






Quantum feature maps for graph machine learning on a neutral atom quantum processor

Boris Albrecht ^{1,*}, Constantin Dalyac,^{1,2,*} Lucas Leclerc,^{1,3,*} Luis Ortiz-Gutiérrez,^{1,*} Slimane Thabet,^{1,2,*} Mauro D'Arcangelo ¹, Julia R. K. Cline ¹, Vincent E. Elfving,¹ Lucas Lassablière,¹ Henrique Silvério ¹, Bruno Ximenez,¹ Louis-Paul Henry,¹ Adrien Signoles ¹ and Loïc Henriot^{1,†}

¹PASQAL, 7 Rue Léonard de Vinci, 91300 Massy, France

²LIP6, CNRS, Sorbonne Université, 4 Place Jussieu, 75005 Paris, France

³Université Paris-Saclay, Institut d'Optique Graduate School, CNRS, Laboratoire Charles Fabry, 91127 Palaiseau, France



(Received 5 December 2022; accepted 4 April 2023; published 19 April 2023)

Using a quantum processor to embed and process classical data enables the generation of correlations between variables that are inefficient to represent through classical computation. A fundamental question is whether these correlations could be harnessed to enhance learning performances on real data sets. Here we report the use of a neutral atom quantum processor comprising up to 32 qubits to implement machine learning tasks on graph-structured data. To that end, we introduce a quantum feature map to encode the information about graphs in the parameters of a tunable Hamiltonian acting on an array of qubits. Using this tool, we first show that interactions in the quantum system can be used to distinguish nonisomorphic graphs that are locally equivalent. We then realize a toxicity screening experiment, consisting of a binary classification protocol on a biochemistry data set comprising 286 molecules of sizes ranging from 2 to 32 nodes, and obtain results which are comparable to the implementation of the best classical kernels on the same data set. Using techniques to compare the geometry of the feature spaces associated with kernel methods, we then show evidence that the quantum feature map perceives data in an original way, which is hard to replicate using classical kernels.

DOI: [10.1103/PhysRevA.107.042615](https://doi.org/10.1103/PhysRevA.107.042615)

I. INTRODUCTION

Representing data in the form of graphs is ubiquitous in many domains of sciences. They naturally describe relationships in social networks [1], characterize interactions of proteins and genes [2], and can represent the structure of sentences in linguistics [3]. Many impactful applications arise from efficient graph-based methods, such as predicting potential edges in recommendation systems [4], detecting frauds in communication networks [5], or for protein function prediction [6].

While graphs offer a rich structure for manipulating complex data, the level of freedom they afford can lead to resource-consuming data analyses. It is therefore essential to create efficient machine learning (ML) models that correctly and effectively learn and extract information from graph structures. One thus often resorts to graph embedding techniques [7], which refer to finding a representation of a graph or of its individual nodes in a vector space. By finding node representatives which preserve different types of relational information from the graph, node embedding can be used for prediction tasks at the node level, such as node classification [8] or link prediction [9]. Embeddings can also be done at the graph level to distinguish graphs of different nature. Notions of distances and similarities between the representative vectors can then be used to find the best boundary between data points with

different labels in the context of supervised machine learning. This is the main idea behind the notion of a graph kernel, which represents a measure of similarity between input graphs in the form of a scalar product between their representative vectors.

Using the exponentially large Hilbert space accessible to a quantum computer in order to generate graph embeddings is an appealing idea, with many proposals and theoretical studies over the past few years [10–13]. With the recent advances in geometric quantum machine learning, works have shown how graph-structured data could be encoded into quantum states and manipulated for classification, clustering, or regression tasks. These efforts started with quantum convolutional neural networks [14,15], and attempts were made to translate classical graph neural network (GNN) architectures to quantum neural networks [16]. Some of the authors of the present work introduced the quantum evolution kernel (QEK) approach in [17], which is based on evolving a quantum register over alternating layers of (graph-encoding) Hamiltonians and training for classification tasks. Follow-up work from the community offered generalizations of this paradigm [18]. Theoretical studies of geometrical quantum machine learning and their invariant properties include [19,20], the latter studying applications to weighted graphs. More recently, in-depth theoretical studies of equivariant and geometric quantum machine learning aspects were presented [21–23]. Here we specifically focus on a QEK-type quantum feature map [17] for graph-structured data that we experimentally investigate for various learning tasks on a 32-qubit neutral atom quantum processing unit (QPU).

