H}_B}e^{i\gamma_1\hat{H}_C}|+\rangle^{\otimes n} \tag{82}$$

In such formulation, the time evolution via a parameter t has been substituted by $2p$ unitary transformations parameterized by a set of $(\boldsymbol{\gamma}, \boldsymbol{\beta})$ angles. When implementing such operator on a circuit, the number of repetitions over the $e^{-i\beta_j\hat{H}_B}e^{i\gamma_j\hat{H}_C}$ layers stands for the depth p of the circuit itself. The state $|x\rangle$ is therefore evolved naturally to the solution under the action of the following unitary operator:

$$|\Psi(\boldsymbol{\gamma}, \boldsymbol{\beta})\rangle := \left(\prod_{i=1}^{p} \hat{U}_B(\beta_i)\hat{U}_C(\gamma_i)\right)|+\rangle^{\otimes n} \tag{83}$$

Recall the cost function E_C in terms of the number p of layers which are inserted in the evolution from Eq. (83), e.g. E_p: $E_p(\boldsymbol{\gamma}, \boldsymbol{\beta}) = \langle\Psi(\boldsymbol{\gamma}, \boldsymbol{\beta})|\hat{H}_C|\Psi(\boldsymbol{\gamma}, \boldsymbol{\beta})\rangle$. More layers are inserted (i.e. the higher is p), the more the solution is supposed to be exact. Afterwards, it is possible to define the maximum value over the expectation of E_p: $M_p = \max_{\boldsymbol{\beta},\boldsymbol{\gamma}} E_p$. Therefore, by the adiabatic theorem, it is possible to state that $M_{p+1} \ge$

M_p. Eventually, it is possible to map the adiabatic process into an optimization for the $2p$ parameters $\boldsymbol{\gamma}$, $\boldsymbol{\beta}$, which can be achieved by a hybrid algorithm combining gradient descent methods on CPUs/GPUs with backpropagation on the quantum circuits. A useful metric, to assess how far the state is from the solution (i.e. the ground state of $\hat{H}_C$), is given by the approximation ratio parameter r, formulated as [34,167,175–182] $r = E/E_{max}$. Here $E = \langle\Psi|\hat{H}_C|\Psi\rangle$, with $|\Psi\rangle$ being the actual state of the system and E_{max} the maximum eigenvalue of the Hamiltonian operator, whose corresponding eigenstate is the goal of the problem under consideration. When r tunes to 1, the exact solution is provided. The approximation ratio, by such definition, returns the cost function (in our case, the Hamiltonian spectrum) normalized in the $[0; 1]$ codomain.

Said this, some critical considerations have to be done. Once the QAOA was proposed for finding approximate solutions to combinatorial optimization problems [94], it was subsequently shown that QAOA solves the combinatorial problem Max E3LIN2 with better approximation ratio with respect to any polynomial-time classical algorithm known, at the time, but soon a better classical algorithm with better approximation ratio was found [183]. The assignment of a quantum algorithm to a class of speed-up may suggest the priority around the aspects which can be investigated.

11. Quantum reinforcement learning

Reinforcement learning (RL) has been poorly explored in the field of quantum information, and just in the last years some interest has been raising towards this branch of Machine Learning [184–186]. For instance, Chen et al. [187] proposed a variational quantum reinforcement learning algorithm via evolutionary optimization with no evidence of quantum speed-up. Another variational implementation is due to Acuto et al. [188]. Dalla Pozza et al. [189] developed a quantum RL framework to solve a quantum maze with speed-up, and Cherrat et al. [190] show a quadratic speed-up under certain conditions for their quantum RL based on policy iteration. The field looks currently less developed with respect to quantum supervised and unsupervised paradigms and more development should be expected before prospecting an evident impact on security related tasks. Nevertheless, for sake of completeness, this Section outlines some key aspects of quantum RL, which may inspire future research around its intersection with cybersecurity.

According to one of the first proposals, when transposing classical algorithms of reinforcement learning in the quantum domain, the actions and the states of the system can be described as elements spanning two different Hilbert spaces $\mathcal{A} = \{|a_i\rangle\}$ and $\mathcal{S} = \{|s_i\rangle\}$, or even $\mathcal{E}$ (where $\mathcal{E}$ stands for environment) [191,192]. Apart from the qubits belonging to these two $\mathcal{A}$ and $\mathcal{S}$ systems, an auxiliary system $\mathcal{R}$, called register, can be added [26,184,193]. In such case, when initializing the overall system, the state will be presented as

$$|\Psi\rangle = |\mathcal{A}\rangle|\mathcal{E}\rangle|\mathcal{R}\rangle \tag{84}$$

As from the classical RL algorithms, three functions are required: a policy function, a reward function (RF) and a value function (VF) [26,194]. The reward function is the criterion to evaluate the goodness of an action taken by the agent, with respect to the fixed task. The value function evaluates the general convergence of the algorithm to the goal it has to be achieved. The policy function defines which action to take with respect to the fixed purpose. However, due to the nature of quantum mechanics, even extracting information from the environment to the space of actions needs a decision problem, which task is relied to the policy function. The process of extracting information from the environment to the actions can be though as an interaction with the two systems.

The simplest case deals with one qubit for the environment $\mathcal{E}$ and one qubit for the action space $\mathcal{A}$. Depending on the RL protocol to implement, the register space can be endowed with one or two qubits. Generally, such states are initialized to $|0\rangle$, i.e. $|\Psi\rangle = |0\rangle_{\mathcal{A}}|0\rangle_{\mathcal{E}}|0\rangle_{\mathcal{R}}|0\rangle_{\mathcal{R}}$.

In the first step, the data need to be uploaded into the $\mathcal{E}$ space:

$$|\Psi\rangle = |0\rangle_{\mathcal{A}}\left[\cos(\theta/2)|0\rangle_{\mathcal{E}} + e^{i\phi}\sin(\theta/2)|1\rangle_{\mathcal{E}}\right]|0\rangle_{\mathcal{R}}|0\rangle_{\mathcal{R}} \tag{85}$$

In the second place, apply a set of CNOT gates with $|a\rangle_{\mathcal{E}}$ as control and the $|0\rangle_{\mathcal{R}}$ as targets:

$$|\Psi\rangle = |0\rangle_{\mathcal{A}}\left[\cos(\theta/2)|0\rangle_{\mathcal{E}}|0\rangle_{\mathcal{R}}|0\rangle_{\mathcal{R}} + e^{i\phi}\sin(\theta/2)|1\rangle_{\mathcal{E}}|1\rangle_{\mathcal{R}}|1\rangle_{\mathcal{R}}\right] \tag{86}$$

11.1. Quantum adaptation algorithm

The quantum adaptation algorithm, proposed by F. Albarrán-Arriagada et al. [26] and applied by Shang Yu et al. (including Albarrán-Arriagada himself) in a semiquantum way [193] has been tested to rebuild a quantum state, in order to describe a quantum system. It consists of the following steps: start from a system where all of the $\mathcal{A}$, $\mathcal{E}$ and $\mathcal{R}$ subsystems take into account a single qubit. In the first place, encode the overall system in a similar fashion as in Eq. (85):

$$|\Psi\rangle = |0\rangle_{\mathcal{A}}\left[\cos(\theta/2)|0\rangle_{\mathcal{E}} + e^{i\phi}\sin(\theta/2)|1\rangle_{\mathcal{E}}\right]|0\rangle_{\mathcal{R}} \tag{87}$$

Therefore, apply a $\hat{CX}_{\mathcal{ER}}$ operator (CNOT on $\mathcal{E}$ as control, $\mathcal{R}$ as target):

$$|\Psi\rangle = |0\rangle_{\mathcal{A}}\left[\cos(\theta/2)|0\rangle_{\mathcal{E}}|0\rangle_{\mathcal{R}} + e^{i\phi}\sin(\theta/2)|1\rangle_{\mathcal{E}}|1\rangle_{\mathcal{R}}\right] \tag{88}$$

