communications
physics

ARTICLE

https://doi.org/10.1038/s42005-023-01390-y OPEN

Anomaly detection speed-up by quantum restricted Boltzmann machines

Lorenzo Moro[1,2] & Enrico Prati[2,3]✉

Quantum machine learning promises to revolutionize traditional machine learning by efficiently addressing hard tasks for classical computation. While claims of quantum speed-up have been announced for gate-based quantum computers and photon-based boson samplers, demonstration of an advantage by adiabatic quantum annealers (AQAs) is open. Here we quantify the computational cost and the performance of restricted Boltzmann machines (RBMs), a widely investigated machine learning model, by classical and quantum annealing. Despite the lower computational complexity of the quantum RBM being lost due to physical implementation overheads, a quantum speed-up may arise as a reduction by orders of magnitude of the computational time. By employing real-world cybersecurity datasets, we observe that the negative phase on sufficiently challenging tasks is computed up to 64 times faster by AQAs during the exploitation phase. Therefore, although a quantum speed-up highly depends on the problem's characteristics, it emerges in existing hardware on real-world data.

[1] Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Via Colombo 81, I-20133 Milano, Italy. [2] Istituto di Fotonica e Nanotecnologie, Consiglio Nazionale delle Ricerche, Piazza Leonardo da Vinci 32, I-20133 Milano, Italy. [3] Dipartimento di Fisica Aldo Pontremoli, Università degli Studi di Milano, Via Celoria 16, I-20133 Milano, Italy. ✉email: enrico.prati@unimi.it

In the last decade, machine learning (ML) techniques have become essential to find patterns in big data and to mine valuable information. Over the past half-century, the rapid progression of computing power has allowed the spawn of novel ML methods from simple linear algebraic data analysis techniques to sophisticated deep learning neural networks with billions of synaptic weights. Today, many state-of-the-art models require a significant amount of computational capacity to be trained and processed, with artificial intelligence and deep learning solutions at the top. Groundbreaking neural network models, such as GPT-3[1] and AlphaFold[2], require a massive amount of resources. Still, the computational power demand and the need for new approaches to process and undermine information faster from ever more extensive datasets are rapidly increasing.

In such a framework, applying quantum computing to ML[3–7] is a promising approach to train and exploit models, potentially in a fraction of the time and cost, without the need for power-consuming high-performance computing and data centers. However, the advantage offered by quantum computers can be quantified differently, depending on whether the focus is on increasing some ML metrics, such as either the classification score, or the precision, or the accuracy—by addressing classically intractable systems, or by reducing the training and query times, respectively. Indeed, quantum computers promise to accelerate ML algorithms from physics to chemistry[8–11], biology[12,13], finance[14,15], and material science[16], allowing unprecedented opportunities in almost any industry.

Although many algorithms have proven a theoretical advantage over classical computation[17–26], quantum computers are currently in an early stage of technological development, limited by hardware engineering constraints.

Several quantum computer technologies and computational architecture paradigms exist, each characterized by different trade-offs in terms of the number of qubits, fidelity, and speed of quantum operations. The lack of sufficient resources to implement error-correction techniques implies noisy gates and decohering qubits, which in turn strongly affect the capability of such devices to provide an edge in real-world applications. Indeed, it is unclear which applications could provide a true advantage over classical computers, given the current noisy quantum devices.

Considering the current development of quantum computer hardware, adiabatic quantum annealers (AQAs)—suitable to physically implement quantum annealing—offer a viable path to demonstrate a quantum speed-up in today's ML applications. In fact, AQA currently provides a higher number of qubits compared to any other gate-model quantum architectures, being able to address much larger datasets and problems. Although AQA is not a general-purpose quantum computer, it has many applications designed to solve combinatorial optimization problems, including chemistry and materials[27–29], scheduling[3,30,31], ML[32–34], and finance[14,15,35]. Among all the quantum ML models that could exhibit a speed-up, restricted Boltzmann machines[36] (RBMs), generative models based on neural networks, are widely investigated. Indeed, they can be efficiently trained on quantum annealers[31,37,38] and be employed for several ML tasks from data reduction[39] and feature extraction[36], to classification[40,41] and collaborative filtering[42]. Despite the high potential of the representative power, the adoption of RBMs is limited since they require a demanding computational training process, such as contrastive divergence (CD) and persistent contrastive divergence (PCD)[43]. In fact, the classical algorithm requires performing a thermalization cycle to sample states from an RBM, with a computational complexity that scales as $O(k \cdot N)$, where k is the number of thermalization steps and N are the number of RBM units[38]. Moreover, the classical procedure only approximates the correct model distribution, therefore intrinsically affecting the overall performance. On the other hand, the advent of commercial AQAs promises to address those problems, avoiding long thermalization cycles and sampling correctly distributed states by querying the QPU in principle once. If the computational cost of initializing the quantum computer is neglected, the quantum algorithm computational complexity to obtain a single sample scales as $O(1)$.

Although AQAs offer a fundamentally different way to train and query the model, providing an asymptotically faster way to sample states and perform better than conventional methods, we are currently limited by the dimension of problems that can be embedded in the quantum device. In particular, D-Wave quantum annealers, the only commercial quantum annealers available today, are physical devices restrained by technological and engineering constraints. Sampling from a D-Wave quantum machine has a computational complexity that depends on the number of the involved physical qubits and the number of samples extracted, making it non-trivial to forecast the advantage the quantum computer would offer in the future. Today, the quantitative answer to the question of what scaling edge and which range of problem RBMs can provide on realistic, noisy finite-size quantum devices still lacks.

In this work, we address those questions by separately comparing the time to sample states during the training (training time) and querying the model after the training phase (exploitation time) and the performance of both classical and quantum-RBMs (QRBMs) on standardized cybersecurity datasets. In particular, we trained an RBM as a network intrusion detection system[44–47] (NIDS) on two real-world cybersecurity datasets (NSL-KDD and CSE-CIC-IDS2018, respectively) to detect a quantum speed-up of today's hardware by employing a quantum annealer and an equivalent model trained using the CD on classical hardware. Specifically, we compare a single-core CPU, a 128-core processor, and a quantum chip with a single copy of the model. RBM on classical computers[48–51] and QRBM on an AQA have already been trained on cybersecurity data[52], providing a proof of concept that a QA-based RBM can be trained on a 64-bit binary dataset.

Data from the cybersecurity world have previously been used for RBM training. Specifically, Aldwairi et al.[50] conducted a study where they trained an RBM using CD and PCD on the ISCX 2012 dataset. The results showed an accuracy rate of over 88%. Dixit et al.[53] trained an RBM on the D-Wave 2000Q employing it as both a classifier and a generative model, showing a proof of concept that a QA-based RBM can be trained on a 64-bit binary dataset. Hybrid approaches were also employed by Li et al.[54], who trained 10% of the KDD-99 dataset by first reducing its dimensionality using an autoencoder and then employing a deep belief network for classification. Similarly, Salama et al.[55] used a hybrid approach, employing a deep belief network for dimensionality reduction and a support vector machine for classification.

Previously, one of us has already shown the training of a fully connected QRBM by embedding techniques[37], also including reverse annealing[38], and quantum supervised learning in gate-model quantum computers by both variational methods[56], quantum neural networks[21], and quantum adversarial networks[57]. Here we find that sampling from the D-Wave Advantage quantum annealer returns a computational complexity that scales linearly with the number of QRBM units and the number of samples extracted, comparable with the classical algorithm. Still, the computational advantage of the quantum device could result in a reduction by orders of magnitude of the computational times. Such a speed-up is problem dependent since the computational time depends on the number of units, the number of Gibbs steps performed, and the number of quantum samples extracted from the quantum machine. The reduction in the computational times could be prominent today in all applications where datasets are massive, with many features, or alternatively, the models need to be re-trained to adapt to continuous changes, such as within the context of cybersecurity applications and intrusion detection systems.