*These authors contributed equally to this work.

†loic@pasqal.com

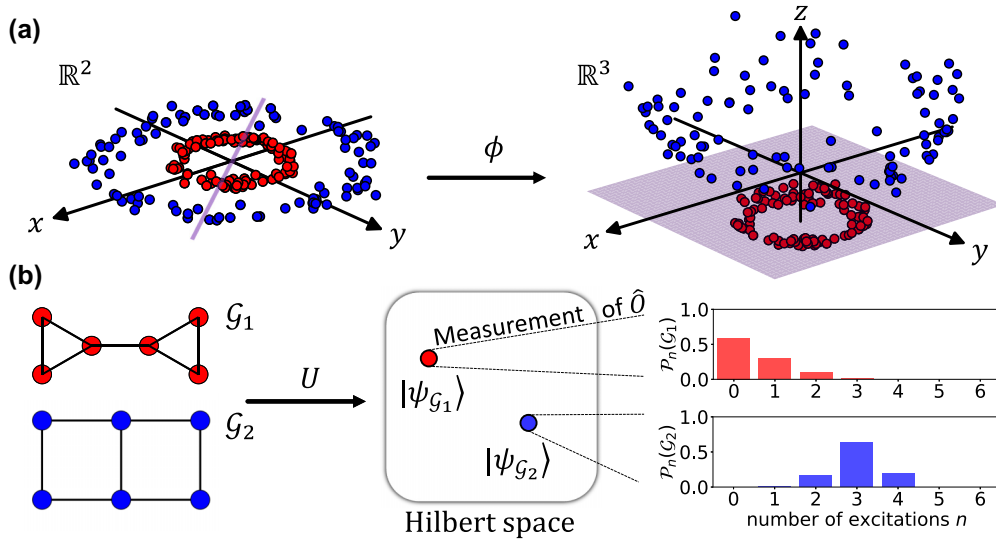


FIG. 1. (a) We seek a binary classifier enabling the separation of the two data classes by a hyperplane (purple). After a transformation $\phi(x, y) = (x, y, x^2 + y^2)$ mapping the data point from \mathbb{R}^2 to \mathbb{R}^3 , the transformed data points become linearly separable. (b) Illustration of the hardware-implemented quantum feature map $U(\mathcal{G}; t)$ and subsequent measurements for graph-structured data. A parametrized quantum unitary U is applied to atomic registers arranged under the form of UD graphs. Experimentally measured observable distributions are then used for learning tasks.

The structure of the paper is as follows. We first introduce the concept of a graph quantum feature map using neutral atom technology in Sec. II. We then assess in Sec. III its expressive power by showing that it enables us to distinguish two graphs that are locally equivalent but nonisomorphic [24]. In Sec. IV we experimentally realize QEK on a real-world classification task to predict toxicity for a data set of molecules [predictive toxicity challenge on female mice (PTCFM)] [25]. Importantly, we benchmark our approach by comparing its performance with several classical kernels in Sec. IV C. Finally, we evaluate the potential advantage of our method in Sec. IV D by means of a novel metric that is sensitive to the similarity between the geometry of the feature spaces of two kernels [26]. This is a strong indication of the potential of the method and of its capacity to capture new features that classical kernels would miss.

II. QUANTUM FEATURE MAP FOR GRAPH-STRUCTURED DATA

In many classical machine learning methods, one seeks to map input data into a different space called the feature space using a transform called the feature map, making it easier to work with. An example is shown in Fig. 1(a), where we illustrate how to easily solve a binary classification task for data points in a two-dimensional plane by embedding them into three-dimensional space. After a transformation ϕ taking data points from \mathbb{R}^2 to \mathbb{R}^3 , the resulting vectors can be easily separated by an horizontal plane. The class of any new data point can then be directly deduced from its location relative to this plane. In quantum machine learning [10,11], the embedding is usually done on a quantum feature space which is a Hilbert space associated with a set of qubits. Such an embedding is usually built from the dynamics of a

quantum system depending on the input data as well as external variational parameters.