Afterwards, perform a measurement over $\mathcal{R}$ in the computational basis $\{|0\rangle, |1\rangle\}$, so that the $\mathcal{E}$ state is going to collapse in $|0\rangle_{\mathcal{E}}$ or $|1\rangle_{\mathcal{E}}$ with probabilities $\cos^2(\theta/2)$ or $\sin^2(\theta/2)$, respectively. If the superposition collapses to $|0\rangle$, the environment $\mathcal{E}$ and the action $\mathcal{A}$ share the same state, otherwise the latter needs to be updated. To update $|0\rangle_{\mathcal{A}}$, introduce the following operator:

$$\hat{U}^{(k)}_{\mathcal{A}}(\alpha^{(k)}, \beta^{(k)}) = e^{-i\hat{Z}_{\mathcal{A}}\alpha^{(k)}}e^{-i\hat{X}_{\mathcal{A}}\beta^{(k)}} \tag{89}$$

Here $\hat{Z}$ and $\hat{X}$ stand for the elements from Pauli algebra $\mathfrak{su}(2)$, and the suffix $\mathcal{A}$ shows that they are acting over the action space. The index k stands for the iteration over the process. The angles of rotation are defined in the following range: $\alpha^{(k)}, \beta^{(k)} \in [-\Delta^{(k)}/2; \Delta^{(k)}/2]$, where $\Delta^{(k)}$ is the parameter to update per iteration k. The operator $\hat{U}_{\mathcal{A}}$ acts on the $|\mathcal{A}\rangle$ state depending on the outcome from the measurement:

$$\hat{\mathcal{U}}^{(k)}_{\mathcal{A}} = [m_k\hat{U}^{(k)}_{\mathcal{A}}(\alpha^{(k)}, \beta^{(k)}) + (1 - m_k)\hat{I}_{\mathcal{A}}] \tag{90}$$

where m_k is the outcome from the kth measurement, 1 if $|1\rangle$, otherwise 0. To update the Δ parameter, the following rule has been proposed:

$$\Delta^{(k+1)} = [(1 - m_k)\mathcal{R} + m_k\mathcal{P}]\Delta^{(k)} \tag{91}$$

$\mathcal{R}$ and $\mathcal{P}$ are called the reward and punishment ratios, respectively $\mathcal{R} = \epsilon < 1$ and $\mathcal{P} = 1/\epsilon > 1$, so that every time the outcome is $|0\rangle$ the value of Δ is reduced, when $|1\rangle$ it is increased. ϵ is a hyperparameter to tune for every set of simulations, the better the ϵ parameter, the higher the fidelity between the simulated qubit and the initial state (see Table 4).

At the kth iteration, the system will be set in the state $|\Psi\rangle = \mathbb{U}^{(k)}_{\mathcal{A}}|0\rangle_{\mathcal{A}}|\psi\rangle_{\mathcal{E}}|0\rangle_{\mathcal{R}}$, where the $\mathbb{U}^{(k)}_{\mathcal{A}}$ operator accounts into memory all the previous actions over $\mathcal{A}$:

$$\mathbb{U}^{(k)}_{\mathcal{A}} = \mathcal{U}^{(k)}_{\mathcal{A}}(\alpha^{(k)}, \beta^{(k)})\mathbb{U}^{(k-1)}_{\mathcal{A}} \tag{92}$$

12. Concluding remarks

Anomaly detection performed on quantum computers by quantum machine learning algorithms is at its infancy, but reveals high potential. At the same time, one may expect a transition for what concerns the kind of data and the applications to be managed. Indeed, quantum machine learning suffers of the bottleneck of the data loading issue. Given that no qRAM does still exist, the $1-to-2^n$ parallelized encoding of data in qubits is currently not viable because of its exponential data loading cost. Therefore, three options can be considered: (i) a robust

Table 3
Taxonomy for Quantum feed-forward Neural Networks for all the reviewed algorithms. AF acronym stands for activation functions. All the fields in the table refer to Fig. 1.

Algorithm	Units	Learning	Architecture	NNs	Computation
QFFNN	Qubits	Supervised	Circuital	AF	Fully quantum
QRBM	Qubits	Supervised	Adiabatic	Variational	Fully quantum
CVNN	Qumodes	Supervised	Circuital	AF	Fully quantum
DMKDC	Qudits	Supervised	Circuital	Variational	Classical emulation
QSVM	Qubits	Supervised	Circuital	Variational	Hybrid
QGAN	Qubits	Unsupervised	Circuital	Variational	Hybrid
QAOA	Qubits	Unsupervised	Circuital	Variational	Hybrid
QRL	Qubits	Reinforcement	Circuital	Variational	Hybrid

but qubit-expensive $1-to-1$ encoding of classical data to be loaded by the register used as input of the quantum algorithm, or, alternatively – in some special cases – (ii) to generate the data by a pre-trained quantum circuit returning an approximate probability distribution (derived from another probability distribution easier to generate) which can introduce entangled states as input, or (iii) to feed the quantum algorithms by quantum data – another option which potentially inputs an entangled state. While it is still under investigation to which extent the quantum machine learning can be more precise and faster than classical machine learning methods on classical data, it is likely that the major advantage appears when quantum data are considered. Indeed, a quantum circuit naturally manages quantum states, while instead this is not straightforward in machine learning. In several cases, there is no knowledge whether strong quantum advantage does hold or not. Algorithms with *common* quantum advantage should be better explored by looking at demonstrating some stronger quantum speed-up degree while empirically evaluating the trend of its performances when scaling for instance the number of qubits. One should be aware that the empirical search of asymptotic behavior may change the estimate of the trend as soon as larger number of qubits are achieved. Cybersecurity inherits the algorithms from quantum machine learning, but carries the specificities of dealing with large datasets. Here, the mutually exclusive choice between kernel-based and variational learning shows the trade-off: kernel-based guarantees optimal kernel can always be found, but it scales with $O(N^2)$, while for variational learning it is possible in time $O(N)$. Any decision concerning application of machine learning to real anomaly detection datasets should begin with a real problem based on the dataset, on which a direct benchmarking comparison with classical methods could be evaluated.

CRediT authorship contribution statement

Sebastiano Corli: Writing – review & editing, Writing – original draft, Validation, Project administration, Investigation, Conceptualization. **Lorenzo Moro:** Writing – original draft, Investigation. **Daniele Dragoni:** Writing – original draft. **Massimiliano Dispenza:** Writing – original draft. **Enrico Prati:** Writing – review & editing, Validation, Supervision, Project administration, Methodology, Investigation, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix A. The HHL algorithm

The Harrow, Hassidim and Lloyd (HHL) algorithm aims to solve a linear system $A\mathbf{x} = \mathbf{b}$ using a quantum computer [17]. Such algorithm was proposed in [25]. The classical method known as the best scales roughly $O(Ns\sqrt{(}k)\log(1/\epsilon))$ operations, versus a $O(\log(N)k^2s^2/\epsilon)$ steps on a quantum computer, yielding an exponential speed-up in terms of number of operations N [25]. The other parameters, in the time-scaling complexity, are the sparsity s of the matrix, i.e. the most number of non-null entries from the rows of the A matrix [77], the condition number k, i.e. the ratio between the largest and smallest eigenvalues of A [25] and eventually the error ϵ. Thus, the HHL algorithm, to be better performing than the best classical algorithms, requires some caveats, as the A matrix to be sparse and to read out an expectation value over $\mathbf{x}$, such as $\langle x|\hat{M}|x\rangle$ ($\hat{M}$ being an observable), rather than outputting the exact $|x\rangle$ state. Nevertheless, such routines are quite common in quantum computation, and may pave the road to future applications in quantum machine learning.

In fact, the matrix inversion is a common routine for many computational processes. The HHL algorithm is a frequent subroutine for many machine learning methods. As in Fig. A.1, the HHL algorithm consists of three main blocks:

1. encoding the $\mathbf{b}$ vector into a quantum state $|b\rangle$ (or assume $|b\rangle$ to be already prepared);
2. perform a quantum phase estimation (QPE), apply a conditioned rotation on an auxiliary qubit by the achieved result and transform back the state by an inverse QPE;
3. measure the ancilla qubit.