We observed, by comparing the RBMs and QRBMs, that for the real-world cybersecurity datasets like NSL-KDD and CSE-CIC-IDS2018, sampling states for the negative phase are handled up to 64 times faster by the AQA during the training than the single-core times and 41 times faster than the 128 threads CPU. Therefore, we detected a quantum speed-up in existing hardware on real-world data, although it highly depends on the problem's characteristics.

Results

RBMs form a class of neural networks composed of units that can assume binary values, forming a bipartite system coupled by real weighted connections. The visible units represent the model input/output, while the hidden units raise the RBM's ability to mimic the dataset's structure. RBMs can be trained by both a supervised[58] or unsupervised learning algorithm[36], depending on the specific task. The training procedure consists of carefully modifying the model weights to learn how to generate and reconstruct the essential information encoded in the dataset. Several training procedures exist, including the use of quantum annealers, as shown in Fig. 1. The advantage of employing the D-Wave adiabatic quantum machine to exploit RBMs could emerge as an increase of performance metrics or as a reduction in the computational times, such as the training time and the query time. Reducing the training time is essential in all the applications where the model needs to be re-trained periodically to adapt to constantly changing environments[59–63], while reducing the query time is crucial for all applications that benefit from the faster response possible, such as in some anomaly detection tasks[64].

Evaluation of the computational times. To quantify the potential advantage of the D-Wave quantum annealer in reducing computational time, we independently compared the times required to estimate the negative phase by a classical RBM and a QRBM implemented on the D-Wave quantum annealer. In such analysis, we are neglecting the performances of the models on a specific task, focusing on the computational times to estimate the negative phase only, which depends on the size of the models, the number of CD steps k, or the number of samples extracted by the QPU only.

The only difference between the quantum and classical approaches consists of the different methods to sample states with the correct Boltzmann distribution from the model. Since the computational time depends on the size of the model, we selected 676 different RBMs by varying the number of visible (N) and hidden (N_h) units. We measured the time required to perform k CD steps to benchmark the classical RBM. Figure 2 displays the classical times it takes to execute k steps by using either one thread or 128 threads of an AMD Ryzen Threadripper 3990X processor. The classical times highly depend on the size of the model, growing linearly by increasing the number of involved nodes and the number of CD steps.

To measure the sampling times from the QPU, we embedded the same RBMs models on the D-Wave Advantage 4.1 machine by running the minorminer[65] algorithm, commonly employed to map virtual qubits[21] to physical qubits on the hardware. Differently from refs. [34,66], we run minorminer three times per machine to reduce the number of qubits involved and the qubit chain lengths, as described in the Method section. As the graph

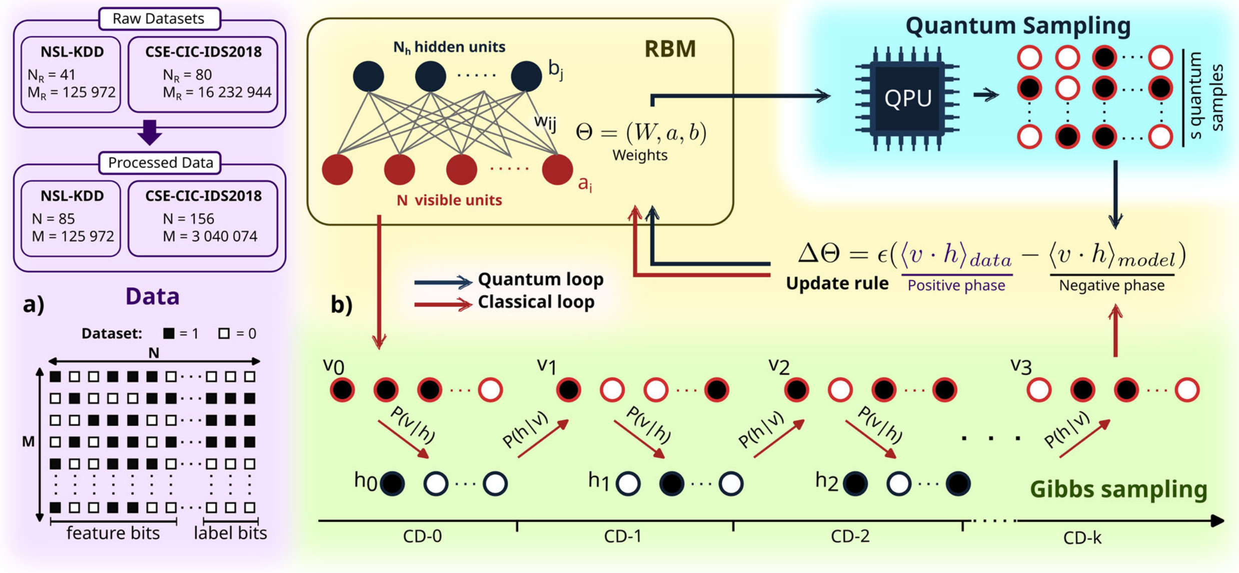

Fig. 1 Overview of the quantum and classical loop to train a restricted Boltzmann machine. a The purple shade shows the structure of the datasets used in this work. The NSL-KDD and CSE-CIC-IDS2018 datasets contain M_R records of N_R miscellaneous data type features. The processed NSL-KDD and CSE-CIC-IDS2018 dataset contain $M = 125972$ and $M = 3040074$ records, respectively, and N binary value column features. The last three bits of each record are reserved for encoding the label. **b** The blue arrows show the quantum training loop, while the red arrows show the classical training loop. In the former, the restricted Boltzmann machine (RBM) weights Θ are encoded in the quantum processor unit (QPU) and next, the s quantum samples are sampled from the QPU to estimate the negative phase statistic. In the latter, the RBM weights are used to perform k Gibbs sampling steps. The samples are subsequently used to estimate the negative phase. The positive phase statistic is calculated by loading a batch of elements sampled from the dataset in the visible units (red dots) and updating the N_h hidden units (blue dots) accordingly. Once the positive and negative phases have been evaluated, the RBM weights are updated, and the training loop continues until a maximum number of epochs is reached.

