Harnessing a 256-qubit Neutral Atom Simulator for Graph Classification

Harnessing a 256-qubit Neutral Atom Simulator for Graph Classification

Edoardo Giusto*, Gabriele Iurlaro†, Bartolomeo Montrucchio†,
Alberto Scionti‡, Olivier Terzo‡, Chiara Vercellino†‡, Giacomo Vitali†‡§, Paolo Viviani‡,

*University of Naples, Federico II, Napoli, Italy
†DAUIN, Politecnico di Torino, Torino, Italy
‡Fondazione LINKS, Torino, Italy
§giacomo.vitali@linksfoundation.com

***Abstract*—Neutral atom platforms are analogue quantum simulators that offer the possibility to map graphs onto a 2D qubit register using programmable Rubidium atoms arrays, whose valence electrons' energy state is used as qubits, using optical tweezers. This makes it possible to implement algorithms for solving graph combinatorial optimization and Quantum Machine Learning (QML) tasks, such as graph classification. However, the restrictions of real hardware, as well as the very low number of publicly available machines, make such implementation non-trivial. In this work, we manage to compute the Quantum Evolution Kernel (QEK) to extract the features from graphs of the *PROTEINS* dataset using the 256-qubits Aquila platform (available through AWS) and then we apply classical Machine Learning (ML) techniques for the final classification. The method is benchmarked against classical kernels, resulting in slightly better performance, proving the effectiveness of the method, even in the case of a noisy quantum simulator.**

***Index Terms*—Quantum Simulation, Quantum Machine Learning, Graph Classification, Neutral Atoms**

I. Introduction

Quantum Machine Learning is one of the most recent and interesting fields of application of Quantum Computing (QC). The wide range of QC technologies in rapid development in recent years, all with their own set of characteristic, such as relaxation times, qubit connectivity, gate fidelities, etc., contribute to create a complex scenario where it is not straightforward to choose the best suited combination of algorithm and hardware platform for a specific Machine Learning (ML) task.

In this sense, quantum simulators are a QC typology very different from most well-known platforms, such as IBM superconducting machines, as they operate in analogue mode, i.e., the qubit state evolves following the machine Hamiltonian which depends on the specific technology employed. This typology of quantum computers allows to natively explore quantum system, such as spin models, which have applications especially in the field of quantum chemistry and new materials development.

Among simulators, neutral atom platforms [1] are rapidly developing thanks to some peculiarities. These machines operate optical tweezers to arbitrarily position alkaline atoms [2], [3], commonly Rubidium (^{87}Rb), in a 2D plane, while the valence electron's energy state serves as the two-state qubit, for example, the ground state $|g\rangle$ as $|0\rangle$ and a high-energy state (e.g., $70S$ for ^{87}Rb), called Rydberg state $|r\rangle$, as $|1\rangle$. Furthermore, the connectivity of the qubits can be tuned due to the Rydberg blockade effect, which creates entanglement among the qubits within the Rydberg radius r_b. In fact, the valence electron can be excited to a high-energy level through a dedicated Rydberg laser. In formal terms, the Hamiltonian of a typical neutral atom simulator is:

$$\hat{\mathcal{H}}(t) = \sum_{i=1}^{n} \frac{\hbar\Omega_i(t)}{2}(e^{i\phi(t)}\left|0_i\right\rangle\left\langle 1_i\right| + e^{-i\phi(t)}\left|1_i\right\rangle\left\langle 0_i\right|) - \sum_{i=1}^{n}\hbar\Delta_i(t)\hat{n}_i + \sum_{j>i}\frac{C_6}{|\overrightarrow{x}_i - \overrightarrow{x}_j|^6}\hat{n}_i\hat{n}_j, \quad (1)$$

where Ω_i and Δ_i are the time-dependent Rabi frequency and detuning, respectively, acting on the i-th atom (both measured in $rad/\mu s$), ϕ is the phase of the Rabi drive, $\overrightarrow{x}_i$ is the position of the i-th atom, $\hat{n}_i$ is the operator $\left|1_i\right\rangle\left\langle 1_i\right|$ which counts the atoms in the excited state and C_6 is the Rydberg interaction coefficient, whose value depends on the chosen Rydberg level. The last term, the Van der Waals interaction, is often rewritten using $V_{ij} = \frac{C_6}{|\overrightarrow{x}_i - \overrightarrow{x}_j|^6}$. With this formalization, the Rydberg radius of a neutral atom can be defined as $r_b = (\frac{C_6}{\hbar\sqrt{\Omega^2+\Delta^2}})^{1/6}$. When two atoms atoms are inside r_b, we have that $\Omega \ll V_{ij}$ $\Delta \ll V_{ij}$, and they are in the Blockade regime [4], [5], i.e., the $|11\rangle$ is energetically prohibited and therefore suppressed. The programmability of the qubit array [6] and the connectivity characteristic of these devices allow the mapping of graphs to the register by associating graph nodes to qubits, which share an edge when $|\overrightarrow{x}_i - \overrightarrow{x}_j| < r_b$. It is important to note that, while each atom can be theoretically fully addressable, in practice the available neutral atom platforms are at the moment restricted to global pulses, i.e., the Ω and Δ terms can be taken outside of the summations of Eq. 1. This means that qubits in the register represent Unit-Disk (UD) graphs. For this reason an embedding step is necessary, as detailed later.

The features just mentioned allow one to apply quantum algorithms to several problems from different domains, such as graph combinatorial optimization and ML techniques. In this paper, we describe the development and implementation of a QML technique for graph classification, where we use the

time evolution determined by the neutral atom Hamiltonian to extract features from a test dataset's graphs, compute the Quantum Evolution Kernel (QEK) defined in [7], and use a Support Vector Machine (SVM) to classify them. Finally, the method is benchmarked against a well-known classical kernel. In particular, we perform a binary classification of proteins in the PROTEINS dataset [8], [9] in the two-class labeled as *enzymes* and *non-enzymes*. To this end, we used the Aquila 256-qubit neutral atom simulator built by QuEra Computing [10], available through Amazon Web Services (AWS) Braket. This paper is structured as follows: first we report the context of this work, together with references to relevant bibliography. In Section III we described the methodology of the developed classification procedure and the characteristics of the PROTEINS dataset, as well as the preprocessing steps needed for the QEK implementation on neutral atom simulators. In Section V we report the results obtained by emulating with classical resource the quantum evolution on a small subset of the considered dataset and detail the pulse optimization procedure performed. Finally, in Section VI we compare the classification results obtained using Aquila with a well-known classical kernel and summarize the results of the work in Section VII.

II. Related Work

Graphs-based data are widely used in scientific and industrial applications. From telecommunication [11] to biology [12], [13] and social science [14], they allow describing interactions among data.