Until the qRAM or other techniques of encoding will be leveraged, the first step could turn to be the main overhead [17], in terms of number N of operations, as the classical information, encoded in 2^n bits, needs to be compressed into n qubits. Once such step has been accounted, the QPE algorithm takes as input a state $|\psi\rangle_m \in \mathbb{C}^{2^m}$, an ancilla $|0\rangle_n \in \mathbb{C}^{2^n}$ and a unitary operation $\hat{U} \in \mathbb{C}^{2^m\times 2^m}$ to perform on $|\psi\rangle_m$. The $|\psi\rangle_m$ vector must be eigenvector for $\hat{U}$, under which hypothesis the QPE works in the following manner:

$$Q\hat{P}E\left[\hat{U}|0\rangle_n|\psi\rangle_m\right] = |\theta\rangle_n|\psi\rangle_m \tag{A.1}$$

with $\hat{U}$ acting over $|\psi\rangle_m$. From $|\theta\rangle_n$, it is possible to get the binary encoding of $2^n\theta/(2\pi)$. Therefore, to get θ it is mandatory to divide the result by 2^{n-1} and multiply by π. Just for instance, suppose to apply a $\hat{T}$ gate on the $|1\rangle$ qubit:

$$\hat{T}|1\rangle = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{4}} \end{bmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} \tag{A.2}$$

For $n = 3$, the QPE algorithm, applied on $\hat{T}|1\rangle$, returns 001, which is the binary encoding for 1. Thus, in order to get the correct phase, multiply by π and divide by 2^2, obtaining $\theta = \pi/4$.

To apply the QPE for the HHL algorithm, the unitary operator $\hat{U}$ can be decomposed as the complex exponentiation of a Hermitian generator $\hat{A}$:

$$\hat{U} = e^{i\hat{A}t} = \sum_{j=0}^{2^m-1} e^{i\lambda_j t}|u_j\rangle\langle u_j| \tag{A.3}$$

where λ_j are the eigenvalues for $\hat{A}$ and $|u_j\rangle$ its eigenvectors. Secondly, as any $\mathbb{C}^{2^m\times 2^m}$ Hermitian operator can generate a basis in $\mathbb{C}^{2^m}$ by its eigenstates, the $|\psi\rangle_m$ vector can be decomposed into its $|u_j\rangle$ generators:

$$|\psi\rangle_m = \sum_{j=0}^{2^m-1} b_j|u_j\rangle \tag{A.4}$$

b_j being the coefficients for $|\psi\rangle_m$ in the $\{|u_j\rangle\}_{j=0}^{2^m-1}$ basis. Afterwards, it is possible to apply the QPE transformation:

$$Q\hat{P}E\left[|0\rangle_n e^{i\hat{A}t}|\psi\rangle_m\right] = Q\hat{P}E\left[|0\rangle_n \sum_{j=0}^{2^m-1} e^{i\lambda_j t} b_j|u_j\rangle\right] = \sum_{j=0}^{2^m-1} b_j|\lambda_j\rangle|u_j\rangle \tag{A.5}$$

Up to this point, the $\mathbf{b}$ vector has been encoded into a $|\psi\rangle_m = \sum_{j=0}^{2^m-1} b_j|u_j\rangle$ state, on which a QPE routine has been acted. The next step is to introduce another ancilla qubit in the $|0\rangle$ state on which to perform a conditioned rotation, using the $|\lambda_j\rangle$ as control qubits:

Table 4
List of generated or available datasets exploited by the authors of the reviewed papers.

Algorithm	Dataset	Task
QFFNN [21,123,126]	Binary images (from 2 × 2 to 4 × 4)	Image classification
QRBM [24]	KDD CUP 99 [195], IDS2018 [196]	Web monitoring
CVNN [10]	Credit card transaction dataset [197]	Fraud detection
DMKDC [22]	Moonsandcircles dataset, approximation of probability density functions	Image classification
QGAN [23]	Kaggle credit card fraud detection	Fraud detection
QRL [26,194]	Gridworld (20 × 20 size)	Strategy planning

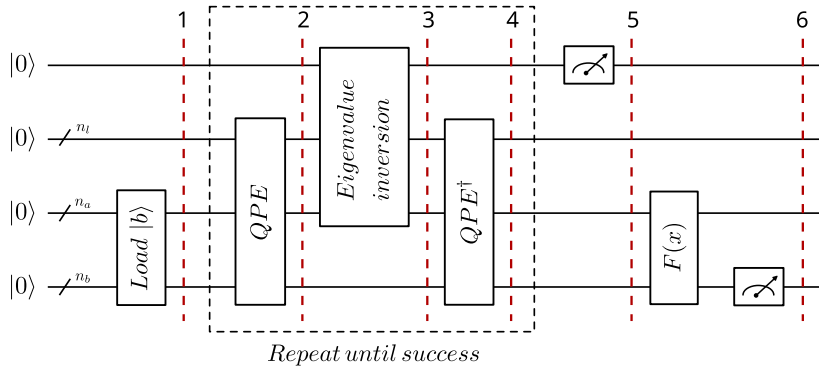

Fig. A.1. A scheme for the HHL circuit, as reported from qiskit documentation.

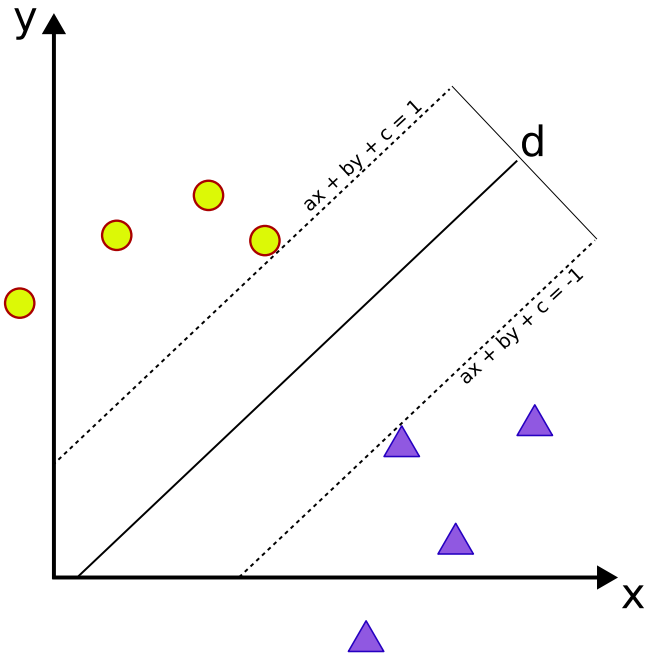

Fig. B.2. The distance $d = 2/\|\mathbf{w}\|$.

$$\sum_{j=0}^{2^m-1} b_j|\lambda_j\rangle|u_j\rangle|0\rangle \longrightarrow \sum_{j=0}^{2^m-1} b_j|\lambda_j\rangle|u_j\rangle\left[\sqrt{1-\frac{\Lambda^2}{\lambda_j^2}}|0\rangle+\frac{\Lambda}{\lambda_j}|1\rangle\right] \tag{A.6}$$

with $|\Lambda| < \min_j \lambda_j$. Applying the inverse for the QPE yields

$$\sum_{j=0}^{2^m-1} b_j|0\rangle|u_j\rangle\left[\sqrt{1-\frac{\Lambda^2}{\lambda_j^2}}|0\rangle+\frac{\Lambda}{\lambda_j}|1\rangle\right] \tag{A.7}$$

Measuring in $|1\rangle$ the ancillary qubit (otherwise the algorithm needs to be run again), the global output turns to be

$$\Lambda\sum_{j=0}^{2^m-1}\frac{b_j}{\lambda_j}|0\rangle|u_j\rangle \tag{A.8}$$

Apart from a normalization factor, the final output is the state encoding $A^{-1}\mathbf{b}$ (λ_j^{-1} being the eigenvalues for A^{-1}). In case A not being Hermitian, it is always possible to fix it by building up the following matrix:

$$\tilde{A} = \begin{pmatrix} \mathbb{O} & A \\ A^\dagger & \mathbb{O} \end{pmatrix} \tag{A.9}$$

with the linear system turning to be

$$\tilde{A}\begin{pmatrix} 0 \\ \mathbf{x} \end{pmatrix} = \begin{pmatrix} \mathbf{b} \\ 0 \end{pmatrix} \tag{A.10}$$

Appendix B. Distance between two hyperplanes

In the following, we prove the distance between the $\mathbf{x}_i$ data closest to the hyperplane $\mathbf{w} \cdot x + \mathbf{b} = 0$ to be $2/\|\mathbf{w}\|$. For such points, it holds that $\mathbf{w} \cdot \mathbf{x} + b = \pm 1$. In the first place, we choose two points $\mathbf{x}_1$, $\mathbf{x}_2$ such that $\mathbf{w} \cdot \mathbf{x}_1 + b = 1$, $\mathbf{w} \cdot \mathbf{x}_2 + b = -1$ and their midpoint $\mathbf{x}_M$ to lie on the hyperplane, i.e. $\mathbf{w} \cdot \mathbf{x}_M + b = 0$, as pictured in Fig. B.2 for the 2D case. The distance d between the two points is given by the sum of the distances between the points and the hyperplane, which we call $d/2$:

$$\frac{d}{2} = \frac{|\mathbf{w} \cdot \mathbf{x}_i + c|}{\|\mathbf{w}\|} = \frac{1}{\|\mathbf{w}\|} \tag{B.11}$$

Therefore, the overall distance d is set to be

$$d = \frac{d}{2} + \frac{d}{2} = \frac{2}{\|\mathbf{w}\|} \tag{B.12}$$

Data availability

No data was used for the research described in the article.