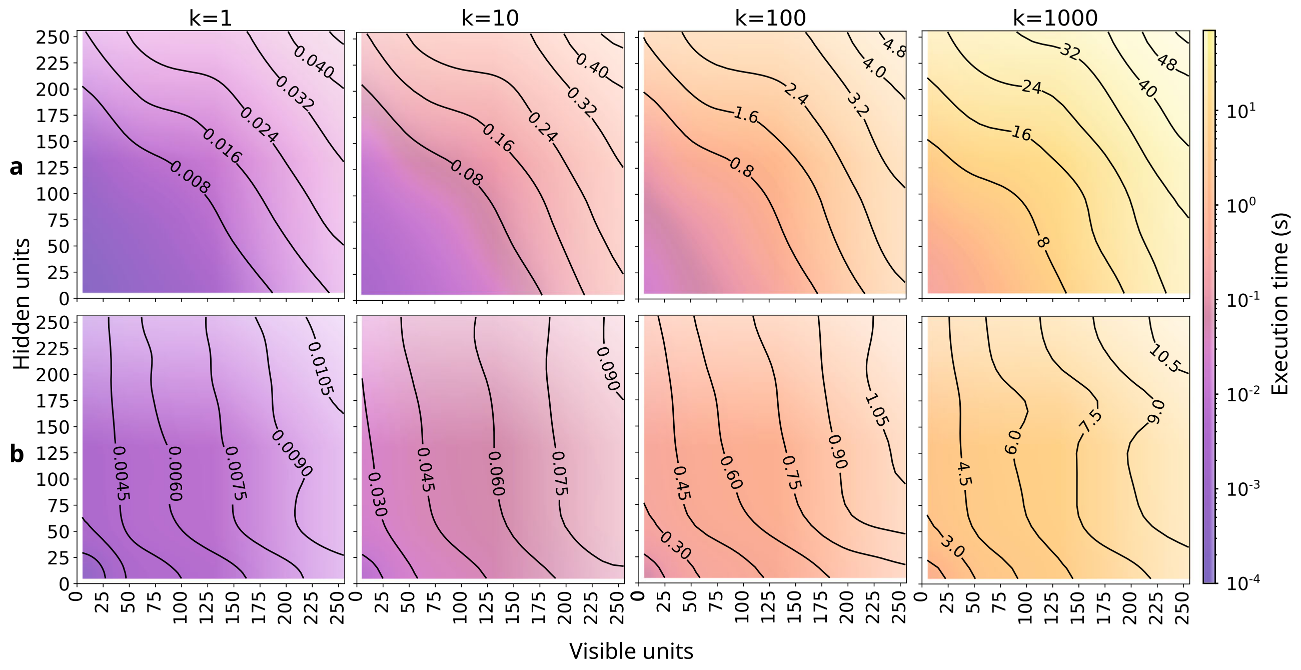

Fig. 2 Time to compute *k* contrastive divergence steps as a function of the restricted Boltzmann machine size. Each contour plot has been obtained by measuring 676 different restricted Boltzmann machines (RBMs) arranged evenly on a square lattice as a function of the number of visible and hidden units. Each RBM has been evaluated 40,000/*k* times to reduce the variance. **a** Single-core times. The time to perform increases linearly with the number of contrastive divergence steps performed. **b** Multi-core times on a 64c/128t machine. The high number of threads available decreases the dependency between the time and hidden units.

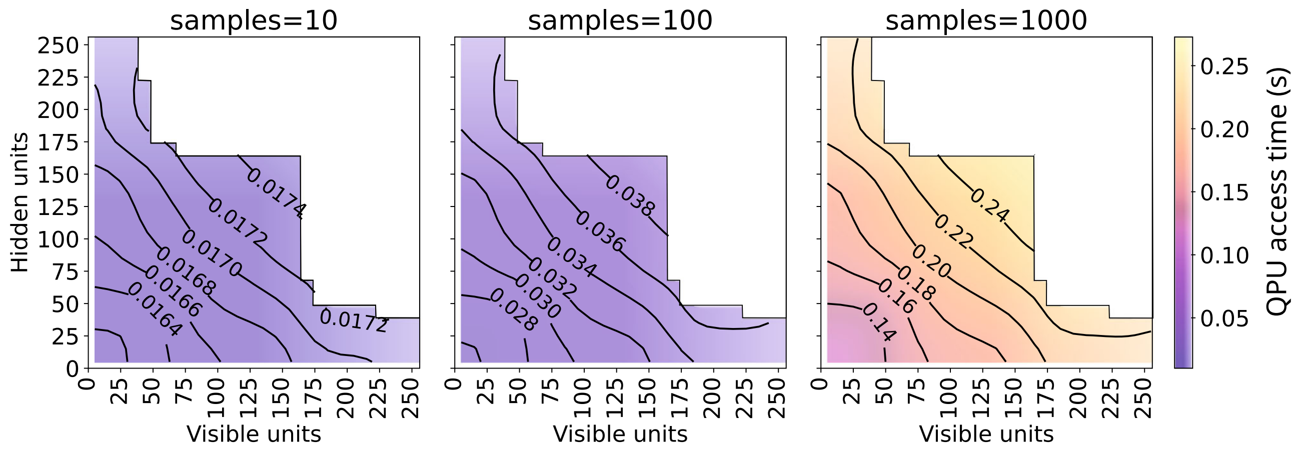

Fig. 3 D-Wave quantum processor units access time for different numbers of samples extracted as a function of the number of the restricted Boltzmann machine (RBM) size. The quantum processor unit (QPU) access time consists of a one-time initialization step to program the QPU and multiple sampling times for the actual execution of the QPU. The QPU access time depends on the number of qubits involved and the number of samples required. The contour plots highlight the region of visible and hidden units of currently embeddable RBMs on the D-Wave Advantage architecture.

representing the hardware has limited qubits and connections, we therefore could embed just a fraction of the RBMs used to benchmark the classical approach. For each QRBM, we sampled 10, 100, and 1000 samples from the quantum processor, respectively. For a fair comparison, since we access the QPU by cloud, we discarded the network latency time by recording the QPU access time instead, corresponding to the execution time during which the QPU is unavailable to any other quantum machine job. Figure 3 shows the time required to sample states from the D-Wave Advantage 4.1 QPU as the model size increases. It shows the QPU access time measured to sample 10, 100, and 1000 states from the QPU. The QPU access time has a mild dependency on the number of visible and hidden units, while it strongly depends on the number of samples, so its value corresponds to $1.6 \cdot 10^{-2}$ s, $3 \cdot 10^{-2}$ s and $2 \cdot 10^{-1}$ s, respectively. Therefore, the quantum hardware may reduce the computational time only in tasks that require both a high number of CD steps and a limited number of quantum samples to be extracted. It is worth noticing that if one wants to compare the speed-up of the classical and quantum models on a specific task, we must choose and fix the above quantities so that their performance metrics are comparable, as empirically done in the following subsection.

Figure 4 shows a linear relation between the number of QRBM units N and the QPU access time. The three data points, corresponding to sample 10, 100, and 1000 states from the QPU, were fitted by linear functions $y_s = m_s x + q_s$. We found angular coefficients $m_{10} = 5.15 \cdot 10^{-6} \pm 1.3 \cdot 10^{-7}$, $m_{100} = 5.14 \cdot 10^{-5} \pm 1.3 \cdot 10^{-6}$, and $m_{1000} = 5.14 \cdot 10^{-4} \pm 1.3 \cdot 10^{-5}$. Therefore, the computational complexity to

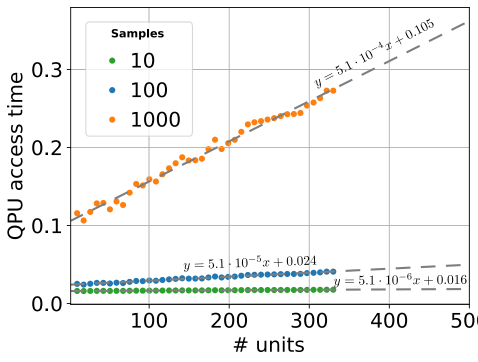

Fig. 4 D-Wave quantum processor units access as a function of the quantum-restricted Boltzmann machine (QRBM) size. The figure shows in green, blue, and orange color the quantum processor unit (QPU) access time measured to sample 10, 100, and 1000 samples, respectively, from the D-Wave Advantage 4.1 quantum annealer as a function of the number of QRBM units (hidden + visible units). The data points show a linear trend between times and QRBM units, as shown by the straight dashed lines. All the data points were fitted by a linear function (dashed gray line) with $R^2 = 0.979$ and root mean square error (RMSE) of $7.1 \cdot 10^{-5}$, $7.1 \cdot 10^{-4}$, and $7.1 \cdot 10^{-3}$, respectively.

sample states from the quantum device scales as $O(N \cdot s)$, where s is the number of samples extracted, which is comparable with the $O(N \cdot k)$ trend of the classical algorithm. Although we are currently limited by the size of the model that can be embedded in the QPU, we can forecast the QPU access time to be 0.514 ± 0.013 µs per unit per sample.

Speed-up on real-world cybersecurity datasets. Let us now search for the signature of a speed-up on real-world datasets. To detect a quantum potential advantage on today's hardware for real-world tasks, we trained both an RBM and a QRBM as NIDSs. The goal is to detect an anomaly, i.e., an attack, from a benign instance. Like other classification and clustering problems, NIDS algorithms process training datasets. In the following, we consider the NSL-KDD[67] and the CSE-CIC-IDS2018[68] datasets. The former represents one of the most widespread datasets in the NIDS domain, while the latter is a modern cyber-defense dataset.