However, given their intrinsic complexity, algorithms that consume graph data are usually very resource-hungry. The exponentially large Hilbert space offered by quantum computers has prompted the interest of scientists and domain experts for this type of application, especially in the field of QML [15]–[17]. Some examples are quantum convolutional neural networks [18]–[20]. In the field of quantum kernels, a version developed for universal gate quantum computers has been presented in [21]. Quantum simulators have become available to the scientific community only in recent years, and there is not a large literature on QML techniques applicable to these typologies of QC, especially in the case of neutral atoms. Since these type of platforms offers the possibility to create lattices of qubit defined by the user, they have the potential to efficiently implement kernels for graph classification. In this context, the QEK defined by Henry et al. [7] can extract features from graphs depending on their connectivity and structure. However, the authors validated the method using only classical emulation of quantum systems, leaving a full and optimized implementation on real quantum simulators for the future. A further generalization of this approach was developed in [22], while other works explored the theoretical foundation of geometric QML [23], [24]. In [25], the QEK has been validated using a 32 qubit simulator with limited connectivity. However, the used Quantum Processing Unit (QPU), besides having a low qubit number, is also restricted to triangular lattice, limiting the embedding flexibility.

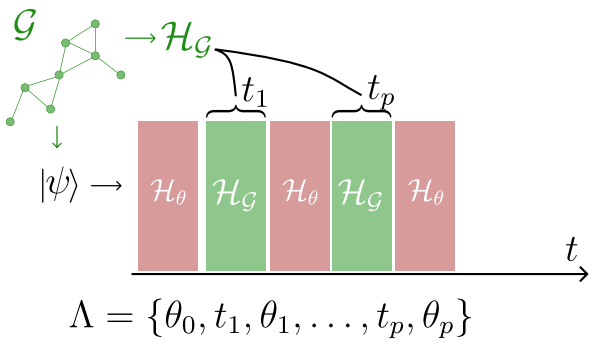


Fig. 1: Layered time evolution, highlighting the order of application of the Hamiltonians and the parameters Λ.

In this context, classical ML methods have been developed to embed general graphs into the neutral atom simulator's register [26], [27]. The goal of this work is to leverage the cited QEK and embedding method in order to implement an optimized procedure to classify for the first time a dataset composed of graphs with up to 256 nodes and arbitrary connectivity using the Aquila simulator.

III. Methodology

In this work, we implement and execute the QKE defined in [7] on a real quantum simulator and use it to classify proteins from the `PROTEINS` dataset with a SVM. Before going into the details of the implementation, we will provide an overview of the overall procedure.

The QKE we are considering is composed of several layers of alternating parameterized constant pulses. In detail, for a given graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ we define the mixing Hamiltonian $\hat{\mathcal{H}}_\mathcal{G}$ and the Ising Hamiltonian $\hat{\mathcal{H}}_\theta$ as,

$$\hat{\mathcal{H}}_\mathcal{G} = \sum_{(i,j)\in\mathcal{E}} \hat{\sigma}_i^z \hat{\sigma}_j^z$$

$$\hat{\mathcal{H}}_\theta = \theta \sum_{i\in\mathcal{V}} \hat{\sigma}_i^y$$

Both Hamiltonians can be natively implemented in available neutral atom simulators by applying global Rabi pulses. Considering two layers, the parameter set

$$\Lambda = \{\theta_0, t_0, \theta_1, t_1, \theta_2\}$$

fully defines the Hamiltonians and, therefore, the time evolution of the system.

For a given Λ, the full *hybrid quantum-classical* procedure is detailed, summarized in Figure 1:

1) **Quantum algorithm**: for every graph sample $\mathcal{G}_i$ and its position, a quantum register is instantiated and let to evolve following the Hamiltonian $\mathcal{H}_\Lambda$ defined by the parameters Λ. A set of measurements $M_{\mathcal{G}_i}$ is then obtained.

2) **Classical post-processing**: a probability distribution is obtained from each $M_{\mathcal{G}_i}$ by computing the corresponding interaction energy. The output of this step is a set of probability distributions $\mathcal{P}_{\mathcal{G}_i}$ for each graph.
3) **Machine Learning algorithm**: the set of $\mathcal{P}_{\mathcal{G}_i}$ is used to compute a kernel function between each graph employed in the training of the SVM. Finally, the optimal SVM hyperparameters have been selected through a K-fold cross-validation.

After the Hamiltonian evolution, we sample the final state $|\psi_f\rangle$ measuring an observable $\hat{\mathcal{O}}$ with $\{|o_i\rangle, \dots |o_K\rangle\}$ eigenstates. From that we build a probability distribution

$$\mathcal{P}_{\mathcal{G}}^{\hat{\mathcal{O}}}(\Lambda) = (p_1, \dots p_K), \;\; p_i = |\langle o_i|\psi_f\rangle|^2$$

Given two probability distribution $\mathcal{P}$ and $\mathcal{P}'$, the Jensen-Shannon divergence is defined as:

$$JS(\mathcal{P}, \mathcal{P}') = H\left(\frac{\mathcal{P}+\mathcal{P}'}{2}\right) - \frac{H(\mathcal{P})+H(\mathcal{P}')}{2}$$

where $H(\cdot)$ is the Shannon entropy of a probability distribution, defined as

$$H(\mathcal{P}) = -\sum_k p_k \log p_k$$

The image of $JS(\cdot,\cdot)$ is the closed set $[0, \log 2]$, and reach the maximum value $(\log 2)$ when the distribution have disjoint support. Finally, the graph kernel function of two graphs, $\mathcal{G}$, $\mathcal{G}'$ and their associated probability distributions $\mathcal{P}$ and $\mathcal{G}'$ is defined as:

$$\mathcal{K}_\mu(\mathcal{G}, \mathcal{G}') = e^{-\mu\, JS(\mathcal{P}, \mathcal{P}')} \tag{2}$$

with image in $[2^{-\mu}, 1]$. The μ hyperparameter can be optimized with the pulse parameters Λ, allowing the kernel to take value in a wider range.
As described in Section V, we used a Bayesian optimization procedure to obtain the best performing Λ. For this purpose, we use a subset of PROTEINS for which the algorithm can be classically emulated with reasonable time resources and time. Once defined the optimal Λ, we use quantum resources, namely the Aquila neutral atom simulator, for a larger subset of the considered dataset and compare the results of the classification.
In the following Sections, we give the details of each step of the procedure.

IV. Dataset and Preprocessing

PROTEINS is one of the most used datasets for benchmarking graph machine learning algorithms. Proteins are the perfect candidate to be modeled as graphs since they are macromolecules consisting of amino acids chain, disposed in a 3-dimensional space. In fact, a graph $\mathcal{G} = (\mathcal{V}, E)$ is obtained by modeling each amino acid as a node $v \in \mathcal{V}$, and an edge between amino acids if they are less than $6\,\mathring{A}$ apart in space. Each protein comes with a binary label that describes whether it is an *enzyme* or not. One of the main problems when working with graph data is the absence of homogeneous data representation. In [28], the authors tried to overcome this problem by collecting and uniforming graph data representation. It comes in the form of different .txt files, each describing graphs, nodes, edges, and labels. The dataset consists of 1113 graphs, divided into 663 enzymes and 450 non-enzymes. Some overall statistics on the dataset are shown in Table I.