References

[1] A. Barenco, et al., Elementary gates for quantum computation, Phys. Rev. A 52 (5) (1995) 3457.

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

[3] S. Morita, H. Nishimori, Mathematical foundation of quantum annealing, J. Math. Phys. 49 (12) (2008) 125210.

[4] W. Van Dam, M. Mosca, U. Vazirani, How powerful is adiabatic quantum computation? in: Proceedings 42nd IEEE Symposium on Foundations of Computer Science, IEEE, 2001, pp. 279–287.

[5] M.J. Hoban, et al., Measurement-based classical computation, Phys. Rev. Lett. 112 (14) (2014) 140505.

[6] E. Pius, Automatic parallelisation of quantum circuits using the measurement based quantum computing model, in: High Performance Computing, 2010.

[7] S. Corli, E. Prati, An efficient algebraic representation for graph states for measurement-based quantum computing, in: 2022 IEEE International Conference on Rebooting Computing, ICRC, 2022, pp. 1–6.

[8] D. Kopczyk, Quantum machine learning for data scientists, 2018, arXiv preprint arXiv:1804.10068.

[9] O. Pfister, Continuous-variable quantum computing in the quantum optical frequency comb, J. Phys. B: At. Mol. Opt. Phys. 53 (1) (2019) 012001.

[10] N. Killoran, T.R. Bromley, J.M. Arrazola, M. Schuld, N. Quesada, S. Lloyd, Continuous-variable quantum neural networks, Phys. Rev. Res. 1 (3) (2019) 033063.

[11] K. Kreis, P. van Loock, Classifying, quantifying, and witnessing qudit-qumode hybrid entanglement, Phys. Rev. A 85 (3) (2012) 032307.

[12] V.M. Kendon, K. Nemoto, W.J. Munro, Quantum analogue computing, Phil. Trans. R. Soc. A 368 (1924) (2010) 3609–3620.

[13] H. Häffner, C.F. Roos, R. Blatt, Quantum computing with trapped ions, Phys. Rep. 469 (4) (2008) 155–203.

[14] R. Blatt, C.F. Roos, Quantum simulations with trapped ions, Nat. Phys. 8 (4) (2012) 277–284.

[15] J.L. O'brien, Optical quantum computing, Science 318 (5856) (2007) 1567–1570.

[16] D.P. García, J. Cruz-Benito, F.J. García-Peñalvo, Systematic literature review: Quantum machine learning and its applications, 2022, arXiv preprint arXiv:2201.04093.

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

[18] Y. Zhang, Q. Ni, Recent advances in quantum machine learning, Quantum Eng. 2 (1) (2020) e34.

[19] A. Zeguendry, Z. Jarir, M. Quafafou, Quantum machine learning: A review and case studies, Entropy 25 (2) (2023) 287.

[20] S. Jerbi, L.J. Fiderer, H. Poulsen Nautrup, J.M. Kübler, H.J. Briegel, V. Dunjko, Quantum machine learning beyond kernel methods, Nature Commun. 14 (1) (2023) 1–8.

[21] F. Tacchino, C. Macchiavello, D. Gerace, D. Bajoni, An artificial neuron implemented on an actual quantum processor, npj Quantum Inform. 5 (1) (2019) 1–8.

[22] D.H. Useche, A. Giraldo-Carvajal, H.M. Zuluaga-Bucheli, J.A. Jaramillo-Villegas, F.A. González, Quantum measurement classification with qudits, Quantum Inf. Process. 21 (1) (2022) 1–12.

[23] D. Herr, B. Obert, M. Rosenkranz, Anomaly detection with variational quantum generative adversarial networks, Quantum Sci. Technol. 6 (4) (2021) 045004.

[24] L. Moro, E. Prati, Anomaly detection speed-up by quantum restricted Boltzmann machines, Commun. Phys. 6 (1) (2023) 269.

[25] A.W. Harrow, A. Hassidim, S. Lloyd, Quantum algorithm for linear systems of equations, Phys. Rev. Lett. 103 (15) (2009) 150502.

[26] F. Albarrán-Arriagada, J.C. Retamal, E. Solano, L. Lamata, Measurement-based adaptation protocol with quantum reinforcement learning, Phys. Rev. A 98 (4) (2018) 042315.

[27] E. Payares, J. Martinez-Santos, Quantum machine learning for intrusion detection of distributed denial of service attacks: a comparative overview, Quantum Comput. Commun. Simul. 11699 (2021) 35–43.

[28] H. Wang, W. Wang, Y. Liu, B. Alidaee, Integrating machine learning algorithms with quantum annealing solvers for online fraud detection, IEEE Access 10 (2022) 75908–75917.

[29] N. Liu, P. Rebentrost, Quantum machine learning for quantum anomaly detection, Phys. Rev. A 97 (4) (2018) 042315.

[30] H. Suryotrisongko, Y. Musashi, Evaluating hybrid quantum-classical deep learning for cybersecurity botnet DGA detection, Procedia Comput. Sci. 197 (2022) 223–229.

[31] V.S. Ngairangbam, M. Spannowsky, M. Takeuchi, Anomaly detection in high-energy physics using a quantum autoencoder, Phys. Rev. D 105 (9) (2022) 095004.

[32] K.A. Woźniak, V. Belis, E. Puljak, P. Barkoutsos, G. Dissertori, M. Grossi, M. Pierini, F. Reiter, I. Tavernelli, S. Vallecorsa, Quantum anomaly detection in the latent space of proton collision events at the LHC, 2023, arXiv preprint arXiv:2301.10780.

[33] M. Abedi, G.-H. Norouzi, A. Bahroudi, Support vector machine for multi-classification of mineral prospectivity areas, Comput. Geosci. 46 (2012) 272–283.

[34] S.L. Wu, S. Sun, W. Guan, C. Zhou, J. Chan, C.L. Cheng, T. Pham, Y. Qian, A.Z. Wang, R. Zhang, et al., Application of quantum machine learning using the quantum kernel algorithm on high energy physics analysis at the LHC, Phys. Rev. Res. 3 (3) (2021) 033221.

[35] J. Schuhmacher, L. Boggia, V. Belis, E. Puljak, M. Grossi, M. Pierini, S. Vallecorsa, F. Tacchino, P. Barkoutsos, I. Tavernelli, Unravelling physics beyond the standard model with classical and quantum anomaly detection, Mach. Learn.: Sci. Technol. 4 (4) (2023) 045031.

[36] M. Davy, S. Godsill, Detection of abrupt spectral changes using support vector machines an application to audio signal segmentation, in: 2002 IEEE International Conference on Acoustics, Speech, and Signal Processing, vol. 2, IEEE, 2002, pp. II–1313.

[37] Z. Chai, Y. Liu, M. Wang, Y. Guo, F. Shi, Z. Li, Y. Wang, J. Du, Quantum anomaly detection with a spin processor in diamond, Adv. Quantum Technol. (2024) 2300385.

[38] I.H. Sarker, Deep cybersecurity: a comprehensive overview from neural network and deep learning perspective, SN Comput. Sci. 2 (3) (2021) 1–16.

[39] Y. Li, Q. Liu, A comprehensive review study of cyber-attacks and cyber security; Emerging trends and recent developments, Energy Rep. 7 (2021) 8176–8186.