Since the QRBM units require binary input values, we binarized the records of the two datasets to 85 binary features for the NSL-KDD dataset and 156 for the CSE-CIC-IDS2018 dataset (see the Methods). The former dataset comes with training and testing datasets with a ratio of roughly 80%:20%. The latter has been split into training and testing datasets with the same ratio. The two training datasets have been balanced by under-sampling the most common class.

Despite being a generative model, RBMs can be used effectively as classifiers encoding the label as part of the input state during the training and reconstructing the label during the exploitation phase. Here we reserved the last three bits for encoding the label. A majority vote rule was performed to obtain a final result from the model during reconstruction. For instance, if the last three bits are 101, it indicates a benign instance, while if they are 001, it reveals an attack.

To measure the classical RBM's performance in the two datasets, we investigate the optimal number of hidden units and the number of Gibbs sampling during the training and once the model has been trained. First, let us compare the performance of classical RBMs by setting the number of CD steps to 1000 and increasing the number of hidden units until no further improvement in performance is detected. Therefore, in all the following experiments, the number of hidden units for the RBM and the QRBM are set to 30 for the NSL-KDD dataset and 90 for the CSE-CIC-IDS2018 dataset, respectively. Then the training performance of the RBMs is compared by varying the number of CD steps k during the training. The number of Gibbs sampling steps does not affect the performance of the model trained on the NSL-KDD dataset, while it has a mild effect on the CSE-CIC-IDS2018 dataset as shown in Fig. 5. Comparing the performance of the model by varying the number k during the exploitation phase, then in both datasets the performance of the RBMs depends on the number of CD steps, as shown in Table 1. In particular, it emerges a 13% increase in accuracy by employing 1000 CD steps compared to a single CD step in the CSE-CIC-IDS2018 dataset.

Finally, the performance of the QRBM in the two datasets is measured by setting the same hyperparameters in common with the classical RBMs. More details on the hyperparameters employed during the training are given in Supplementary Table 1. The number of quantum samples required to train the QRBM is reasonably limited. In particular, it is possible to train the QRBM by extracting ten samples only from the QPU. In addition, the QRBM performance is comparable to the classical RBM, although less effective on the CSE-CIC-IDS2018 dataset, as shown in Table 2 and in Supplementary Fig. 1. Since we have not further optimized the hyperparameters in this work, the difference in the performance could be due to sub-optimal values and could be leveled by carefully choosing the optimal parameters. It is worth noting that the quantum model requires less training epochs to achieve a target accuracy (η) for the NSL-KDD dataset, while for the CSE-CIC-2018 dataset it is the opposite, as shown in Supplementary Table 3.

By comparing the classical and quantum computational time, we detect a speed-up in the computational time by employing the QRBM as NIDS trained on the two real-world cybersecurity datasets. In fact, to maximize the performance of the classical machine during the exploitation phase, it is required to perform more than 10 and 100 CD steps for the NSL-KDD and CIC-IDS2018 datasets, respectively. At the same time, it is necessary to sample 10 samples only from the QPU. Table 3 summarizes the speed-up measured on the two datasets.

Discussion

We now turn our attention to the computational time, the computational complexity, and the performance achieved by the model at the end of the training process on today's classical and quantum hardware.

The time required to train and query the classical RBM depends on both the size of the model and the number of CD steps, respectively. The latter affects the performance achieved by RBMs and it should be chosen carefully depending on the particular problem to balance the times and model accuracy. Increasing the number of thermalization cycles comes at the cost of higher computational time, which can increase by orders of magnitude. Although it is possible to use a single CD step during the training phase, significantly reducing the time while still achieving excellent performance[49,50,52], such choice is not possible in the exploitation phase, where one wants to maximize the performance of the model, as shown in Table 1.

Although in principle quantum annealing offers a different way of training machines by avoiding long thermalization cycles with the complexity of a single quantum operation in theory, in practice, sampling states from D-Wave is an operation that depends on the number of involved qubits and the number of samples extracted. In particular, Fig. 4 shows that the computational complexity scales linearly with the number of units and the number of samples extracted, comparable with the classical

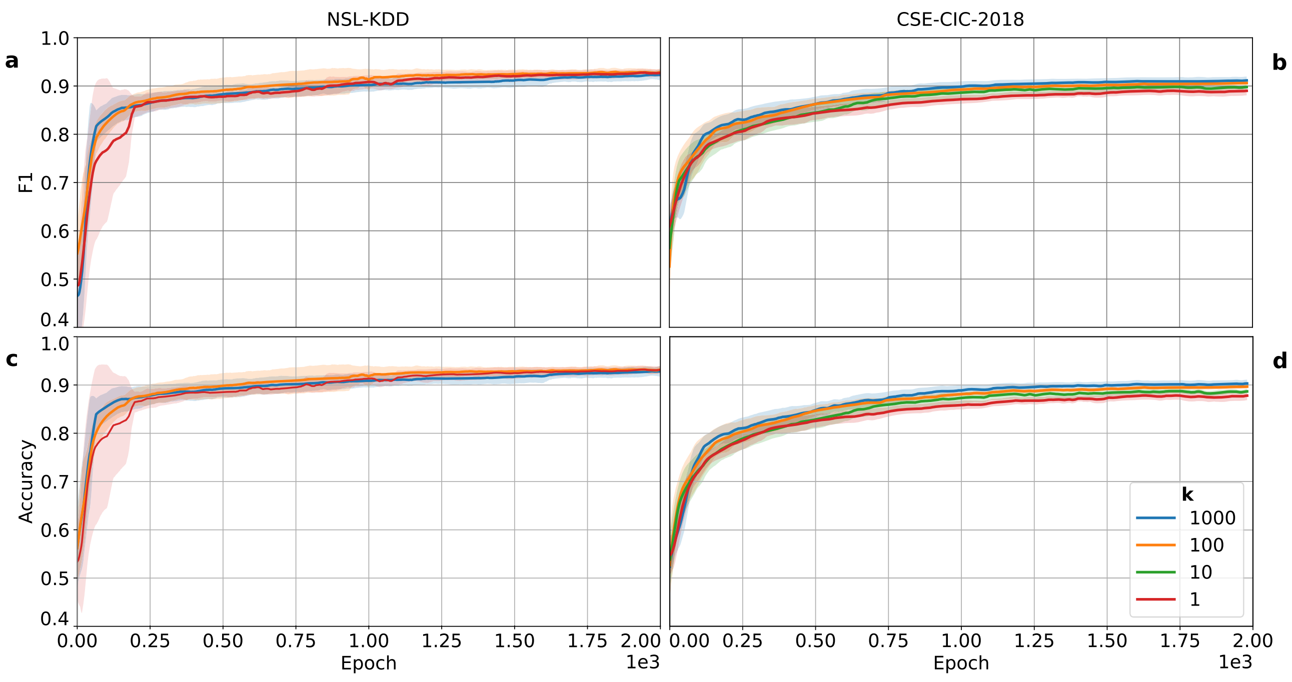

Fig. 5 Restricted Boltzmann machine (RBM) performance during classical training as a function of the number of Gibbs steps (*k*). The figure shows the *F*1 score (**a**, **b**) and the accuracy (**c**, **d**) achieved by training an RBM using different Gibbs step values for the NSL-KDD and the CSE-CIC-IDS2018 datasets during 2000 training epochs. *F*1 score and accuracy were chosen as performance metrics since computing the log-likelihood evaluation is impractical for RBMs of moderate size. In addition, these metrics allow for comparing RBM and QRBM models with existing literature on anomaly detection. The performance was measured using the testing dataset and averaged over 10 identical runs showing the 95% confidence interval as colored shadows. The number of Gibbs sampling steps does not affect the performance of the model trained on the NSL-KDD dataset, while it has a minor effect on the model performance trained on the CSE-CIC-IDS2018 dataset.