In this paper we use a neutral atom-based QPU made of single ^{87}Rb atoms trapped in arrays of optical tweezers [27–31]. The qubits are encoded in the ground state $|0\rangle = |5S_{1/2}, F=2, m_F=2\rangle$ and a Rydberg state $|1\rangle = |60S_{1/2}, m_J=1/2\rangle$. This effective two-level system is addressed with a two-photon laser excitation through an intermediate state $6P_{3/2}$. The first (second) photon excitation is generated by a 420-nm (1013-nm) laser beam. Both of them are far detuned from their addressed transitions so as to ensure a negligible effect of the intermediate state. At the end of the laser sequence, the state of the atomic qubits is read out by fluorescence imaging (see Appendix A).

When promoted to Rydberg states, the atoms behave as large electric dipoles and thus experience dipole-dipole interactions, which, for the chosen Rydberg level, essentially include van der Waals terms only. The dynamics of a set of N qubits at positions $\{\mathbf{r}_i\}_{i=1,\dots,N}$ is thus governed by the Hamiltonian

$$\hat{H} = \hbar \sum_{i=1}^N \left(\frac{\Omega}{2} \hat{\sigma}_i^x - \delta \hat{n}_i \right) + \sum_{i<j} \frac{C_6}{|\mathbf{r}_i - \mathbf{r}_j|^6} \hat{n}_i \hat{n}_j, \quad (1)$$

where $\hat{\sigma}_i^\alpha$ are Pauli matrices, $\hat{n}_i = (1 + \hat{\sigma}_i^z)/2$, $|\mathbf{r}_i - \mathbf{r}_j|$ is the distance between qubits i and j , and $C_6/h \simeq 138 \text{ GHz } \mu\text{m}^6$ for the Rydberg state considered [32,33]. Controlling both intensities and frequencies of each laser field, we can effectively drive the qubit register uniformly with time-dependent tunable Rabi frequency Ω and detuning δ .

Key to our study is the programmability of the qubit register's geometry. In neutral atom processors, one can modify the spatial arrangement of qubits [27,34] and reproduce the geometrical shape of various graphs with atoms in tweezers. We will restrict ourselves to a set of graphs called unit disk

(UD) graphs, for which two nodes in the plane are connected by an edge if the distance between them is smaller than a given threshold. Unit disk graphs are intimately related to Rydberg physics through the mechanism of Rydberg blockade [29,35,36], where an atom excited to a Rydberg state prevents other neighboring atoms from being excited within a certain blockade radius. For an atomic register reproducing a UD graph \mathcal{G} , the $1/r^6$ power law of the van der Waals interactions effectively restricts, in a good approximation, the summation in the third term of Eq. (1) to pairs of indices (i, j) sharing an edge in \mathcal{G} . The topology of the interaction term in Eq. (1) then becomes the one of the graph under consideration, giving rise to a graph-dependent Hamiltonian $\hat{H}_{\mathcal{G}}$. This property has notably been harnessed for solving combinatorial graph optimization problems [37–45].

Starting from a UD graph \mathcal{G} reproduced in the array of tweezers with qubits all starting in $|0\rangle$, we apply a parametrized laser pulse onto the atoms in order to generate a wave function $|\psi_{\mathcal{G}}\rangle$ of the form

$$|\psi_{\mathcal{G}}\rangle = U(\mathcal{G}; t)|0\rangle^{\otimes |\mathcal{G}|}, \quad (2)$$

where we define the time-evolution operator $U(\mathcal{G}; t) = \mathcal{T}\{\exp[-i/\hbar \int_{s=0}^t \hat{H}_{\mathcal{G}}(s) ds]\}$ to be our quantum feature map unitary for graph-structured data. Throughout this paper, we will restrict ourselves to laser pulses with constant detuning δ and Rabi frequency Ω , with an adjustable duration t . Depending on the task at hand, we consider various observables \hat{O} to evaluate on $|\psi_{\mathcal{G}}\rangle$. Measurements of a site-dependent (global) observable give rise to a probability distribution \mathcal{P} which is node (graph) specific and can be used for various machine learning tasks at the node (graph) level. In the following, we show theoretically and experimentally that the graph quantum feature map already shows interesting properties when associated with local or global observables built from single-body expectation values $\langle \hat{O}_{j=1, \dots, |\mathcal{G}|} \rangle$.