[40] M. Ravinder, V. Kulkarni, A review on cyber security and anomaly detection perspectives of smart grid, in: 2023 5th International Conference on Smart Systems and Inventive Technology, ICSSIT, IEEE, 2023, pp. 692–697.

[41] F. Wang, G. Chai, Q. Li, C. Wang, An efficient deep unsupervised domain adaptation for unknown malware detection, Symmetry 14 (2) (2022) 296.

[42] A. Ayodeji, Y.-k. Liu, N. Chao, L.-q. Yang, A new perspective towards the development of robust data-driven intrusion detection for industrial control systems, Nucl. Eng. Technol. 52 (12) (2020) 2687–2698.

[43] Á.L.P. Gómez, L.F. Maimó, A.H. Celdrán, F.J.G. Clemente, SUSAN: A Deep Learning based anomaly detection framework for sustainable industry, Sustain. Comput.: Inform. Syst. (2023) 100842.

[44] E. Tufan, C. Tezcan, C. Acartürk, Anomaly-based intrusion detection by machine learning: A case study on probing attacks to an institutional network, IEEE Access 9 (2021) 50078–50092.

[45] N. Wirkuttis, H. Klein, Artificial intelligence in cybersecurity, Cyber Intell. Secur. 1 (1) (2017) 103–119.

[46] T.C. Truong, I. Zelinka, J. Plucar, M. Čandík, V. Šulc, Artificial intelligence and cybersecurity: Past, presence, and future, in: Artificial Intelligence and Evolutionary Computations in Engineering Systems, Springer, 2020, pp. 351–363.

[47] Z. Yuan, Y. Lu, Z. Wang, Y. Xue, Droid-sec: deep learning in android malware detection, in: Proceedings of the 2014 ACM Conference on SIGCOMM, 2014, pp. 371–372.

[48] D. Yuxin, Z. Siyi, Malware detection based on deep learning algorithm, Neural Comput. Appl. 31 (2) (2019) 461–472.

[49] R. Vinayakumar, M. Alazab, K. Soman, P. Poornachandran, S. Venkatraman, Robust intelligent malware detection using deep learning, IEEE Access 7 (2019) 46717–46738.

[50] C. Bitter, D.A. Elizondo, T. Watson, Application of artificial neural networks and related techniques to intrusion detection, in: The 2010 International Joint Conference on Neural Networks, IJCNN, IEEE, 2010, pp. 1–8.

[51] Y. Kou, C.-T. Lu, S. Sirwongwattana, Y.-P. Huang, Survey of fraud detection techniques, in: IEEE International Conference on Networking, Sensing and Control, 2004, vol. 2, IEEE, 2004, pp. 749–754.

[52] Z. Ahmad, A. Shahid Khan, C. Wai Shiang, J. Abdullah, F. Ahmad, Network intrusion detection system: A systematic study of machine learning and deep learning approaches, Trans. Emerg. Telecommun. Technol. 32 (1) (2021) e4150.

[53] M. Sewak, S.K. Sahay, H. Rathore, Deep reinforcement learning for cybersecurity threat detection and protection: A review, in: International Conference on Secure Knowledge Management in Artificial Intelligence Era, Springer, 2021, pp. 51–72.

[54] N.N.A. Sjarif, S. Chuprat, M.N. Mahrin, N.A. Ahmad, A. Ariffin, F.M. Senan, N.A. Zamani, A. Saupi, Endpoint detection and response: Why use machine learning? in: 2019 International Conference on Information and Communication Technology Convergence, ICTC, IEEE, 2019, pp. 283–288.

[55] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, M. Riedmiller, Deterministic policy gradient algorithms, in: International Conference on Machine Learning, Pmlr, 2014, pp. 387–395.

[56] J. Schulman, S. Levine, P. Abbeel, M. Jordan, P. Moritz, Trust region policy optimization, in: International Conference on Machine Learning, PMLR, 2015, pp. 1889–1897.

[57] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, Proximal policy optimization algorithms, 2017, arXiv preprint arXiv:1707.06347.

[58] J. Schulman, P. Moritz, S. Levine, M. Jordan, P. Abbeel, High-dimensional continuous control using generalized advantage estimation, 2015, arXiv preprint arXiv:1506.02438.

[59] C. Campbell, K. Bennett, A linear programming approach to novelty detection, in: Advances in Neural Information Processing Systems, vol. 13, 2000.

[60] T. Petsche, A. Marcantonio, C. Darken, S. Hanson, G. Kuhn, N. Santoso, A neural network autoassociator for induction motor failure prediction, in: Advances in Neural Information Processing Systems, vol. 8, 1995.

[61] R. Fujimaki, T. Yairi, K. Machida, An approach to spacecraft anomaly detection problem using kernel feature space, in: Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining, 2005, pp. 401–410.

[62] T. Brotherton, T. Johnson, Anomaly detection for advanced military aircraft using neural networks, in: 2001 IEEE Aerospace Conference Proceedings (Cat. No. 01TH8542), vol. 6, IEEE, 2001, pp. 3113–3123.

[63] L.M. Manevitz, M. Yousef, One-class SVMs for document classification, J. Mach. Learn. Res. 2 (Dec) (2001) 139–154.

[64] M. Augusteijn, B. Folkert, Neural network classification and novelty detection, Int. J. Remote Sens. 23 (14) (2002) 2891–2902.

[65] S. Singh, M. Markou, An approach to novelty detection applied to the classification of image regions, IEEE Trans. Knowl. Data Eng. 16 (4) (2004) 396–407.

[66] A.N. Srivastava, Enabling the discovery of recurring anomalies in aerospace problem reports using high-dimensional clustering techniques, in: 2006 IEEE Aerospace Conference, IEEE, 2006, pp. 17–pp.

[67] A.N. Srivastava, B. Zane-Ulman, Discovering recurring anomalies in text reports regarding complex space systems, in: 2005 IEEE Aerospace Conference, IEEE, 2005, pp. 3853–3862.

[68] R. Gowtham, I. Krishnamurthi, A comprehensive and efficacious architecture for detecting phishing webpages, Comput. Secur. 40 (2014) 23–37.

[69] Y. Yu, J. Long, Z. Cai, Network intrusion detection through stacking dilated convolutional autoencoders, Secur. Commun. Netw. 2017 (2017).

[70] D. Deutsch, Quantum theory, the Church–Turing principle and the universal quantum computer, Proc. R. Soc. A 400 (1818) (1985) 97–117.

[71] A. Montanaro, Quantum algorithms: an overview, npj Quantum Inform. 2 (1) (2016) 1–8.

[72] J. Jäger, R.V. Krems, Universal expressiveness of variational quantum classifiers and quantum kernels for support vector machines, Nature Commun. 14 (1) (2023) 576.

[73] T.F. Rønnow, Z. Wang, J. Job, S. Boixo, S.V. Isakov, D. Wecker, J.M. Martinis, D.A. Lidar, M. Troyer, Defining and detecting quantum speedup, Science 345 (6195) (2014) 420–424.

[74] M. Schuld, N. Killoran, Quantum machine learning in feature Hilbert spaces, Phys. Rev. Lett. 122 (4) (2019) 040504.

[75] V. Havlíček, A.D. Córcoles, K. Temme, A.W. Harrow, A. Kandala, J.M. Chow, J.M. Gambetta, Supervised learning with quantum-enhanced feature spaces, Nature 567 (7747) (2019) 209–212.

[76] G. Agliardi, E. Prati, Optimal tuning of quantum generative adversarial networks for multivariate distribution loading, Quantum Rep. 4 (1) (2022) 75–105.

[77] B. Duan, J. Yuan, C.-H. Yu, J. Huang, C.-Y. Hsieh, A survey on HHL algorithm: From theory to application in quantum machine learning, Phys. Lett. A 384 (24) (2020) 126595.

[78] V. Giovannetti, S. Lloyd, L. Maccone, Quantum random access memory, Phys. Rev. Lett. 100 (16) (2008) 160501.

[79] S. Jaques, A.G. Rattew, QRAM: A survey and critique, 2023, arXiv preprint arXiv:2305.10310.

[80] A. Pérez-Salinas, A. Cervera-Lierta, E. Gil-Fuster, J.I. Latorre, Data re-uploading for a universal quantum classifier, Quantum 4 (2020) 226.