Table 1 Performance of restricted Boltzmann machine (RBM) as a function of the number of Gibbs steps (*k*).

Dataset	*k*	Accuracy	*F*1	TP	FP	FN	TN
NSL-KDD	1	0.906 ± 0.005	0.901 ± 0.005	0.909 ± 0.009	0.091 ± 0.009	0.096 ± 0.005	0.903 ± 0.005
	10	0.935 ± 0.002	0.932 ± 0.002	0.936 ± 0.003	0.064 ± 0.003	0.065 ± 0.003	0.935 ± 0.003
	100	0.937 ± 0.002	0.934 ± 0.002	0.939 ± 0.004	0.061 ± 0.004	0.064 ± 0.001	0.935 ± 0.001
	1000	0.938 ± 0.002	0.935 ± 0.002	0.979 ± 0.002	0.020 ± 0.003	0.064 ± 0.002	0.936 ± 0.002
CSE-CIC-IDS2018	1	0.800 ± 0.004	0.805 ± 0.005	0.833 ± 0.003	0.166 ± 0.004	0.234 ± 0.006	0.766 ± 0.006
	10	0.897 ± 0.001	0.903 ± 0.001	0.907 ± 0.002	0.093 ± 0.002	0.113 ± 0.002	0.887 ± 0.002
	100	0.915 ± 0.001	0.921 ± 0.001	0.922 ± 0.003	0.078 ± 0.003	0.092 ± 0.001	0.907 ± 0.001
	1000	0.924 ± 0.001	0.929 ± 0.001	0.932 ± 0.001	0.068 ± 0.002	0.084 ± 0.001	0.916 ± 0.001

The table reports the accuracy, *F*1 score, and the normalized number of true positives (TP), false positives (FP), false negatives (FN), and true negatives (TN) measured for the RBM at the end of the training procedure (2000 epochs) for the two datasets.

Table 2 Performance of quantum-restricted Boltzmann machine.

Dataset	Accuracy	*F*1	TP	FP	FN	TN
NSL-KDD	0.937 ± 0.001	0.935 ± 0.001	0.951 ± 0.002	0.050 ± 0.002	0.073 ± 0.002	0.927 ± 0.002
CSE-CIC-IDS2018	0.8539 ± 0.0004	0.8612 ± 0.0003	0.8730 ± 0.0006	0.1270 ± 0.0005	0.1652 ± 0.0004	0.8348 ± 0.0004

The table reports the accuracy, *F*1 score, and the normalized number of true positives (TP), false positives (FP), false negatives (FN), and true negatives (TN) measured for the quantum-restricted Boltzmann machine at the end of the training procedure (2000 epochs) for the two datasets.

computer, where the dependence on the number of Gibbs steps and the model size is linear.

The quantum speed-up offered by quantum annealers is problem dependent. It could emerge only for a sufficiently large RBM or for tasks that require long thermalization cycles and, therefore, a high number of Gibbs steps. We observed an advantage in terms of the computational time by employing a QRBM as an NIDS trained on two real-world cybersecurity datasets up to 64*x* compared to a single-core trained RBM, as summarized in Table 3.

It is worth highlighting that modern CPUs can process batches of data thanks to parallel computing, while for large RBMs, the

Table 3 Quantum speed-up for the NSL-KDD and CIC-IDS2018 datasets.

Dataset	Processor	*k* - *s*	Computational time (s)	Speed-up
CSE-CIC-IDS2018	Threadripper 3990X – single core	100	1.1	64x
	Threadripper 3990X – 128 cores	100	0.71	41x
	D-Wave Advantage 4.1	10	0.017	1x
NSL-KDD	Threadripper 3990X – single core	10	0.070	4x
	Threadripper 3990X – 128 cores	10	0.035	2x
	D-Wave Advantage 4.1	10	0.016	1x

The table reports the computational time and speed-up to evaluate the negative phase offered by the quantum model for the NSL-KDD and CIC-IDS2018 datasets. The column "*k* - *s*" reports the number of contrastive divergence steps performed by the classical restricted Boltzmann machine or the number of quantum samples extracted from the quantum processor unit depending on the processor employed.

QPU is limited to processing a single data point at each cycle, hugely increasing the computational time of the quantum model during the exploitation phase. Such limitation could be mitigated by embedding multiple QRBM replicas on the QPU at the cost of increasing the number of physical qubits involved.

The advantage carried by a quantum annealer may increase in the following years through quantum hardware improvements. Increasing the number of physical qubits will make it possible to embed larger models and multiple replicas. However, improving the number of connections per qubit could provide a more significant advantage by reducing the number of physical qubits involved and, consequently, the computational complexity of quantum annealers.

Methods

Restricted Boltzmann machine. RBMs are probabilistic neural network models that consist of a layer of visible binary units $\mathbf{v} = (v_1, v_2, \cdots, v_N)$ connected to a layer of hidden binary units $\mathbf{h} = (h_1, h_2, \cdots, h_M)$. Although each unit is connected to all the units in the opposite layer, no intraconnection is allowed, forming a bipartite system. They belong to the class of generative models and they can learn the underlying probability distributions of the dataset inputs once properly trained. The RBM is an energy-based model where every "state", i.e., specific configuration of $\mathbf{v}$ and $\mathbf{h}$, is associated with an energy:

$$E(\mathbf{v},\mathbf{h}) = -\sum_i a_i v_i - \sum_j b_j h_j - \sum_{i,j} v_i W_{ij} h_j \tag{1}$$

where $\mathbf{a}$, $\mathbf{b}$ are biases and $\mathbf{W}$ are the weights that represent the connection strength between units. The joint probability of a state is given by the Boltzmann distribution is:

$$P(\mathbf{v},\mathbf{h}) = \frac{\exp(-E(\mathbf{v},\mathbf{h}))}{\sum_{\mathbf{v},\mathbf{h}} \exp(-E(\mathbf{v},\mathbf{h}))} \tag{2}$$

while the probability of finding an individual visible unit $v_i = 1$ given the hidden units $\mathbf{h}$ is:

$$P(v_i = 1|\mathbf{h}) = \frac{\exp(a_i + W_{ij} b_j)}{1 + \exp(a_i + W_{ij} b_j)} = \sigma(a_i + W_{ij} b_j) \tag{3}$$

where σ is the logistic function. In the same fashion, the probability of finding an individual visible unit $h_j = 1$ given the visible units $\mathbf{v}$ is:

$$P(h_j = 1|\mathbf{v}) = \sigma(b_j + W_{ij} a_i) \tag{4}$$

The goal of the training is to learn the best parameters $\theta = (\mathbf{a}, \mathbf{b}, \mathbf{W})$ that maximize the data log-likelihood. One strategy to achieve that is to perform gradient ascent steps of the log-likelihood function $ll(\theta)$:

$$\theta^{new} = \theta^{old} + \epsilon \frac{\partial ll(\theta)}{\partial \theta} \tag{5}$$

where ϵ corresponds to the learning rate.

It can be proven that

$$\frac{\partial ll(\theta)}{\partial \theta} = \sum_{t=1}^{N} \left\langle \frac{\partial(-E(\mathbf{v}_t,\mathbf{h}_t))}{\partial \theta} \right\rangle_{P(\mathbf{h}_t|\mathbf{v}_t)} - N \cdot \left\langle \frac{\partial(-E(\mathbf{v},\mathbf{h}))}{\partial \theta} \right\rangle_{P(\mathbf{v},\mathbf{h})} \tag{6}$$

where $\langle \cdot \rangle_{P(\cdot)}$ denotes the expectation values with respect a distribution $P(\cdot)$, $\mathbf{h}_t$ is sampled from $\mathbf{v}_t$ by using Eq. (4), and the sum is over a dataset with N records. The first term in Eq. (6) is called "positive phase" and can be estimated by using the data. However, the second term, called "negative phase", is model dependent and requires a sum over all the states, which is intractable except for very small RBMs. Although an exact computation is unfeasible, there exist several methods that can be employed to approximate the expectation, such as CD, PCD, and Fast PCD.