III. EXPRESSIVE POWER OF THE GRAPH QUANTUM FEATURE MAP

The graph quantum feature map already shows interesting properties when associated with single-body observables $\langle \hat{O}_{j=1, \dots, |\mathcal{G}|} \rangle$. The measured values are affected not only by local graph properties such as node degrees, but also by more global ones such as the presence of cycles. This enrichment provided by the quantum dynamics contrasts with the locality of node representations in many classical graph machine learning. This key feature comes from the fact that the quantum dynamics of a given spin model (e.g., an Ising model) will be significantly influenced, beyond short times (given by the Lieb-Robinson bound [46,47]), by the complete structure of the graph.

We illustrate experimentally this behavior for two graphs \mathcal{G}_1 and \mathcal{G}_2 that are nonisomorphic but locally identical. In these graphs, nodes can be separated into two equivalence classes according to their neighborhood: Border nodes B have one degree-3 neighbor and one degree-2 neighbor, while center nodes C have two degree-2 neighbors and one degree-3 neighbor [see Fig. 2(a)]. We will see that the presence of interactions will enable us to discriminate between \mathcal{G}_1 and \mathcal{G}_2

by comparing the dynamics of local observables on border and center nodes.

We first map the graphs in a tweezers array with a nearest-neighbor (NN) distance of $r_{\text{NN}} = 5.3 \mu\text{m}$ and apply a constant pulse with $\Omega/2\pi = 1.0 \text{ MHz}$ and $\delta/2\pi = 0.7 \text{ MHz}$. We then measure the local mean Rydberg excitation $\langle n_j \rangle_{j \in B, C}$ for varying pulse duration $t \in [0, 2.5] \mu\text{s}$. As illustrated in Fig. 2(b), a qualitative difference in the dynamics of both graph appears after $t \sim 0.25 \mu\text{s}$. Precisely, the excitation of the border nodes [see Fig. 2(bi)] is initially increasing with indistinguishable behavior between the two graphs. Then a distinction appears between the two graph instances. The mean density for the border qubits of \mathcal{G}_1 exhibits damped oscillations around $\langle n_B \rangle \sim 0.15$ with period of the order of $0.5 \mu\text{s}$, while for \mathcal{G}_2 it exhibits flatter oscillations centered around 0.25 with period around $1 \mu\text{s}$. We can observe a comparable distinction between the two graphs for the center qubits [see Fig. 2(bii)]. The experimental measurements are consistent with the theoretical predictions and with the expected level of noise (see Appendix B for more details).

When restricted to the mean-field approximation (or similarly in the classical limit), the qubits' dynamics on either graph are far more similar, as illustrated in the insets of Fig. 2(b). We still observe distinct dynamics between the two graphs, which is due to next-nearest-neighbor (NNN) interactions (more pronounced for the center nodes). If we neglect those NNN interactions, the mean-field equations governing the dynamics of each qubit would only depend on its direct neighborhood, i.e., the local structure of the graph. In that case, the qubits dynamics for \mathcal{G}_1 and \mathcal{G}_2 obey the exact same equations [see the black dashed line in the insets of Fig. 2(b)]. We therefore conclude that the presence of interactions in the system enables us to discriminate between the two non-isomorphic graphs \mathcal{G}_1 and \mathcal{G}_2 by evaluating node-level local observables $\langle n_B \rangle$ or $\langle n_C \rangle$.

By looking at $\hat{O} = \sum_{i=1}^6 \hat{n}_i$, we can more quantitatively quantify the difference in the dynamics between the two graphs. To this end, we first compute the histogram \mathcal{P}_i of the number of excitations observed in each shot on graph \mathcal{G}_i . The difference between those graphs is then estimated via the Jensen-Shannon divergence D_{JS} of their respective histograms [48], a commonly used distance measure between probability distributions, which is defined as

$$D_{\text{JS}}(\mathcal{P}_1, \mathcal{P}_2) = H\left(\frac{\mathcal{P}_1 + \mathcal{P}_2}{2}\right) - \frac{H(\mathcal{P}_1) + H(\mathcal{P}_2)}{2}. \quad (3)$$