[81] D. Konar, S. Bhattacharyya, B.K. Panigrahi, E.C. Behrman, Qutrit-inspired fully self-supervised shallow quantum learning network for brain tumor segmentation, IEEE Trans. Neural Netw. Learn. Syst. (2021).

[82] D.P. Srivastava, V. Sahni, P.S. Satsangi, Modelling microtubules in the brain as n-qudit quantum hopfield network and beyond, Int. J. Gen. Syst. 45 (1) (2016) 41–54.

[83] Y. Wang, Z. Hu, B.C. Sanders, S. Kais, Qudits and high-dimensional quantum computing, Front. Phys. 8 (2020) 589504.

[84] S. Bravyi, A. Kliesch, R. Koenig, E. Tang, Hybrid quantum-classical algorithms for approximate graph coloring, Quantum 6 (2022) 678.

[85] V. Danos, E. Kashefi, P. Panangaden, The measurement calculus, J. ACM 54 (2) (2007) 8–es.

[86] S. Morita, H. Nishimori, Convergence of quantum annealing with real-time Schrödinger dynamics, J. Phys. Soc. Japan 76 (6) (2007) 064002.

[87] 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 574 (7779) (2019) 505–510.

[88] H.-S. Zhong, Y.-H. Deng, J. Qin, H. Wang, M.-C. Chen, L.-C. Peng, Y.-H. Luo, D. Wu, S.-Q. Gong, H. Su, et al., Phase-programmable gaussian boson sampling using stimulated squeezed light, Phys. Rev. Lett. 127 (18) (2021) 180502.

[89] Y. Wu, W.-S. Bao, S. Cao, F. Chen, M.-C. Chen, X. Chen, T.-H. Chung, H. Deng, Y. Du, D. Fan, et al., Strong quantum computational advantage using a superconducting quantum processor, Phys. Rev. Lett. 127 (18) (2021) 180501.

[90] L.S. Madsen, F. Laudenbach, M.F. Askarani, F. Rortais, T. Vincent, J.F. Bulmer, F.M. Miatto, L. Neuhaus, L.G. Helt, M.J. Collins, et al., Quantum computational advantage with a programmable photonic processor, Nature 606 (7912) (2022) 75–81.

[91] T.S. Humble, A. McCaskey, D.I. Lyakh, M. Gowrishankar, A. Frisch, T. Monz, Quantum computers for high-performance computing, IEEE Micro 41 (5) (2021) 15–23.

[92] V. Bergholm, J. Izaac, M. Schuld, C. Gogolin, S. Ahmed, V. Ajith, M.S. Alam, G. Alonso-Linaje, B. AkashNarayanan, A. Asadi, et al., Pennylane: Automatic differentiation of hybrid quantum-classical computations, 2018, arXiv preprint arXiv:1811.04968.

[93] A. Peruzzo, J. McClean, P. Shadbolt, M.-H. Yung, X.-Q. Zhou, P.J. Love, A. Aspuru-Guzik, J.L. O'brien, A variational eigenvalue solver on a photonic quantum processor, Nat. Commun. 5 (1) (2014) 4213.

[94] E. Farhi, J. Goldstone, S. Gutmann, A quantum approximate optimization algorithm, 2014, arXiv preprint arXiv:1411.4028.

[95] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, A. Lerer, Automatic differentiation in pytorch, 2017.

[96] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G.S. Corrado, A. Davis, J. Dean, M. Devin, et al., Tensorflow: Large-scale machine learning on heterogeneous distributed systems, 2016, arXiv preprint arXiv:1603.04467.

[97] N. Heurtel, A. Fyrillas, G. de Gliniasty, R. Le Bihan, S. Malherbe, M. Pailhas, E. Bertasi, B. Bourdoncle, P.-E. Emeriau, R. Mezher, et al., Perceval: A software platform for discrete variable photonic quantum computing, Quantum 7 (2023) 931.

[98] T. Vincent, L.J. O'Riordan, M. Andrenkov, J. Brown, N. Killoran, H. Qi, I. Dhand, Jet: Fast quantum circuit simulations with parallel task-based tensor-network contraction, Quantum 6 (2022) 709.

[99] K. Hornik, Approximation capabilities of multilayer feedforward networks, Neural Netw. 4 (2) (1991) 251–257.

[100] J. Kübler, S. Buchholz, B. Schölkopf, The inductive bias of quantum kernels, Adv. Neural Inf. Process. Syst. 34 (2021) 12661–12673.

[101] S. Jerbi, C. Gyurik, S.C. Marshall, R. Molteni, V. Dunjko, Shadows of quantum machine learning, Nature Commun. 15 (1) (2024) 5676.

[102] Y. Suzuki, Q. Gao, K.C. Pradel, K. Yasuoka, N. Yamamoto, Natural quantum reservoir computing for temporal information processing, Sci. Rep. 12 (1) (2022) 1353.

[103] Y. Huang, H. Lei, X. Li, Q. Zhu, W. Ren, X. Liu, Quantum generative model with variable-depth circuit, CMC-Comput. Mater. Continua 65 (1) (2020) 445–458.

[104] A. Chalumuri, R. Kune, B. Manoj, A hybrid classical-quantum approach for multi-class classification, Quantum Inf. Process. 20 (3) (2021) 119.

[105] S. Li, K. Jia, Y. Wen, T. Liu, D. Tao, Orthogonal deep neural networks, IEEE Trans. Pattern Anal. Mach. Intell. 43 (4) (2019) 1352–1368.

[106] Y. Li, R.-G. Zhou, R. Xu, J. Luo, W. Hu, A quantum deep convolutional neural network for image recognition, Quantum Sci. Technol. 5 (4) (2020) 044003.

[107] M. Srikumar, C.D. Hill, L.C. Hollenberg, Clustering and enhanced classification using a hybrid quantum autoencoder, Quantum Sci. Technol. 7 (1) (2021) 015020.

[108] B.-Q. Chen, X.-F. Niu, Quantum neural network with improved quantum learning algorithm, Internat. J. Theoret. Phys. 59 (2020) 1978–1991.

[109] M. Maronese, E. Prati, A continuous rosenblatt quantum perceptron, Int. J. Quantum Inf. 19 (04) (2021) 2140002.

[110] J.R. McClean, S. Boixo, V.N. Smelyanskiy, R. Babbush, H. Neven, Barren plateaus in quantum neural network training landscapes, Nat. Commun. 9 (1) (2018) 1–6.

[111] E. Grant, L. Wossnig, M. Ostaszewski, M. Benedetti, An initialization strategy for addressing barren plateaus in parametrized quantum circuits, Quantum 3 (2019) 214.

[112] A. Pesah, M. Cerezo, S. Wang, T. Volkoff, A.T. Sornborger, P.J. Coles, Absence of barren plateaus in quantum convolutional neural networks, Phys. Rev. X 11 (4) (2021) 041011.

[113] Y. Cao, G.G. Guerreschi, A. Aspuru-Guzik, Quantum neuron: an elementary building block for machine learning on quantum computers, 2017, arXiv preprint arXiv:1711.11240.

[114] W. Hu, Towards a real quantum neuron, Nat. Sci. 10 (3) (2018) 99–109.

[115] A.J. da Silva, W.R. de Oliveira, T.B. Ludermir, Weightless neural network parameters and architecture selection in a quantum computer, Neurocomputing 183 (2016) 13–22.

[116] N. Matsui, H. Nishimura, T. Isokawa, Qubit neural network: Its performance and applications, in: Complex-Valued Neural Networks: Utilizing High-Dimensional Parameters, IGI Global, 2009, pp. 325–351.

[117] A.J. da Silva, T.B. Ludermir, W.R. de Oliveira, Quantum perceptron over a field and neural network architecture selection in a quantum computer, Neural Netw. 76 (2016) 55–64.

[118] D. Ventura, T. Martinez, Quantum associative memory, Inform. Sci. 124 (1–4) (2000) 273–296.

[119] A.J. da Silva, R.L.F. de Oliveira, Neural networks architecture evaluation in a quantum computer, in: 2017 Brazilian Conference on Intelligent Systems, BRACIS, IEEE, 2017, pp. 163–168.

[120] M. Schuld, I. Sinayskiy, F. Petruccione, The quest for a quantum neural network, Quantum Inf. Process. 13 (11) (2014) 2567–2586.