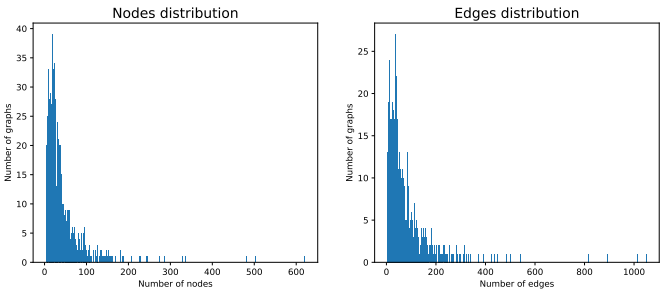


Fig. 2: Nodes and edges distribution in PROTEINS dataset.

Overall graph statistics	
min number of nodes	4
avg number of nodes	39.5
max number of nodes	620
min number of edges	5
avg number of edges	72.81
max number of edges	1049

TABLE I: Overall statistics of the PROTEINS dataset.

In order to fit the graph on a quantum register, it should be limited to a maximum of 256 nodes (i.e., the maximum number of physical qubits available on Aquila). By plotting the node number distribution, one can notice that most of the graphs contain less than 100 nodes. Details on the distribution of nodes and edges are shown in Figure 2.

Since emulation time on classical machines grows exponentially with the number of qubits (i.e., the graph nodes) the dataset has been reduced, producing two different but overlapping datasets:

- PROTEINS256: limited to graphs with a maximum of 256 nodes (Aquila current limit, corresponding to the number of physical qubits). It consists of 276 graphs.
- PROTEINS12: limited to graphs with a maximum of 12 nodes. The number of nodes is selected based on a qualitative benchmarking of emulation time. It consists of 143 graphs.

It is important to note that both datasets are also limited by the constraint of being representable as UD graphs [29], as detailed later.

A. Data preprocessing

As described in Section III, computing the probability distribution associated with each graph requires measurements from a quantum state that evolves following a time-dependent Hamiltonian (i.e., the alternation of mixing Hamiltonian $\mathcal{H}_\theta$

and Ising Hamiltonian $\mathcal{H}_\mathcal{G}$) that depends on the graph topology. Given a graph, two methods can be implemented:

1) In [7] a simple rescaling method such that a minimum atom distance of $5\,\mu m$ is applied. However, this approach strongly limits the number of graphs embeddable in a 2D register with the hardware constraints.
2) Find the UD representation of the graph. This way, the topology of the interaction of neutral atoms machine directly corresponds to the Ising Hamiltonian. This method is the only one that is applicable to real quantum hardware without approximating the Hamiltonian.

Since the final objective of this work is to assess the quality of the methods on real hardware, the second method is the only one that allows the execution on a quantum computer. As mentioned before, the first step is to embed the dataset's graphs into the register, which means finding the related UD graphs. For this, the DEN model has been used as a basis. All details about UD embedding with DEN can be found in [26], [27].

Some additional constraints had to be considered for the embedding into the Aquila register. The neural-enhanced framework mentioned above allows further generalization, thanks to its flexibility. In particular, these are the additional constraints of the chosen platform for the CUDG (Constrained Unit Disk Graph) problem:

- *Register area constraint*: the atoms' position should belong to a rectangle of size $75\mu m \times 76\mu m$.
- *Row spacing constraint*: atoms should be positioned in discrete rows, with $4\mu m$ spacing in between.

The last constraint is specific to Aquila, it has no correspondence with other neutral atom platforms, and is related to how the Aquila scheduler loads the atoms into the register. The first constraint can be easily modeled by modifying the multiplier of the *tanh* activation function, so that vertexes coordinates belong to the specific register's dimensions. The row spacing constraint is instead implemented through an approximation layer that adjusts the position in discrete rows, and by adding a new component to the Embedding Loss Function (ELF) that penalizes pairs that are vertically closer than 4 μm. The ELF, defined in [27], is designed to have a global minimum value for each feasible embedding found by the DEN model.

Finally, the UD constraints are then re-verified, in order to produce a feasible embedding.

B. Embedded dataset

The embedded datasets are produced by first filtering on the number of nodes and then employing DEN as described above, which reduces the number of graphs by filtering out graphs that cannot have a feasible UD embedding. The characteristics of the generated datasets are presented in Table II.

	Number of graphs	#Class 1	#class 2
PROTEINS12	143	33	110
PROTEINS256	276	108	168

TABLE II: Final dataset composition.

An example of a starting graph and the generated feasible embedding is presented in Figure 3.

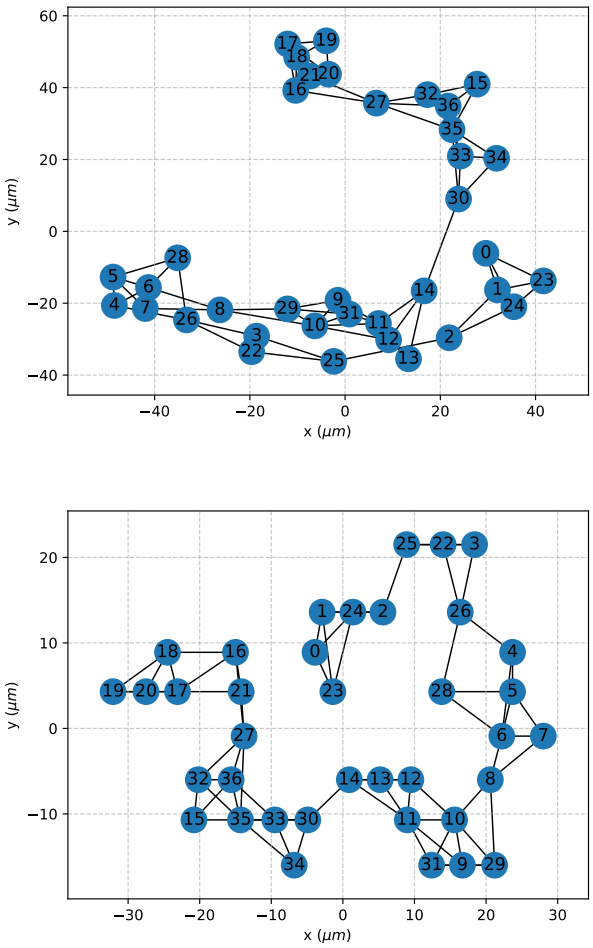


Fig. 3: example of initial not-UD graph (top) and the associated UD embedding produced by the modified DEN model (bottom). The discrete row constraint of Aquila is visible.