Here $H(\mathcal{P}) = -\sum_k p_k \ln p_k$ is the Shannon entropy of $\mathcal{P} = (p_1, \dots, p_{|\mathcal{G}|})$. The Jensen-Shannon divergence is minimal $D_{\text{JS}}(\mathcal{P}, \mathcal{P}) = 0$ if $\mathcal{P}_1 = \mathcal{P}_2 = \mathcal{P}$ and it is maximal if \mathcal{P}_1 and \mathcal{P}_2 have disjoint supports, with $D_{\text{JS}}(\mathcal{P}_1, \mathcal{P}_2) = \ln 2$. This is illustrated in Fig. 2(c), where the largest difference $D_{\text{JS}_{\text{max}}} \approx 0.28$ is achieved (roughly 40% of the maximal value) at a time $t \sim 0.57 \mu\text{s}$. At this duration, the distribution for \mathcal{G}_1 is sharply peaked at $n = 0$, while that of \mathcal{G}_2 is wider and peaks around $n = 2$, as illustrated in the inset of Fig. 2(c). We note that the local observables $\langle n_j \rangle_{j \in B, C}$ exhibit maximal deviation at this same duration t , indicating direct correspondence between measurements at the node and graph levels.

The dependence of local observables evaluated after the application of the quantum feature map on global graph