[121] C. Shao, A quantum model for multilayer perceptron, 2018, arXiv preprint arXiv:1808.10561.

[122] A. Kamruzzaman, Y. Alhwaiti, A. Leider, C.C. Tappert, Quantum deep learning neural networks, in: Future of Information and Communication Conference, Springer, 2019, pp. 299–311.

[123] F. Tacchino, P. Barkoutsos, C. Macchiavello, I. Tavernelli, D. Gerace, D. Bajoni, Quantum implementation of an artificial feed-forward neural network, Quantum Sci. Technol. 5 (4) (2020) 044010.

[124] R. Molteni, C. Destri, E. Prati, Optimization of the memory reset rate of a quantum echo-state network for time sequential tasks, Phys. Lett. A 465 (2023) 128713.

[125] M. Pritt, G. Chern, Satellite image classification with deep learning, in: 2017 IEEE Applied Imagery Pattern Recognition Workshop, AIPR, IEEE, 2017, pp. 1–7.

[126] F. Tacchino, P.K. Barkoutsos, C. Macchiavello, D. Gerace, I. Tavernelli, D. Bajoni, Variational learning for quantum artificial neural networks, in: 2020 IEEE International Conference on Quantum Computing and Engineering, QCE, IEEE, 2020, pp. 130–136.

[127] P. Rebentrost, M. Mohseni, S. Lloyd, Quantum support vector machine for big data classification, Phys. Rev. Lett. 113 (13) (2014) 130503.

[128] A. Delilbasic, G. Cavallaro, M. Willsch, F. Melgani, M. Riedel, K. Michielsen, Quantum support vector machine algorithms for remote sensing data classification, in: 2021 IEEE International Geoscience and Remote Sensing Symposium IGARSS, IEEE, 2021, pp. 2608–2611.

[129] J. Cervantes, F. Garcia-Lamont, L. Rodríguez-Mazahua, A. Lopez, A comprehensive survey on support vector machine classification: Applications, challenges and trends, Neurocomputing 408 (2020) 189–215.

[130] L. Zhang, W. Zhou, L. Jiao, Wavelet support vector machine, IEEE Trans. Syst. Man Cybern. B 34 (1) (2004) 34–39.

[131] C. Ding, T.-Y. Bao, H.-L. Huang, Quantum-inspired support vector machine, IEEE Trans. Neural Netw. Learn. Syst. (2021).

[132] Z. Li, X. Liu, N. Xu, J. Du, Experimental realization of a quantum support vector machine, Phys. Rev. Lett. 114 (14) (2015) 140504.

[133] M. Lazzarin, D.E. Galli, E. Prati, Multi-class quantum classifiers with tensor network circuits for quantum phase recognition, Phys. Lett. A 434 (2022) 128056.

[134] H.-J. Jeon, S. Lang, C. Vogel, R. Behrens, Anomaly detection from image classification, in: 2024 9th International Conference on Control and Robotics Engineering, ICCRE, IEEE, 2024, pp. 377–381.

[135] Q. Wei, Y. Ren, R. Hou, B. Shi, J.Y. Lo, L. Carin, Anomaly detection for medical images based on a one-class classification, in: Medical Imaging 2018: Computer-Aided Diagnosis, vol. 10575, SPIE, 2018, pp. 375–380.

[136] J. Liu, G. Xie, J. Wang, S. Li, C. Wang, F. Zheng, Y. Jin, Deep industrial image anomaly detection: A survey, Mach. Intell. Res. 21 (1) (2024) 104–135.

[137] M. Maronese, C. Destri, E. Prati, Quantum activation functions for quantum neural networks, Quantum Inf. Process. 21 (4) (2022) 1–24.

[138] G.E. Hinton, Training products of experts by minimizing contrastive divergence, Neural Comput. 14 (8) (2002) 1771–1800.

[139] T. Tieleman, Training restricted Boltzmann machines using approximations to the likelihood gradient, in: Proceedings of the 25th International Conference on Machine Learning, 2008, pp. 1064–1071.

[140] L. Ning, R. Pittman, X. Shen, LCD: A fast contrastive divergence based algorithm for restricted Boltzmann machine, Neural Netw. 108 (2018) 399–410.

[141] M. Maronese, L. Moro, L. Rocutto, E. Prati, Quantum compiling, Quantum Comput. Environ. (2022) 39–74.

[142] L. Rocutto, C. Destri, E. Prati, Quantum semantic learning by reverse annealing of an adiabatic quantum computer, Adv. Quantum Technol. 4 (2) (2021) 2000133.

[143] L. Rocutto, E. Prati, A complete restricted Boltzmann machine on an adiabatic quantum computer, Int. J. Quantum Inf. 19 (04) (2021) 2141003.

[144] L. Rocutto, D. Noè, L. Moro, E. Prati, Fast training of fully-connected Boltzmann Machines on an Adiabatic Quantum Computer, QCE, in: 2023 IEEE International Conference on Quantum Computing and Engineering, vol. 1, IEEE, 2023, pp. 630–635.

[145] F.A. González, V. Vargas-Calderón, H. Vinck-Posada, Classification with quantum measurements, J. Phys. Soc. Japan 90 (4) (2021) 044002.

[146] D.J. Sutherland, J. Schneider, On the error of random Fourier features, 2015, arXiv preprint arXiv:1506.02785.

[147] G. Barrué, T. Quertier, Quantum machine learning for malware classification, 2023, arXiv preprint arXiv:2305.09674.

[148] A. Kariya, B.K. Behera, Investigation of quantum support vector machine for classification in NISQ era, 2021, arXiv preprint arXiv:2112.06912.

[149] B.E. Boser, I.M. Guyon, V.N. Vapnik, A training algorithm for optimal margin classifiers, in: Proceedings of the Fifth Annual Workshop on Computational Learning Theory, 1992, pp. 144–152.

[150] J.A. Suykens, J. Vandewalle, Least squares support vector machine classifiers, Neural Process. Lett. 9 (3) (1999) 293–300.

[151] H. Cao, X. Guo, M. Laurière, Connecting GANs, MFGs, and OT, 2020, arXiv preprint arXiv:2002.04112.

[152] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, A.C. Courville, Improved training of Wasserstein gans, in: Advances in Neural Information Processing Systems, vol. 30, 2017.

[153] S. Aaronson, A. Arkhipov, The computational complexity of linear optics, in: Proceedings of the Forty-Third Annual ACM Symposium on Theory of Computing, 2011, pp. 333–342.

[154] M.J. Bremner, A. Montanaro, D.J. Shepherd, Average-case complexity versus approximate simulation of commuting quantum computations, Phys. Rev. Lett. 117 (8) (2016) 080501.

[155] R. Sweke, J.-P. Seifert, D. Hangleiter, J. Eisert, On the quantum versus classical learnability of discrete distributions, Quantum 5 (2021) 417.

[156] S. Aaronson, Read the fine print, Nat. Phys. 11 (4) (2015) 291–293.

[157] S. Alvi, C. Bauer, B. Nachman, Quantum anomaly detection for collider physics, 2022, arXiv preprint arXiv:2206.08391.

[158] J.-M. Liang, S.-Q. Shen, M. Li, L. Li, Quantum anomaly detection with density estimation and multivariate Gaussian distribution, Phys. Rev. A 99 (5) (2019) 052310.

[159] M. Guo, H. Liu, Y. Li, W. Li, F. Gao, S. Qin, Q. Wen, Quantum algorithms for anomaly detection using amplitude estimation, Phys. A 604 (2022) 127936.

[160] O. Kyriienko, E.B. Magnusson, Unsupervised quantum machine learning for fraud detection, 2022, arXiv preprint arXiv:2208.01203.

[161] M.G. Omran, A.P. Engelbrecht, A. Salman, An overview of clustering methods, Intell. Data Anal. 11 (6) (2007) 583–605.

[162] W. Li, X.-Y. Zhang, H. Bao, Q. Wang, Z. Li, Robust network traffic identification with graph matching, Comput. Netw. 218 (2022) 109368.

[163] S. Lagraa, M. Husák, H. Seba, S. Vuppala, R. State, M. Ouedraogo, A review on graph-based approaches for network security monitoring and botnet detection, Int. J. Inf. Secur. (2023) 1–22.