Unfortunately, the DEN model does not guarantee convergence, since it is an approximation method employing neural networks. Not all graphs admit a UD representation, and this problem is more evident when the position space is limited to $\mathbb{R}^2$. However, the number of embedded graphs is sufficient to make a preliminary analysis of the QEK benchmark. It is important to note that the reduced dataset (i.e., `PROTEINS12`) presents a high-class imbalance. Even if `PROTEINS256` does not present the issue to the same extent (class 1, representing 39% of the total dataset, while in `PROTEINS12` only the 23%), in the following sections some methods to overcome the imbalance are analyzed and then applied.

V. Emulation on Classical Hardware

Classical emulation of the quantum evolution is performed using *Bloqade* [30], a framework written in *Julia* [31] developed by QuEra for experimenting and interacting with

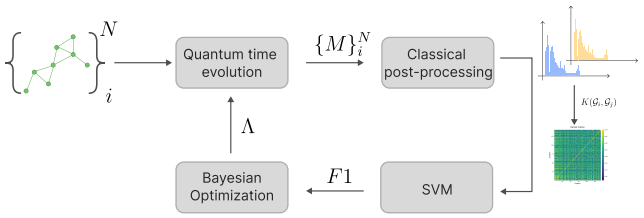


Fig. 4: The full Hybrid quantum-classical emulation approach.

their neutral atom quantum simulator. Through its ecosystem, it supports emulating quantum systems, measuring different observables (even user-defined), and interacting with neutral atom hardware by validating and creating a representation for the Hamiltonian that can be sent to Aquila and run. In particular, the used libraries are:

- *BloqadeODE*: which contains all the code for defining and emulating the time evolution of a quantum system by solving the Schrödinger equation.
- *BloqadeSchema*: which contains all the functions for validating the Hamiltonian, in particular for checking the feasibility of the atoms' position, and contains the function for smoothing the waveform in the case of piecewise protocols, according to the bandwidth of the lasers.

As described before, the emulation step is very computationally onerous, but still necessary, for two main reasons:

1) Confirm the validity of the method before executing on quantum resources.
2) Each iteration of the Bayesian optimization algorithm, as described later, requires a total number of measurements (also called shots) $N_{shots} = n_{shots} \times n_{graphs} = 276 \cdot 10^3$ (where $n_{graphs} = 276$ is the number of graphs and $n_{shots} = 10^3$ is the number of shots per graph) to run on the quantum computer, which is too resource-expensive. This step is required to find the waveform parameters Λ.

The full *hybrid quantum-classical* emulation approach is summarized in Figure 4.

After the steps reported in Section III, the **Bayesian Optimization** phase receives the new pair $(\Lambda, f(\Lambda)$ and updates the posterior probability distributions. The acquisition function produces a new set of parameters Λ for the next iteration.
In the following, the experimental settings and results are discussed.

A. Probability distribution

As described in Section III, the core of the method is the computation of a graph kernel as the distance between two probability distributions representing the graphs. Aquila allows measurements only in the computational basis. The *energy observable* is then computed by post-processing the set of measurements.

Given a graph $\mathcal{G}$, and a set of measurements M:

$$M = \{m_1, \dots, m_k\}, m_i = [n_1, \cdots, n_n],$$
$$n_i = \begin{cases} 1 & \text{if atom i is measured in } |g\rangle \\ 0 & \text{if atom i is measured in } |r\rangle \end{cases}$$

The used number of shots is $k \leq n_{shots}$ because the measurement process can fail (due to an erroneous register initialization), in which case the shot is discarded. Each measurement m_i is composed by a *bitstring* of length $|\mathcal{V}|$ (i.e. the number of nodes/atoms). For each measurement m_i we can associate an energy:

$$e_i = \sum_{j<k} V_{jk} n_j n_k$$

By repeating the process for every correctly initialized shot, a set of energies related to each single graph is computed, indicated as $E_{\mathcal{G}_i}$. In order to have comparable distributions, they should have the same support. For this reason, a binning procedure is used.

Once every energy distribution is computed, these two values are calculated:

$$e_1 = \min_i \{\min_{e_j} E_{\mathcal{G}_i}\}\ , \quad e_2 = \max_i \{\max_{e_j} E_{\mathcal{G}_i}\}$$

Where e_1 and e_2 are respectively the minimum and maximum measured energy. Selecting 100 equal-sized bins in the interval $[e_1, e_2]$ allows to compute a probability distribution $\mathcal{P}(\mathcal{G})$. Bins are normalized to sum up to 1 by dividing by the number of measurements k. Examples of the probability distributions and the associated graphs can be found in Figure 5.

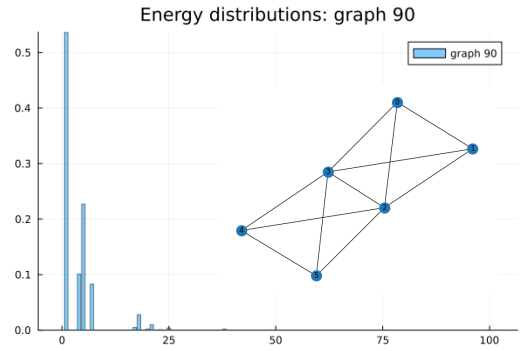
B. Kernel estimation and training protocol

Once the probability distribution is estimated, the kernel is computed as described in Eq. 2. The only hyperparameter involved in this task is μ. Following [7], μ is set equal to 1. Further hyperparameter tuning is possible, knowing that μ influences the image of the kernel. With $\mu = 1$ $K(\cdot) \in \left[\frac{1}{2}, 1\right]$, while for bigger values the image increases, and in the limit of $\mu \to \infty$ reaches $[0, 1]$. Once the QEK is estimated, it is organized in a kernel matrix K. The entries are organized as:

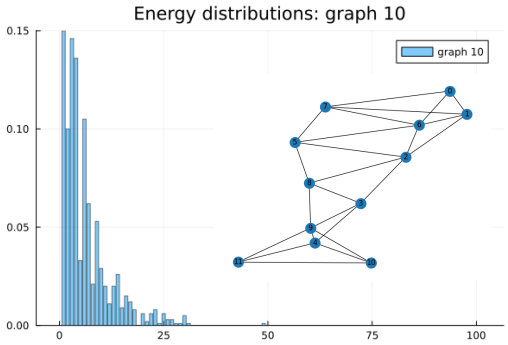
$$K_{i,j} = K(\mathcal{G}_i, \mathcal{G}_j)$$

The kernel matrix expresses the similarity between pairs of graphs and can be visualized graphically using a heatmap, as in Figure 6. As we can notice, there is a brighter zone in the bottom-left corner associated to higher kernel values. This group is composed of graphs labeled as *enzymes*, demonstrating qualitatively that the quantum kernel is able to find structures in data. The kernel matrix is then used to train a SVM. To evaluate the approach and tune the SVM hyperparameters, validation data are generated using a K-fold cross-validation scheme (with $K = 10$), in combination with a *grid search* to find the optimal parameters. The hyperparameters tuned in this phase are:

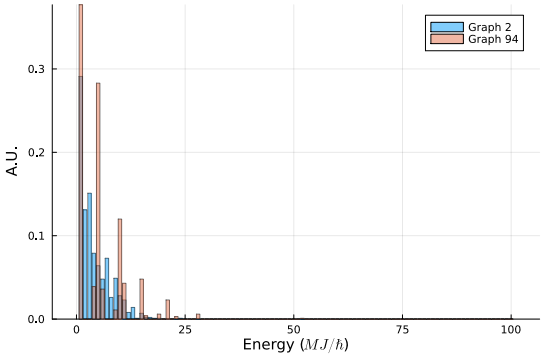
- C: 100 points $[10^{-4}, 10^4]$, logarithmic scale.
- w: 30 pairs $[1, x_i]$ with x_i in $[1, 1000]$.



(a) Energy distribution and the associated graph.



(b) Energy distribution and the associated graph.



(c) Comparison of the energy distributions of two graphs belonging to different classes.

Fig. 5: Example of obtained energy distributions for two different graphs.

For every possible combination of the parameter, 10 models are trained on 9 fold of the data, and validated on the remaining. Each model score is then collected and averaged, producing single metrics used for evaluation.

The C hyperparameter expresses the possibility of the classifiers of making misclassification errors during the training procedure. This improves the generality of the solution and prevents overfitting. The additional pair of class weights w is used to prevent the model from predicting always the majority class. They act as a penalty for misclassifying the minority class.

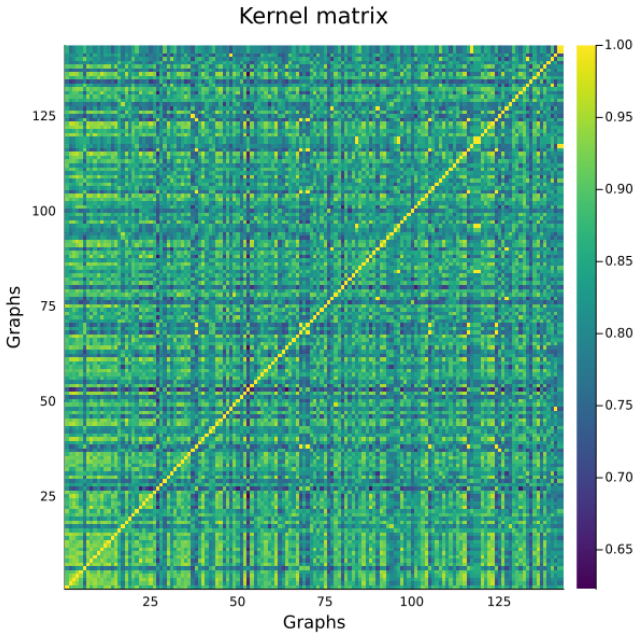


Fig. 6: Kernel matrix of PROTEINS12 dataset, with $\mu = 1$.

C. Bayesian optimization of Waveform parameters

In terms of pulse sequence, the selected approach consists of three pulses of a single qubit drive (i.e., the mixing Hamiltonian) alternated by two free evolutions (no pulse). The amplitude of the Rabi drive is kept constant and equal to the maximum amount reachable by the QPU ($\Omega_0 = 15.8\, rad/\mu s$, in order to reduce the overall evolution time, and consequently to reduce the noise sources). As discussed in Section III, the full dynamics can be realized using a time-dependent Hamiltonian with a $\Omega(t)$ governed by a pulse that alternates values where the drive is on (corresponding to the mixing Hamiltonian) with a period where the drive is off, corresponding to free evolution of the quantum state. Given this setup, we can redefine the parameter set as:

$$\Lambda_{BO} = \{\tau_0, t_0, \tau_1, t_1, \tau_2\}$$

Where τ_i represents the Rabi drive duration (and consequently the angle of rotation), while t_i represents the free evolutions. It is possible to retrieve the angle of rotation of the mixing Hamiltonian by remembering that $\theta_i = \Omega_0\, \tau_i$.
Given a set of parameters Λ, the training protocol described before returns the SVM hyperparameter set that reaches the best average score among the 10 folds. However, the pulse parameter has to be tuned, in order to find the pulse that extracts a good amount of knowledge (i.e., has the maximum F1 score). Formally this can be seen as a maximization problem, finding the Λ that maximizes $f : \mathcal{X} \to \mathbb{R}$, where $\mathcal{X}$ is the space of the parameters Λ, and f is the average selected score (F-1 in this case) among the 10 folds of the PROTEINS12 dataset. However, the evaluation of f is costly,

since it consists of training a model and computing the state several times for each graph, and common heuristic methods are not feasible. Bayesian optimization [32] is particularly well suited for optimizing *black-box functions*, in which evaluating different sets of parameters can be costly. The framework is made of two components:

- *Surrogate function* $\tilde{f}$: approximates the costly objective function starting from a set of evaluations of f. In Bayesian optimization, it is usually called *prior* and reflects the knowledge we have on the function f.
- *Acquisition function* $\alpha(x)$: indicates where to sample the next point to evaluate f.

It is common to select the surrogate function as a Gaussian Process $\mathcal{GP}$, characterized by a mean value μ and a covariance matrix Σ. The Gaussian Process so defined has a probability density function of the form:

$$P(X) = \frac{1}{\sqrt{(2\pi)^k|\Sigma|}} \cdot \exp\left(-\frac{1}{2}(X-\mu)^T\Sigma^{-1}(X-\mu)\right)$$

where k is the variable's dimensionality. The selected mean μ is Normal distributed as $\mathcal{N}(0, 10)$, while the covariance function is selected as the *Isotropic* $5/2$ *Matèrn Kernel*, with length scale $\ell = 10$ and signal standard deviation $\sigma = 10$:

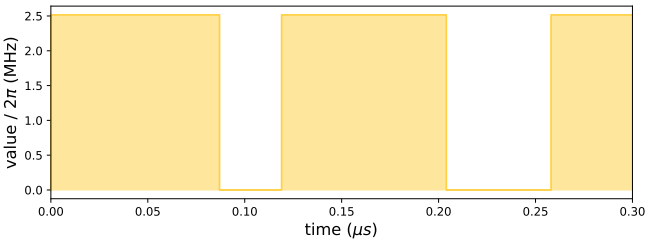
$$\begin{aligned} K(x, x') =& \sigma^2(1 + \sqrt{5|x-x'|/\ell + 5|x-x'|^2/(3\ell^2)}) \\ & exp\Big(-\sqrt{5|x-x'|/\ell}\Big) \end{aligned} \tag{3}$$