Contrastive divergence. The CD procedure is a technique to approximate the negative phase by running a Monte Carlo Markov chain until a near-to-equilibrium distribution is reached. Ideally, the number of iterations should be high enough to get almost unbiased samples of the distribution modeled by the RBM (Gibbs sampling). Since the computational time required to get unbiased samples is considerable, the procedure is usually stopped after k iterations (CD-k).

The basic training idea of CD-k is to start the learning procedure by randomly setting the parameters θ. Then k sequences of Gibbs sampling steps are performed by starting from a vector $\mathbf{v}(\mathbf{0})$ sampled from the dataset to reach state $\mathbf{v}(k)$ and $\mathbf{h}(k)$. More precisely, during each step $l < k$ of the Gibbs sampling, we sequentially sample $\mathbf{h}(l)$ starting from $\mathbf{v}(l)$ using Eq. (4) followed by sampling $\mathbf{v}(l+1)$ from $\mathbf{h}(l)$ using Eq. (3). The state $\mathbf{v}(k)$, $\mathbf{h}(k)$ are then used to estimate the negative phase and, therefore, the log-likelihood derivatives. Finally, θ is updated by a single gradient ascent step using Eq. (5).

It is worth noticing that the CD procedure does not approximate the correct model distribution $P(\mathbf{v},\mathbf{h})$, returning a biased estimate of the desired update direction[69]. Nevertheless, it has proven to be successful in training challenging applications.

Quantum training. Quantum annealers can be used to estimate the negative phase by extracting samples from the distribution associated with the RBM. The basic idea is to embed the RBM weights and biases on the QPU and then perform an annealing cycle. While in principle the machine is expected to obtain samples representing the system's ground state, in practice Dumoulin et al.[70] observed that the AQA generates configurations with higher energy levels, as if they were extracted from a Boltzmann distribution. Specifically, a quantum annealer at very long annealing times returns a final population that is close to such a distribution, up to a point where the dynamics freeze and the system deviates from the equilibrium[71]. If the dynamics slow down and freeze out within a short period of time, then an AQA with a linear annealing schedule will provide samples from the Boltzmann distribution[22]. However, in such a region, the actual probability distribution sampled by the AQA[52,70,72] deviates slightly from Eq. (2). In fact, at each annealing cycle, samples are extracted with probability:

$$P(\mathbf{v},\mathbf{h}) = \frac{\exp(-E(\mathbf{v},\mathbf{h})/T_{eff})}{\sum_{\mathbf{v},\mathbf{h}}\exp(-E(\mathbf{v},\mathbf{h})/T_{eff})} \qquad (7)$$

where T_{eff} is an effective temperature determined by thermal noise inside the chip[73,74]. To compensate for this problem, the energy is rescaled by a factor α so that we sample with an effective temperature $T_{eff} = 1$. The optimal hyperparameter α should be set periodically since it changes over time, and its time evolution is challenging to forecast[38,52]. The mismatch between α and T_{eff} might lead to sub-optimal RBM training. Supplementary Table 1 reports the value for the annealing time and α employed during the QRBM training for the two cybersecurity datasets.

Embedding. To sample states from a QRBM directly on a D-Wave system, or in general, to solve a QUBO problem directly on a D-Wave system, it is required to map the model onto the QPU topology. Such a procedure, called embedding, consists of identifying groups of physical qubits in the QPU (chains) so that they form the topology of the problem under investigation by behaving as individual units. The connectivity of each group can be enhanced by creating strong ferromagnetic couplings between the qubits (Jchain), which forces coupled qubits to stay in the same state. Achieving the optimal chain strength involves finding a balance where setting it too low increases the likelihood of encountering broken chains in the output, while setting it too high may result in weights that are not sufficiently distinguished by the analog control system[73]. Finding the best embedding is generally an NP-hard problem[65]. Although it is possible to get the optimal embedding of a QRBM manually by exploiting the specific topology of the QPU[37,52,53], we exploited the minorminer algorithm[65] provided by D-Wave to find the embedding for the QRBMs investigated in this work. Since the minorminer algorithm is heuristic and finds an embedding with some probability, we run the algorithm three times per QRBM to reduce the number of physical qubits involved in the mapping and the length of the chains. Supplementary Table 2 provides information on the lengths of the chains and the corresponding Jchain values used in the embedding for representing the QRBM utilized in the NSL-KDD and CSE-CIC-IDS2018 datasets.

The datasets. We selected two datasets for training RBMs on quantum and on classical computers. The first one, the NSL-KDD dataset, comes from the effort to partially solve the problems of the KDD-99 cup dataset[67], such as duplicate data and a large number of records. Its wide diffusion makes it suitable for training ML models in the field of NIDS since it is essential to compare and relate the different performances of the proposed models. However, it suffers from some problems, as pointed out by McHugh[75]. For example, it is not representative of real networks because it lacks public datasets for network-based IDS and is not representative of low-footprint attacks. For this reason, we selected a second dataset, the CSE-CIC-IDS2018 dataset representing a modern, realistic dataset. The CSE-CIC-IDS2018 is one of the most recent intrusion detection datasets publicly available, with an extensive range of attack classes.

Feature extraction. The NSL-KDD dataset contains 41 variables (N_R), excluding labels. The training dataset has 125,972 observations (M), while the testing dataset has 22,543 records. The training dataset includes 21 types of attacks, while the testing dataset contains 37 kinds of attacks. These attacks are classified into Denial Of Service, Probe, User to Root, and Root to Local. In this work, we perform a binary classification, i.e., we do not consider the different classes of attacks.

The CSE-CIC-IDS2018 dataset cannot be employed as it is, since it contains millions of records, with more than 80 network traffic features (N_R), duplicate data, and missing values. Moreover, it is impossible to encode all the dataset features in the current D-Wave Advantage QPU due to their number and the limited number of physical qubits available. Therefore, after removing all the duplicated records and missing data, we selected a subset of features by applying hierarchical clustering on the Spearman rank-order correlation and keeping a single feature from each cluster[76]. As a result, we discarded the following features: “Fwd Seg Size Avg”, “TotLen Fwd Pkts”, “Bwd IAT Tot”, “Bwd Pkt Len Max”, “Subflow Fwd Byts”,"Pkt Size Avg”, “Bwd Seg Size Avg”, “Subflow Fwd Pkts”, “Bwd IAT Max”, “Subflow Bwd Pkts”, “Flow Pkts_over_s”, “Flow IAT Mean”, “Idle Max”, “Active Min”, “Active Max”, and “Pkt Len Min”.

QRBM models require binary value numbers for the input units. The two datasets were processed to fulfill this requirement. We used a one-hot encoding for nominal features with a limited number of distinct values, while we used a binary encoding for those with many different values. All the real-valued records have been digitized, i.e., we computed the histogram of the record, and we substituted the value with the indices of the bins to which the input belongs. We associated normal data with $+1$, and we used 0 for attacked records. For the sake of redundancy, the labels have been encoded three times, i.e., attacks are labeled as $(0, 0, 0)$ and normal data as $(1, 1, 1)$. At the end of the processing the NSL-KDD dataset in compressed by 85 bits (N) while the CSE-CIC-IDS2018 dataset 156 (N). Finally, we balance the datasets by under-sampling the most common class.

Performance metrics. During this evaluation process, we measured six metrics to test the performance of the models: accuracy, $F1$ score, true positive (TP), false negative (FN), false positive (FP), and true negative (TN).

True positive: the number of TP indicates the number of attacks correctly classified by the model.