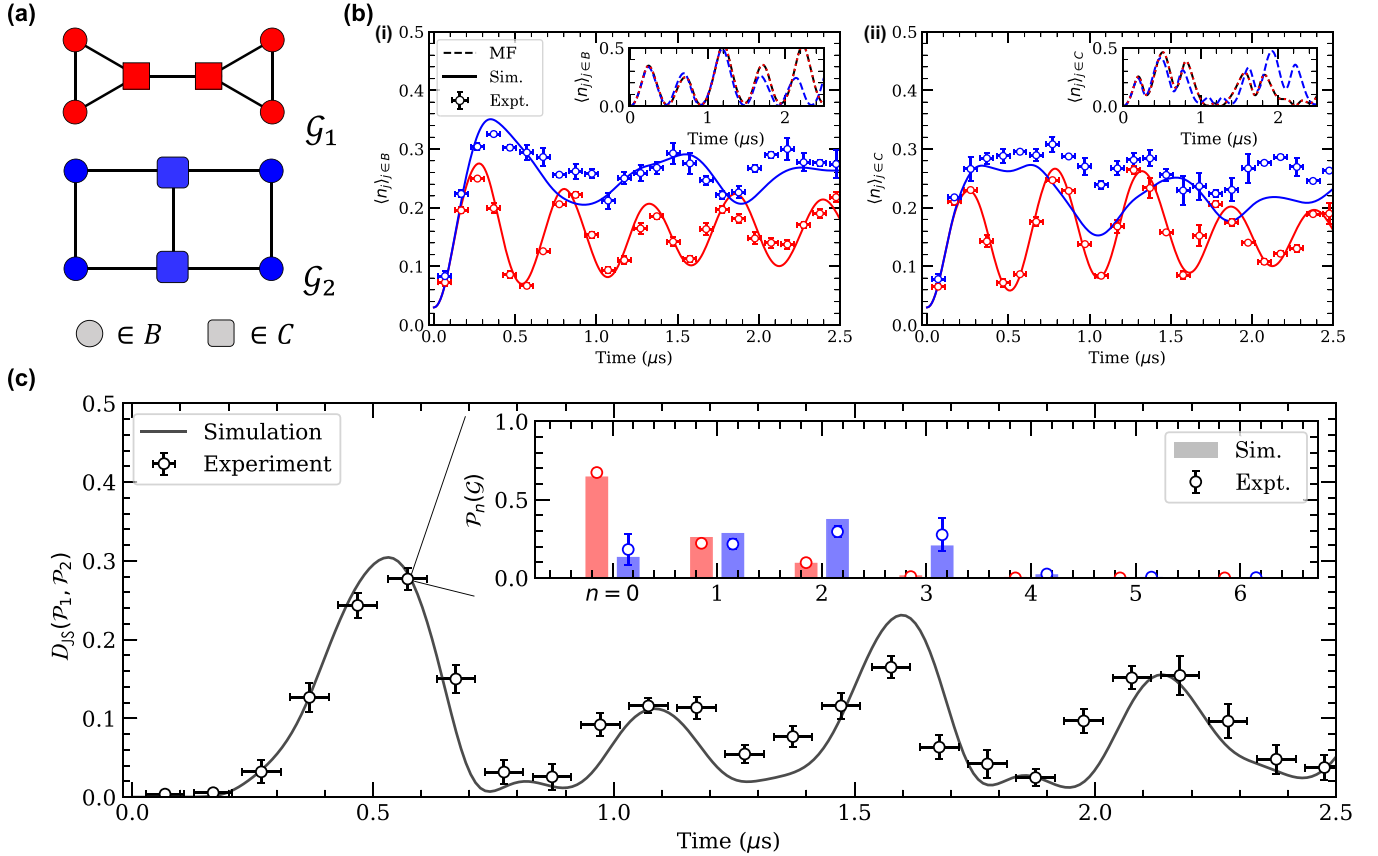


FIG. 2. (a) Here \mathcal{G}_1 (red) and \mathcal{G}_2 (blue) are two different graphs with identical local structures. Based on their neighborhood, the nodes belong to either the border B (circle) or the center C (square). (b) Evolution of the mean occupation $\langle n_i \rangle$ of the two regions (i) B and (ii) C for both graphs \mathcal{G}_1 (red) and \mathcal{G}_2 (blue). The symbols represent the experimental results while the solid curves show noisy simulation results. Horizontal error bars account for the sequence-trigger uncertainty (approximately equal to 40 ns), while the vertical ones account for the sampling noise. The insets show the corresponding mean-field dynamics (dashed lines) with only NN (black) or full (colored) interactions. (c) Evolution of the Jensen-Shannon divergence obtained experimentally (symbols) compared to the noisy simulation (solid line). At each point in time, $D_{\text{JS}}(\mathcal{P}_1, \mathcal{P}_2)$ is computed using the excitation distributions $\mathcal{P}_{1/2} = \{\mathcal{P}_n(\mathcal{G}_{1/2})\}_{n=0,\dots,6}$ obtained either numerically (bars) or experimentally (circles). The inset depicts $\mathcal{P}_{1/2}$ obtained at $t \approx 0.57 \mu\text{s}$, which yields the maximum value $D_{\text{JS}_{\text{max}}} \approx 0.28$ reached.

structures has interesting consequences regarding quantum-enhanced versions of GNNs [18,49,50]. In standard GNN architectures, information is only propagated along the graphs edges (see Appendix C for details). Incorporating a propagation rule built from the quantum feature map above would enable us to go beyond this well-known limitation of GNN architectures.

IV. BINARY CLASSIFICATION TASK

We now use the graph distance metric introduced in Eq. (3) to tackle a binary classification task on a data set of chemical compounds called PTCFM [25,51]. The objective is to accurately predict the reactivity of chemical compounds (toxic and positive or nontoxic and negative) based on their structural properties. Indeed, in many cases poisonous proteins act as enzyme inhibitors, where the geometry of the protein fits to the binding site of an enzyme and perturbs its usual functioning [52,53]. We will estimate the quality of the classification by using the F1 score $F_1 = \frac{t_p}{t_p + (f_p + f_n)/2}$. Here t_p , f_p , and f_n are the numbers of true positives, false positives, and false negatives of the predicted distribution, respectively. Correctly predicting

the toxicity of a compound (increasing t_p) leads to a better F_1 score. On the other hand, classifying a toxic compound as harmless (increasing f_n) or a harmless compound as toxic (increasing f_p) results in a lower score.

To realize the classification task on this data set, we need to turn the quantum graph embedding introduced in Sec. II into a kernel K , the quantum evolution kernel. We follow the approach originally proposed in Ref. [17] to define this measure of similarity. We compute the distributions \mathcal{P} of the total number of Rydberg excitations $\sum_j \hat{n}_j$ measured in the final state on graph \mathcal{G} and we use again the Jensen-Shannon divergence from Eq. (3) to build a kernel out of those distributions:

$$K(\mathcal{G}, \mathcal{G}') = \exp[-D_{\text{JS}}(\mathcal{P}, \mathcal{P}')]. \quad (4)$$

This kernel is well defined, i.e., the kernel matrix is always positive definite [48].