The acquisition function is the commonly employed *Upper Confidence Bound*, $\alpha(x) = \mu(x) + k\sigma(x)$, where k is a hyperparameter that expresses the trade-off between exploration (μ parameter) and exploitation (σ parameter). The kernel hyperparameters (length scale and signal standard deviation) are optimized every 10 iterations, using Maximum A Posteriori (MAP) estimate. The maximum number of iterations is set to 50. In addition, some constraints are applied to the optimized variables:

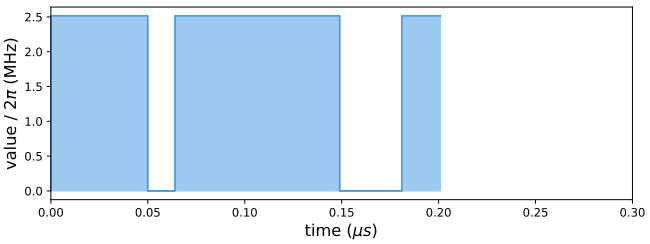
$$\begin{aligned} \tau_0 + t_0 + \tau_1 + t_1 + \tau_2 \quad & < 500ns \\ t_i > 5ns \quad & \forall i \in [0, 1] \\ \tau_i > 5ns \quad & \forall i \in [0, 1, 2] \end{aligned}$$

The first constraint limits the total execution time, preventing decoherence and consequently reducing both the simulation and classical emulation time, while the second and third constraints are related to the finite bandwidth of the optical components. The waveform obtained by the entire process on the `PROTEINS12` is then compared with the one obtained in [7], named here for simplicity Λ_a, in Figure 7.

In general, the two waveforms are similar. Both are characterized by short free-evolution and longer periods of $\mathcal{H}_{\theta_i}$. However, the total duration of Λ_a waveform is $317\,ns$, in contrast to the $201\,ns$ of the one found in this work. This difference of $\simeq 100\,ns$ has one major benefit: a shorter protocol is expected to produce less noisy results, allowing quantum simulations to be more similar to classical emulation and to produce higher quality results, as showed in the following.



(a) Λ_a parameter set.



(b) Λ_{BO} parameter set.

Fig. 7: Waveforms comparison.

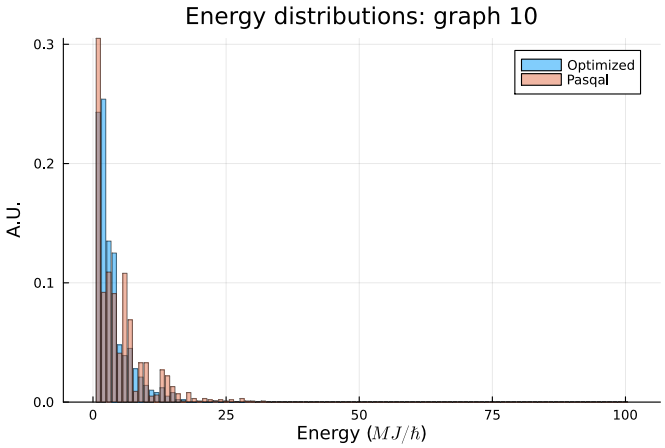


Fig. 8: Comparison of the energy distributions obtained using two different sets of waveform parameters on the same graph.

Another relevant difference consists in the mixing Hamiltonian duration: for Λ_a, the three mixing Hamiltonian have similar duration ($87\,ns$, $84\,ns$ and $72\,ns$), while Λ_{BO} is characterized by a long central pulse (comparable to the one in Λ_a waveform, $85\,ns$ corresponding to a rotation $\simeq 80°$) and two shorter pulses of $50\,ns$ and $20\,ns$, corresponding respectively to a rotation of $45°$ and $18°$. It is possible to qualitatively compare the energy distribution of the same graph obtained with the two different Λ, as shown in Figure 8.

D. Results on PROTEINS12

The results obtained in the emulation experiment can be presented, in the form of *F1 score*, *Accuracy*, *Precision* and *Recall* metrics. As a benchmark, the method is compared with the version using the Λ_a parameter set, and with a classical algorithm, the *Shortest Path graph Kernel* (SPK) [33]. The emulation platform is based on a *Dell-XPS 15 7590*. The

results are presented in Table III.

	F-1 (%)	Accuracy (%)	Precision (%)	Recall (%)
Λ_a	41.9	44.8	29.3	89.1
Λ_{BO}	**46.3**	52.4	37.1	85
Shortest Path kernel	44.1	**61.4**	35	68

TABLE III: Performance comparison of QEK using Λ_a and Λ_{BO} parameter sets with the SPK on PROTEINS12.

As expected, the performance scores of all the kernels on the reduced dataset are generally low due to the small size of the training sample, even if using a K-fold-cross validation approach with an high K ($K = 10$). However, the kernel computed using Λ_{BO} outperforms the one obtained with Λ_a by $\simeq 5\%$ on the F1 score. The improvement on the F1 score is reflected also in an higher accuracy (44% against 52.4%) and precision (29.3% against 37%). Notably, the implemented QEK method performs even better than the classical SPK in terms of F1 score, but with a smaller margin (44.1% of the SPK vs. the 46.1% of QEK). The same is true for precision and recall. However, the accuracy of the classical model is higher, because of the high-class imbalance. This shows how in this case the accuracy score can be misleading.

VI. Simulation on Aquila: Results and Discussion

We proceed to execute the proposed algorithm on the real noisy quantum simulator Aquila for both datasets.
The interaction with the hardware is ensure by the *Bloqade-Hardware* library, which in particular provides support to properly formatting the defined Hamiltonian and to check the compliance with the hardware constraints. Before delving into the results, it is important to report the number of shots that have been performed on the quantum hardware. Each graph in the dataset is associated with a single Hamiltonian evolution, i.e., a single task for Aquila. As per the classical emulation case (see Section V), the total number of measurement for the whole dataset is $n_{shots} \times N_{graphs} = 276 \cdot 10^3$ for each set of parameters Λ. In the following, we report the detailed results.

A. Comparison of energy distributions

Before executing the QKE on the full dataset, a random subset is extracted and the results of the quantum simulation are compared to the one obtained with the classical emulation. The comparison is done in terms of the obtained energy distribution since computing the statevector would require an exponential number of measurements. Figure 9 shows the energy distribution comparison for an example graph. The two distributions are similar, demonstrating a low impact of noise on the approach. Notably, in simulation there is a higher probability to have very low energy configuration (i.e., almost zero atoms in the Rydberg state) than emulation. This is probably due to the coupling to the ground state, which is missing in the considered noiseless classical emulation. At the same time, some higher energy configurations are present in the simulation. These can be partially explained by the detection errors that affect atoms in the ground state $|g\rangle$ to be detected as in the Rydberg state $|r\rangle$, increasing the probability of detecting a double-excited Rydberg state.