[164] S. Lagraa, J. François, A. Lahmadi, M. Miner, C. Hammerschmidt, R. State, BotGM: Unsupervised graph mining to detect botnets in traffic flows, in: 2017 1st Cyber Security in Networking Conference, CSNet, IEEE, 2017, pp. 1–8.

[165] P. Festa, P.M. Pardalos, M.G. Resende, C.C. Ribeiro, Randomized heuristics for the MAX-CUT problem, Optim. Methods Softw. 17 (6) (2002) 1033–1058.

[166] S. Burer, R.D. Monteiro, Y. Zhang, Rank-two relaxation heuristics for max-cut and other binary quadratic programs, SIAM J. Optim. 12 (2) (2002) 503–521.

[167] L. Zhou, S.-T. Wang, S. Choi, H. Pichler, M.D. Lukin, Quantum approximate optimization algorithm: Performance, mechanism, and implementation on near-term devices, Phys. Rev. X 10 (2) (2020) 021067.

[168] C.H. Ding, X. He, H. Zha, M. Gu, H.D. Simon, A min-max cut algorithm for graph partitioning and data clustering, in: Proceedings 2001 IEEE International Conference on Data Mining, IEEE, 2001, pp. 107–114.

[169] D. Beaulieu, A. Pham, Max-cut clustering utilizing warm-start qaoa and ibm runtime, 2021, arXiv preprint arXiv:2108.13464.

[170] M. Proietti, F. Cerocchi, M. Dispenza, Native measurement-based quantum approximate optimization algorithm applied to the Max K-Cut problem, Phys. Rev. A 106 (2) (2022) 022437.

[171] S. Corli, D. Dragoni, M. Proietti, M. Dispenza, C. Cavazzoni, E. Prati, A Max K-Cut implementation for QAOA in the measurement based quantum computing formalism, QCE, in: 2023 IEEE International Conference on Quantum Computing and Engineering, vol. 2, IEEE, 2023, pp. 284–285.

[172] D. An, L. Lin, Quantum linear system solver based on time-optimal adiabatic quantum computing and quantum approximate optimization algorithm, ACM Trans. Quantum Comput. 3 (2) (2022) 1–28.

[173] Y. Sun, J.-Y. Zhang, M.S. Byrd, L.-A. Wu, Adiabatic quantum simulation using trotterization, 2018, arXiv preprint arXiv:1805.11568.

[174] M. Streif, M. Leib, Comparison of QAOA with quantum and simulated annealing, 2019, arXiv preprint arXiv:1901.01903.

[175] J.R. Weggemans, A. Urech, A. Rausch, R. Spreeuw, R. Boucherie, F. Schreck, K. Schoutens, J. Minář, F. Speelman, Solving correlation clustering with QAOA and a Rydberg qudit system: a full-stack approach, Quantum 6 (2022) 687.

[176] J. Choi, J. Kim, A tutorial on quantum approximate optimization algorithm (QAOA): Fundamentals and applications, in: 2019 International Conference on Information and Communication Technology Convergence, ICTC, IEEE, 2019, pp. 138–142.
[177] P.C. Lotshaw, T.S. Humble, R. Herrman, J. Ostrowski, G. Siopsis, Empirical performance bounds for quantum approximate optimization, Quantum Inf. Process. 20 (2021) 1–32.
[178] X. Lee, Y. Saito, D. Cai, N. Asai, Parameters fixing strategy for quantum approximate optimization algorithm, in: 2021 IEEE International Conference on Quantum Computing and Engineering, QCE, IEEE, 2021, pp. 10–16.
[179] J. Wurtz, D. Lykov, Fixed-angle conjectures for the quantum approximate optimization algorithm on regular MaxCut graphs, Phys. Rev. A 104 (5) (2021) 052419.
[180] Y. Pan, Y. Tong, Y. Yang, Automatic depth optimization for a quantum approximate optimization algorithm, Phys. Rev. A 105 (3) (2022) 032433.
[181] Z. Wang, S. Hadfield, Z. Jiang, E.G. Rieffel, Quantum approximate optimization algorithm for MaxCut: A fermionic view, Phys. Rev. A 97 (2) (2018) 022304.
[182] S.H. Sack, M. Serbyn, Quantum annealing initialization of the quantum approximate optimization algorithm, Quantum 5 (2021) 491.
[183] B. Barak, A. Moitra, R. O'Donnell, P. Raghavendra, O. Regev, D. Steurer, L. Trevisan, A. Vijayaraghavan, D. Witmer, J. Wright, Beating the random assignment on constraint satisfaction problems of bounded degree, 2015, arXiv preprint arXiv:1505.03424.
[184] F.A. Cárdenas-López, L. Lamata, J.C. Retamal, E. Solano, Multiqubit and multilevel quantum reinforcement learning with quantum technologies, PLoS One 13 (7) (2018) e0200455.
[185] N. Mishra, M. Kapil, H. Rakesh, A. Anand, N. Mishra, A. Warke, S. Sarkar, S. Dutta, S. Gupta, A. Prasad Dash, et al., Quantum machine learning: A review and current status, Data Manag. Anal. Innov. (2021) 101–145.
[186] J.D. Martín-Guerrero, L. Lamata, Quantum machine learning: A tutorial, Neurocomputing 470 (2022) 457–461.
[187] S.Y.-C. Chen, C.-M. Huang, C.-W. Hsing, H.-S. Goan, Y.-J. Kao, Variational quantum reinforcement learning via evolutionary optimization, Mach. Learn.: Sci. Technol. 3 (1) (2022) 015025.
[188] A. Acuto, P. Barillà, L. Bozzolo, M. Conterno, M. Pavese, A. Policicchio, Variational quantum soft actor-critic for robotic arm control, 2022, arXiv preprint arXiv:2212.11681.
[189] N. Dalla Pozza, L. Buffoni, S. Martina, F. Caruso, Quantum reinforcement learning: the Maze problem, Quant. Mach. Intell. 4 (1) (2022) 1–10.
[190] E.A. Cherrat, I. Kerenidis, A. Prakash, Quantum reinforcement learning via policy iteration, 2022, arXiv preprint arXiv:2203.01889.
[191] V. Dunjko, J.M. Taylor, H.J. Briegel, Quantum-enhanced machine learning, Phys. Rev. Lett. 117 (13) (2016) 130501.
[192] D. Dong, C. Chen, H. Li, T.-J. Tarn, Quantum reinforcement learning, IEEE Trans. Syst. Man Cybern. B 38 (5) (2008) 1207–1220.
[193] S. Yu, F. Albarrán-Arriagada, J.C. Retamal, Y.-T. Wang, W. Liu, Z.-J. Ke, Y. Meng, Z.-P. Li, J.-S. Tang, E. Solano, et al., Reconstruction of a photonic qubit state with reinforcement learning, Adv. Quantum Technol. 2 (7–8) (2019) 1800074.
[194] F. Albarrán-Arriagada, J.C. Retamal, E. Solano, L. Lamata, Reinforcement learning for semi-autonomous approximate quantum eigensolver, Mach. Learn.: Sci. Technol. 1 (1) (2020) 015002.
[195] M. Tavallaee, E. Bagheri, W. Lu, A.A. Ghorbani, A detailed analysis of the KDD CUP 99 data set, in: 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications, Ieee, 2009, pp. 1–6.
[196] I. Sharafaldin, A.H. Lashkari, A.A. Ghorbani, et al., Toward generating a new intrusion detection dataset and intrusion traffic characterization, ICISSp 1 (2018) 108–116.
[197] A. Dal Pozzolo, O. Caelen, R.A. Johnson, G. Bontempi, Calibrating probability with undersampling for unbalanced classification, in: 2015 IEEE Symposium Series on Computational Intelligence, IEEE, 2015, pp. 159–166.

Prof. Enrico Prati received his Ph.D. in Physics from the University of Pisa in 2002 and did postdoctoral work at Istituto Nazionale di Fisica della Materia (Agrate Brianza, Italy). After being researcher of Consiglio Nazionale delle Riceehrche from 2006 to 2021, he is now Professor at the Dipartimento di Fisica of Università degli Studi di Milano where he chairs the Quantum Intelligence Lab. Prof. Prati's research interests are in the field of quantum computing and artificial intelligence.