True negative: the number of TN indicates the number of normal events correctly classified by the model.

False negative: the number of FN indicates the number of attacks classified as normal events by the model.

False positive: the number of FP indicates the number of normal events classified as attacks by the model.

Accuracy: represents the proportion of total predictions that have been classified correctly.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FN + FP}$$

$F1$ score: represents a harmonic mean of precision and recall, where 1 represents the best score and 0 represents the worst score.

$$F1 = \frac{2TP}{2TP + FP + FN}$$

Data availability

The data that support the findings of this study are available from the corresponding author upon reasonable request.

Code availability

The code and the algorithm used in this study are available from the corresponding author upon reasonable request.

Received: 1 March 2023; Accepted: 12 September 2023;
Published online: 23 September 2023

References

1. Brown, T. et al. Language models are few-shot learners. *Adv. Neural Inf. Process. Syst.* **33**, 1877–1901 (2020).
2. Jumper, J. et al. Highly accurate protein structure prediction with alphafold. *Nature* **596**, 583–589 (2021).
3. Crispin, A. & Syrichas, A. Quantum annealing algorithm for vehicle scheduling. In *2013 IEEE International Conference on Systems, Man, and Cybernetics*, 3523–3528 (IEEE, 2013).
4. Wittek, P. *Quantum Machine Learning: What Quantum Computing Means to Data Mining* (Academic Press, 2014).
5. Wiebe, N., Kapoor, A. & Svore, K. M. Quantum deep learning. *Quantum Information & Computation* **16**, 7–8, 541–587 (2016).
6. Biamonte, J. et al. Quantum machine learning. *Nature* **549**, 195–202 (2017).
7. Prati, E. Quantum neuromorphic hardware for quantum artificial intelligence. In *Journal of Physics: Conference Series*, vol. 880, 012018 (IOP Publishing, 2017).
8. Sajjan, M. et al. Quantum machine learning for chemistry and physics. *Chem. Soc. Rev.* **51**, 6475–6573 (2022).
9. Atz, K., Isert, C., Böcker, M. N., Jiménez-Luna, J. & Schneider, G. δ-quantum machine-learning for medicinal chemistry. *Phys. Chem. Chem. Phys.* **24**, 10775–10783 (2022).
10. Li, J., Topaloglu, R. O. & Ghosh, S. Quantum generative models for small molecule drug discovery. *IEEE Trans. Quantum Eng.* **2**, 1–8 (2021).
11. Li, J. et al. Drug discovery approaches using quantum machine learning. In *2021 58th ACM/IEEE Design Automation Conference (DAC)*, 1356–1359 (IEEE, 2021).
12. Outeiral, C. et al. The prospects of quantum computing in computational molecular biology. *Wiley Interdiscip. Rev. Comput. Mol. Sci.* **11**, e1481 (2021).
13. Cao, Y., Romero, J. & Aspuru-Guzik, A. Potential of quantum computing for drug discovery. *IBM J. Res. Dev.* **62**, 6–1 (2018).
14. Herman, D. et al. A survey of quantum computing for finance. Preprint at *arXiv:2201.02773* (2022).
15. Venturelli, D. & Kondratyev, A. Reverse quantum annealing approach to portfolio optimization problems. *Quantum Mach. Intell.* **1**, 17–30 (2019).
16. Huang, B., Symonds, N. O. & von Lilienfeld, O. A. Quantum machine learning in chemistry and materials. In *Handbook of Materials Modeling: Methods: Theory and Modeling*, 1883–1909 (Springer, 2020).
17. Rebentrost, P., Mohseni, M. & Lloyd, S. Quantum support vector machine for big data classification. *Phys. Rev. Lett.* **113**, 130503 (2014).
18. Shor, P. W. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Rev.* **41**, 303–332 (1999).
19. Harrow, A. W., Hassidim, A. & Lloyd, S. Quantum algorithm for linear systems of equations. *Phys. Rev. Lett.* **103**, 150502 (2009).
20. Farhi, E., Goldstone, J. & Gutmann, S. A quantum approximate optimization algorithm. Preprint at *arXiv:1411.4028* (2014).
21. Maronese, M., Destri, C. & Prati, E. Quantum activation functions for quantum neural networks. *Quantum Inf. Process.* **21**, 1–24 (2022).
22. Amin, M. H., Andriyash, E., Rolfe, J., Kulchytskyy, B. & Melko, R. Quantum Boltzmann machine. *Phys. Rev. X* **8**, 021050 (2018).
23. Molteni, R., Destri, C. & Prati, E. Optimization of the memory reset rate of a quantum echo-state network for time sequential tasks. *Phys. Lett. A* **465**, 128713 (2023).
24. Adachi, S. H. & Henderson, M. P. Application of quantum annealing to training of deep neural networks. Preprint at *arXiv:1510.06356* (2015).
25. Neven, H. et al. NIPS 2009 demonstration: binary classification using hardware implementation of quantum annealing. *Quantum* **4**, 1–17 (2009).
26. Agliardi, G., Grossi, M., Pellen, M. & Prati, E. Quantum integration of elementary particle processes. *Phys Lett B* **832**, 137228 (2022).
27. Bauer, B., Bravyi, S., Motta, M. & Chan, G. K.-L. Quantum algorithms for quantum chemistry and quantum materials science. *Chem. Rev.* **120**, 12685–12717 (2020).
28. Babbush, R., Love, P. J. & Aspuru-Guzik, A. Adiabatic quantum simulation of quantum chemistry. *Sci. Rep.* **4**, 1–11 (2014).
29. Hernandez, M. & Aramon, M. Enhancing quantum annealing performance for the molecular similarity problem. *Quantum Inf. Process.* **16**, 1–27 (2017).
30. Ikeda, K., Nakamura, Y. & Humble, T. S. Application of quantum annealing to nurse scheduling problem. *Sci. Rep.* **9**, 1–10 (2019).
31. Venturelli, D., Marchand, D. J. & Rojo, G. Quantum annealing implementation of job-shop scheduling. Preprint at *arXiv:1506.08479* (2015).
32. Li, R. Y., Di Felice, R., Rohs, R. & Lidar, D. A. Quantum annealing versus classical machine learning applied to a simplified computational biology problem. *NPJ Quantum Inf.* **4**, 1–10 (2018).
33. Lloyd, S., Mohseni, M. & Rebentrost, P. Quantum algorithms for supervised and unsupervised machine learning. Preprint at *arXiv:1307.0411* (2013).
34. Benedetti, M., Realpe-Gómez, J., Biswas, R. & Perdomo-Ortiz, A. Quantum-assisted learning of hardware-embedded probabilistic graphical models. *Phys. Rev. X* **7**, 041052 (2017).
35. Yalovetzky, R., Minssen, P., Herman, D. & Pistoia, M. Nisq-hhl: Portfolio optimization for near-term quantum hardware. *Bulletin of the American Physical Society*, **67**, 3, K36.00011 (2022).
36. Hinton, G. E. Training products of experts by minimizing contrastive divergence. *Neural Comput.* **14**, 1771–1800 (2002).
37. Rocutto, L. & Prati, E. A complete restricted Boltzmann machine on an adiabatic quantum computer. *Int. J. Quantum Inf.* **19**, 2141003 (2021).
38. Rocutto, L., Destri, C. & Prati, E. Quantum semantic learning by reverse annealing of an adiabatic quantum computer. *Adv. Quantum Technol.* **4**, 2000133 (2021).
39. Hinton, G. E. & Salakhutdinov, R. R. Reducing the dimensionality of data with neural networks. *science* **313**, 504–507 (2006).
40. Larochelle, H., Mandel, M., Pascanu, R. & Bengio, Y. Learning algorithms for the classification restricted Boltzmann machine. *J. Mach. Learn. Res.* **13**, 643–669 (2012).
41. Zhang, N., Ding, S., Zhang, J. & Xue, Y. An overview on restricted Boltzmann machines. *Neurocomputing* **275**, 1186–1199 (2018).
42. Salakhutdinov, R., Mnih, A. & Hinton, G. Restricted Boltzmann machines for collaborative filtering. In *Proceedings of the 24th International Conference on Machine Learning*, 791–798 (Association for Computing Machinery, 2007).
43. Tieleman, T. Training restricted Boltzmann machines using approximations to the likelihood gradient. In *Proceedings of the 25th International Conference on Machine Learning*, 1064–1071 (Association for Computing Machinery, 2008).
44. Liao, H.-J., Lin, C.-H. R., Lin, Y.-C. & Tung, K.-Y. Intrusion detection system: a comprehensive review. *J. Netw. Comput. Appl.* **36**, 16–24 (2013).
45. Javaid, A., Niyaz, Q., Sun, W. & Alam, M. A deep learning approach for network intrusion detection system. In *Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS)*, 21–26 (Association for Computing Machinery, 2016).
46. Ravipati, R. D. & Abualkibash, M. Intrusion detection system classification using different machine learning algorithms on KDD-99 and NSL-KDD datasets—a review paper. *Int. J. Comput. Sci. Inf. Technol.* **11** 205-212 (2019).
47. Ingre, B., Yadav, A. & Soni, A. K. Decision tree based intrusion detection system for NSL-KDD dataset. In *International Conference on Information and Communication Technology for Intelligent Systems*, 207–218 (Springer, 2017).
48. Rosa, G. H. D., Roder, M., Santos, D. F. & Costa, K. A. Enhancing anomaly detection through restricted Boltzmann machine features projection. *Int. J. Inf. Technol.* **13**, 49–57 (2021).
49. Imamverdiyev, Y. & Abdullayeva, F. Deep learning method for denial of service attack detection based on restricted Boltzmann machine. *Big Data* **6**, 159–169 (2018).
50. Aldwairi, T., Perera, D. & Novotny, M. A. An evaluation of the performance of restricted Boltzmann machines as a model for anomaly network intrusion detection. *Comput. Netw.* **144**, 111–119 (2018).
51. Seo, S., Park, S. & Kim, J. Improvement of network intrusion detection accuracy by using restricted Boltzmann machine. In *2016 8th International Conference on Computational Intelligence and Communication Networks (CICN)*, 413–417 (IEEE, 2016).
52. Dixit, V. et al. Training a quantum annealing based restricted Boltzmann machine on cybersecurity data. *IEEE Trans. Emerg. Top. Comput. Intell.* **6**, 417–428 (2021).
53. Dixit, V., Selvarajan, R., Alam, M. A., Humble, T. S. & Kais, S. Training and classification using a restricted Boltzmann machine on the d-wave 2000q. Preprint at *arXiv:2005.03247* (2020).
54. Li, Y., Ma, R. & Jiao, R. A hybrid malicious code detection method based on deep learning. *Int. J. Secur. Appl.* **9**, 205–216 (2015).
55. Salama, M. A., Eid, H. F., Ramadan, R. A., Darwish, A. & Hassanien, A. E. Hybrid intelligent intrusion detection scheme. In *Soft Computing in Industrial Applications*, 293–303 (Springer, 2011).
56. Lazzarin, M., Galli, D. E. & Prati, E. Multi-class quantum classifiers with tensor network circuits for quantum phase recognition. *Phys. Lett. A* **434**, 128056 (2022).
57. Agliardi, G. & Prati, E. Optimal tuning of quantum generative adversarial networks for multivariate distribution loading. *Quantum Rep.* **4**, 75–105 (2022).
58. Larochelle, H. & Bengio, Y. Classification using discriminative restricted Boltzmann machines. In *Proceedings of the 25th International Conference on Machine Learning*, 536–543 (Association for Computing Machinery, 2008).
59. Gonçalves Jr, P. M., de Carvalho Santos, S. G., Barros, R. S. & Vieira, D. C. A comparative study on concept drift detectors. *Expert Syst. Appl.* **41**, 8144–8156 (2014).
60. Hoi, S. C., Sahoo, D., Lu, J. & Zhao, P. Online learning: a comprehensive survey. *Neurocomputing* **459**, 249–289 (2021).
61. Van de Ven, G. M. & Tolias, A. S. Three scenarios for continual learning. Preprint at *arXiv:1904.07734* (2019).