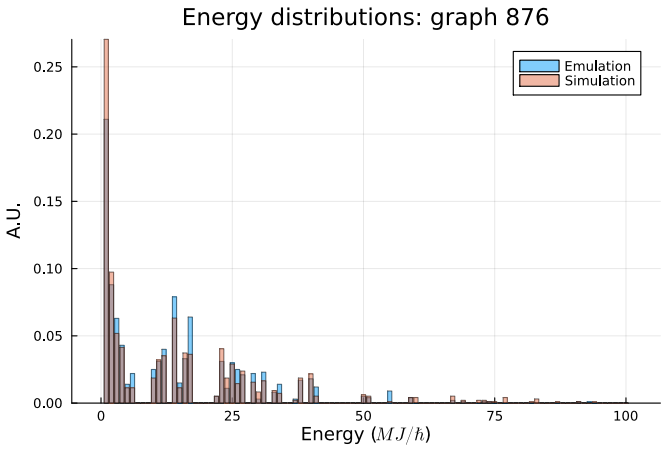


Fig. 9: Comparison between the emulated and simulated energy distributions a randomly selected graph.

B. Results on PROTEINS12

The experiment is repeated with the same experimental setup presented before for the full PROTEINS12, in order to compare the classification of emulated and simulated QEK. The experiment is repeated with both Λ_{BO} and Λ_a sets. As reported in the Section V, the Λ_{BO} protocol has a 33% shorter duration. This difference in circuit duration can help to evaluate the effect of noise for longer protocols. The result presented in Table IV confirms that the method is effective, even in the case of the real noisy simulation.

	F-1 (%)	Accuracy (%)	Precision (%)	Recall (%)
Λ_a (Emulation)	41.9	44.8	29.3	89.1
Λ_{BO} (Emulation)	**46.3**	52.4	37.1	85
SPK	44.1	**61.4**	35	68
Λ_a (Aquila)	**48**	62.2	35.9	80.8
Λ_{BO} (Aquila)	**49**	63.7	39.5	78

TABLE IV: Performance comparison of QEK using Λ_a and Λ_{BO} sets, in emulated and simulated execution, with the SPK.

The simulation on the Aquila quantum computer with the optimized waveform parameters has the best result, outperforming both the chosen classical method and the emulated one (of about 3% in the F1 metric). This result is also reflected in the accuracy metric of 63.7%, the highest among the considered methods. Interestingly, also in the case of the Λ_a parameter set, the simulation scores are higher than the emulated ones.

C. Results on PROTEINS256

Finally, the experiment using the full dataset has been conducted. The results can be found in Table V.

	F-1 (%)	Accuracy (%)	Precision (%)	Recall (%)
Aquila simulation	**65,6**	**65,4**	**55,1**	86,6
SPK	65,3	64,9	53,5	**87,1**

TABLE V: QEK and SPK performances on PROTEINS256.

In general, performance scores are higher (about 20% on the F1 metric) than the PROTEINS12 case due to the larger

size of the dataset and the higher class balance. Even if a weighting training objective and a cross-validation approach with a high value of K are used, the number of training samples is not enough to provide reliable classification results. Even in this experimental setting, the simulation of the QEK on Aquila with the optimized parameter has performances that are comparable to that of the classical SPK kernel. It is important to note that, has mentioned before, the Λ_{BO} has been optimized only on the PROTEINS12 dataset in emulation, due to limited computational and quantum resources available. This could have had a detrimental effect on the results obtained. Moreover, the larger difference between the QEK and the SPK for the PROTEINS12 case with respect to the PROTEINS256 one seems to experimentally corroborate the claim that quantum kernels are able to extract features more efficiently than classical ones using fewer training data [34].

In order to save quantum resources and time in future works (Aquila is able to sample at an approximate rate of 10 Hz [10] with an associated cost per shot), a statistical analysis can be performed on the number of shots required to accurately compute the energy distribution. To this end, a sampling approach has been used. From the whole set of measurements, a number k of shots is drawn randomly and without replacement, simulating an experiment with k number of shots. Then, the same post-processing and cross-validation approach was used. The results are shown in Table VI.

number of shots	F-1 (%)	Accuracy (%)	Precision (%)	Recall (%)
10	59,3	63,1	52,9	71
100	**65,1**	64,4	53,8	87
1000	**65,6**	65,4	55,1	86,6

TABLE VI: PROTEINS256 performances against the number of shots.

The overall performance does not degrade much when the number of shots is reduced to 100 (about 1% average degradation), while it rapidly decreases when the number of samples is reduced to 10.

VII. Conclusion

This work demonstrated the ability to successfully implement a QML algorithm for graph classification on the Aquila simulator. By mapping graphs from the PROTEINS dataset onto the register of Rydberg atoms using state-of-the-art ML methods and computing the Quantum Evolution Kernel, competitive classification performance was achieved compared to classical graph kernel methods like the Shortest Path Kernel. An optimization procedure based on Bayesian optimization was used to find a set of waveform parameters for the quantum evolution that outperformed previously reported parameters found in literature. When simulated on the noisy Aquila hardware, the optimized QEK achieved an F1 score of 65.6% on the full PROTEINS256 dataset, comparable to 65.3% for the SPK.

Despite the limited number of qubits and noise present on current neutral atom quantum hardware, these results show the feasibility of implementing useful QML workloads on these systems. As neutral atom architectures scale up to more qubits and improve in terms of fidelity and coherence times, their inherent connectivity mappings may provide advantages for graph analytics tasks. In summary, this study serves as an important first proof-of-concept demonstration for executing QML workloads for graph problems with hundreds of nodes and arbitrary connectivity on neutral atom quantum computing platforms. As this emerging quantum hardware develops further, exploring the interplay of algorithms and hardware will be crucial to unlocking its potential advantages.

Acknowledgment

We would like to thank AWS for the research credit grant and QuEra for the support and useful discussions.

References

[1] Loïc Henriet, Lucas Beguin, Adrien Signoles, Thierry Lahaye, Antoine Browaeys, Georges-Olivier Reymond, and Christophe Jurczak. Quantum computing with neutral atoms. *Quantum*, 4:327, September 2020.

[2] Daniel Barredo, Vincent Lienhard, Sylvain de Léséleuc, Thierry Lahaye, and Antoine Browaeys. Synthetic three-dimensional atomic structures assembled atom by atom. *Nature*, 561(7721):79–82, September 2018.

[3] F. Nogrette, H. Labuhn, S. Ravets, D. Barredo, L. Béguin, A. Vernier, T. Lahaye, and A. Browaeys. Single-atom trapping in holographic 2d arrays of microtraps with arbitrary geometries. *Physical Review X*, 4(2), May 2014.