We feed this kernel to a support vector machine (SVM) algorithm in order to discriminate between graphs of the data set pertaining to one or the other class. A SVM is a supervised machine learning algorithm which, given the kernel K , aims at

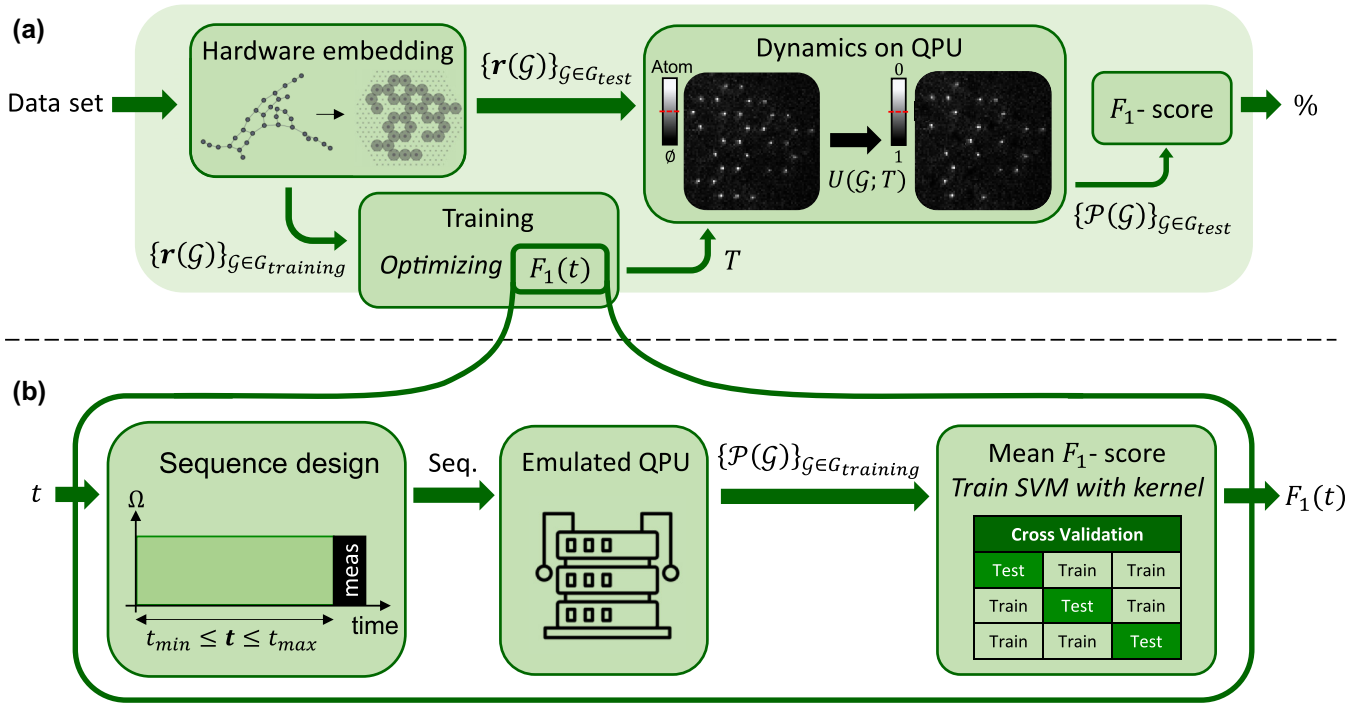


FIG. 3. (a) Data set of graphs \mathcal{G} first mapped onto atomic registers $\mathbf{r}(\mathcal{G})$ implementable on the QPU and separated between a training set $\mathcal{G}_{\text{training}}$ and a test set $\mathcal{G}_{\text{test}}$. We use the training set to determine numerically the optimal pulse sequence to be applied on the hardware using a grid search algorithm for optimizing $F_1(t)$ [see (b)]. This training phase outputs the optimal parameter T used to design the laser-pulse sequence applied experimentally on each register of the test set. The resulting dynamics performed on the QPU generates $U(\mathcal{G}; T)$, driving the system from $|0\rangle^{\otimes \mathcal{G}}$ to $|\psi_{\mathcal{G}}\rangle$. Then F_1 is derived from the measured probability distributions $\{\mathcal{P}(\mathcal{G})\}_{\mathcal{G} \in \mathcal{G}_{\text{test}}}$. (b) Optimization of the score function F_1 during the training includes several steps. The input t , taken from the parameter space $[t_{\text{min}}, t_{\text{max}}]$ defines a laser sequence with Ω and δ fixed parameters followed by a measurement. The dynamics of the system is emulated and enables us to compute the probability distributions associated with this particular value of t for the whole training part of the data set. Finally, $F_1(t)$ is obtained by fitting the SVM with the kernel constructed from those probability distributions.

finding a hyperplane in feature space that distinctly classifies the data points (see Appendix D for details).

A. Data set and mapping on hardware

In the original PTCFM data set, the 349 molecules are represented under the form of graphs where each node is labeled by atomic type and each edge is labeled according to its bond type. We first truncate the data set to small graph sizes in order to be able to train the kernel in a reasonable time and discard larger molecules. For the $M = 286$ remaining graphs of this data set, we take into account the adjacency matrix of the graphs representing the compounds and discard the nodes and edges labels. Note that the results of our implementation are therefore not directly comparable to kernel results in the literature which take into account edge and node labels (see, for example, Ref. [54]).