62. Lee, C. S. & Lee, A. Y. Clinical applications of continual learning machine learning. *Lancet Digit. Health* **2**, e279–e281 (2020).
63. Lesort, T. et al. Continual learning for robotics: definition, framework, learning strategies, opportunities and challenges. *Inf. Fusion* **58**, 52–68 (2020).
64. Pang, G., Shen, C., Cao, L. & Hengel, A. V. D. Deep learning for anomaly detection: a review. *ACM Comput. Surv.* **54**, 1–38 (2021).
65. Cai, J., Macready, W. G. & Roy, A. A practical heuristic for finding graph minors. Preprint at *arXiv:1406.2741* (2014).
66. Henderson, M., Novak, J. & Cook, T. Leveraging adiabatic quantum computation for election forecasting. Preprint at *arXiv:1802.00069* (2018).
67. Tavallaee, M., Bagheri, E., Lu, W. & Ghorbani, A. A. A detailed analysis of the KDD CUP 99 data set. In *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, 1–6 (IEEE, 2009).
68. Sharafaldin, I., Lashkari, A. H. & Ghorbani, A. A. Toward generating a new intrusion detection dataset and intrusion traffic characterization. *ICISSp* **1**, 108–116 (2018).
69. Sutskever, I. & Tieleman, T. On the convergence properties of contrastive divergence. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 789–795 (JMLR Workshop and Conference Proceedings, 2010).
70. Dumoulin, V., Goodfellow, I., Courville, A. & Bengio, Y. On the challenges of physical implementations of rbms. In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 28 (Association for the Advancement of Artificial Intelligence, 2014).
71. Amin, M. H. Searching for quantum speedup in quasistatic quantum annealers. *Phys. Rev. A* **92**, 052323 (2015).
72. Raymond, J., Yarkoni, S. & Andriyash, E. Global warming: temperature estimation in annealers. *Front. ICT* **3**, 23 (2016).
73. D-Wave Systems, Technical description of the D-wave quantum processing unit, Tech. Rep.. D-Wave User Manual 09-1109A-V, D-Wave Systems Inc., Burnaby, BC, Canada (2020)
74. Denil, M. & de Freitas, N. Toward the implementation of a quantum RBM, *Proceedings of the NIPS 2011 Deep Learning and Unsupervised Feature Learning Workshop*, 1–9 (2011).
75. McHugh, J. Testing intrusion detection systems: a critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by Lincoln Laboratory. *ACM Trans. Inf. Syst. Secur.* **3**, 262–294 (2000).
76. Yu, M. et al. Hierarchical clustering in minimum spanning trees. *Chaos* **25**, 023107 (2015).

Acknowledgements

This research has been partially funded by Vista Technology SRL. The authors acknowledge support from the University of Milan through the APC initiative and the grant PSR2021DIP008. E.P. would like to thank the CINECA for providing access to the D-Wave quantum computer through the ISCRA-C project Q-Learn. E.P. and L.M. would like to thank Lorenzo Rocutto for useful discussions.

Author contributions

L.M. wrote all the codes and performed the experiments. E.P. conceived and coordinated this research. The authors contributed to discussing the results and to the writing of the manuscript.

Competing interests

The authors declare no competing interests.

Additional information

Supplementary information The online version contains supplementary material available at https://doi.org/10.1038/s42005-023-01390-y.

Correspondence and requests for materials should be addressed to Enrico Prati.

Peer review information *Communications Physics* thanks the anonymous reviewers for their contribution to the peer review of this work.