[4] Alpha Gaëtan, Yevhen Miroshnychenko, Tatjana Wilk, Amodsen Chotia, Matthieu Viteau, Daniel Comparat, Pierre Pillet, Antoine Browaeys, and Philippe Grangier. Observation of collective excitation of two individual atoms in the rydberg blockade regime. *Nature Physics*, 5(2):115–118, January 2009.

[5] D. Jaksch, J. I. Cirac, P. Zoller, S. L. Rolston, R. Côté, and M. D. Lukin. Fast quantum gates for neutral atoms. *Physical Review Letters*, 85(10):2208–2211, September 2000.

[6] M. Morgado and S. Whitlock. Quantum simulation and computing with rydberg-interacting qubits. *AVS Quantum Science*, 3(2), May 2021.

[7] Louis-Paul Henry, Slimane Thabet, Constantin Dalyac, and Loïc Henriet. Quantum evolution kernel: Machine learning on graphs with programmable arrays of qubits. *Physical Review A*, 104(3):032416, 2021.

[8] Paul D Dobson and Andrew J Doig. Distinguishing enzyme structures from non-enzymes without alignments. *Journal of molecular biology*, 330(4):771–783, 2003.

[9] Karsten M Borgwardt, Cheng Soon Ong, Stefan Schönauer, SVN Vishwanathan, Alex J Smola, and Hans-Peter Kriegel. Protein function prediction via graph kernels. *Bioinformatics*, 21(suppl_1):i47–i56, 2005.

[10] Jonathan Wurtz, Alexei Bylinskii, Boris Braverman, Jesse Amato-Grill, Sergio H Cantu, Florian Huber, Alexander Lukin, Fangli Liu, Phillip Weinberg, John Long, et al. Aquila: Quera's 256-qubit neutral-atom quantum computer. *arXiv preprint arXiv:2306.11727*, 2023.

[11] Tahereh Pourhabibi, Kok-Leong Ong, Booi H. Kam, and Yee Ling Boo. Fraud detection: A systematic literature review of graph-based anomaly detection approaches. *Decision Support Systems*, 133:113303, 2020.

[12] Athanasios Theocharidis, Stjin Dongen, Anton Enright, and Tom Freeman. Network visualisation and analysis of gene expression data using biolayout. *Nature protocols*, 4:1535–50, 10 2009.

[13] Giulia Muzio, Leslie O'Bray, and Karsten Borgwardt. Biological network analysis with deep learning. *Briefings in Bioinformatics*, 22(2):1515–1530, 11 2020.

[14] Carlos D. Correa and Kwan-Liu Ma. *Visualizing Social Networks*, pages 307–326. Springer US, Boston, MA, 2011.

[15] Maria Schuld and Nathan Killoran. Quantum machine learning in feature hilbert spaces. *Physical Review Letters*, 122(4), February 2019.

[16] Maria Schuld, Kamil Brádler, Robert Israel, Daiqin Su, and Brajesh Gupt. Measuring the similarity of graphs with a gaussian boson sampler. *Phys. Rev. A*, 101:032314, Mar 2020.

[17] Kaito Kishi, Takahiko Satoh, Rudy Raymond, Naoki Yamamoto, and Yasubumi Sakakibara. Graph kernels encoding features of all subgraphs by quantum superposition. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 12(3):602–613, September 2022.

[18] Iris Cong, Soonwon Choi, and Mikhail D. Lukin. Quantum convolutional neural networks. *Nature Physics*, 15(12):1273–1278, August 2019.

[19] Jin Zheng, Qing Gao, and Yanxuan Lv. Quantum graph convolutional neural networks, 2021.

[20] Guillaume Verdon, Trevor McCourt, Enxhell Luzhnica, Vikash Singh, Stefan Leichenauer, and Jack Hidary. Quantum graph neural networks, 2019.

[21] Vojtěch Havlíček, Antonio D. Córcoles, Kristan Temme, Aram W. Harrow, Abhinav Kandala, Jerry M. Chow, and Jay M. Gambetta. Supervised learning with quantum-enhanced feature spaces. *Nature*, 567(7747):209–212, March 2019.

[22] Péter Mernyei, Konstantinos Meichanetzidis, and İsmail İlkan Ceylan. Equivariant quantum graph circuits, 2022.

[23] Andrea Skolik, Michele Cattelan, Sheir Yarkoni, Thomas Bäck, and Vedran Dunjko. Equivariant quantum circuits for learning on weighted graphs, 2023.

[24] Martín Larocca, Frédéric Sauvage, Faris M. Sbahi, Guillaume Verdon, Patrick J. Coles, and M. Cerezo. Group-invariant quantum machine learning. *PRX Quantum*, 3(3), September 2022.

[25] Boris Albrecht, Constantin Dalyac, Lucas Leclerc, Luis Ortiz-Gutiérrez, Slimane Thabet, Mauro D'Arcangelo, Julia R. K. Cline, Vincent E. Elfving, Lucas Lassablière, Henrique Silvério, Bruno Ximenez, Louis-Paul Henry, Adrien Signoles, and Loïc Henriet. Quantum feature maps for graph machine learning on a neutral atom quantum processor. *Physical Review A*, 107(4), April 2023.

[26] Chiara Vercellino, Paolo Viviani, Giacomo Vitali, Alberto Scionti, Andrea Scarabosio, Olivier Terzo, Edoardo Giusto, and Bartolomeo Montrucchio. Neural-powered unit disk graph embedding: qubits connectivity for some qubo problems. In *2022 IEEE International Conference on Quantum Computing and Engineering (QCE)*, pages 186–196. IEEE, 2022.

[27] Chiara Vercellino, Giacomo Vitali, Paolo Viviani, Alberto Scionti, Andrea Scarabosio, Olivier Terzo, Edoardo Giusto, and Bartolomeo Montrucchio. Neural optimization for quantum architectures: graph embedding problems with distance encoder networks. In *2023 IEEE 47th Annual Computers, Software, and Applications Conference (COMPSAC)*, pages 380–389, 2023.

[28] Ryan A. Rossi and Nesreen K. Ahmed. The network data repository with interactive graph analytics and visualization. In *AAAI*, 2015.

[29] Brent N Clark, Charles J Colbourn, and David S Johnson. Unit disk graphs. *Discrete mathematics*, 86(1-3):165–177, 1990.

[30] Bloqade.jl.

[31] The Julia Programming Language.

[32] Peter I Frazier. A tutorial on bayesian optimization. *arXiv preprint arXiv:1807.02811*, 2018.

[33] Karsten M Borgwardt and Hans-Peter Kriegel. Shortest-path kernels on graphs. In *Fifth IEEE international conference on data mining (ICDM'05)*, pages 8–pp. IEEE, 2005.

[34] Matthias C. Caro, Hsin-Yuan Huang, M. Cerezo, Kunal Sharma, Andrew Sornborger, Lukasz Cincio, and Patrick J. Coles. Generalization in quantum machine learning from few training data. *Nature Communications*, 13(1), August 2022.