Each node of a graph will be represented by a qubit in the QPU. We first need to determine the positions of these qubits in order to implement an interaction term in Eq. (1) that effectively reflects the graph topology. To this end we design a local optimizer detailed in Appendix E to estimate in free space a preliminary two-dimensional (2D) layout for each graph. Starting from a Fruchterman-Reingold layout [55], our optimizer minimizes the average distance between two connected nodes while maximizing the distances between

unconnected nodes. Then taking advantage of our ability to tailor the spatial disposition of the tweezers generated by a spatial light modulator (SLM) to fit the optimized layout, we can replicate the graph in the hardware. Following a batching method also detailed in Appendix E, we group similar graphs and superimpose them on the same SLM pattern, effectively mapping the whole data set on only six different SLM patterns over a triangular grid. We therefore reduce the time needed to implement the whole data set on the QPU.

B. Model training

To test the performance of our implementation, we perform a standard procedure called cross validation. Cross validation consists of dividing the data set into five equal parts called splits and using each split for testing while the rest of the data set is used for training. During the training phase, we construct for each pulse duration t the corresponding kernel and train a SVM model with it. We then evaluate the F_1 score on the part of the data set that was left as a test set. We repeat the splitting ten times, and the cross-validation score is defined as the average of the F_1 scores of each split (50 splits in total). We perform a grid search on the penalty hyperparameter C of the SVM on the range $[10^{-3}, 10^3]$ such that the final score of a given pulse is the best cross-validation score among the tested values of C (see Appendix D for details).

TABLE I. The F_1 score reached experimentally on the PTCFM data set by QEK (plus or minus the standard deviation on the splits) and the scores reached numerically by the SVM ϑ , size, graphlet sampling, random walk, and shortest-path classical kernels. The values reported are the average over a fivefold cross validation repeated ten times.

Kernel	F_1 score (%)
QEK	60.4 ± 5.1
QEK (size compensated)	45.1 ± 3.7
SVM ϑ	58.2 ± 5.5
size	56.7 ± 5.6
graphlet sampling	56.9 ± 5.0
random walk	55.1 ± 6.9
shortest path	49.8 ± 6.0

Including graphs with sizes $|\mathcal{G}| \leq 20$, we numerically compute the score for a nearest-neighbor distance of $r_{\text{NN}} = 5.6 \mu\text{m}$ and a resonant constant pulse with fixed $\Omega/2\pi = 1 \text{ MHz}$ and we vary its duration between $t_{\text{min}} = 0.1 \mu\text{s}$ and $t_{\text{max}} = 2.5 \mu\text{s}$. We select the optimal duration $T = 0.66 \mu\text{s}$ that exhibits the maximum F_1 score. We then implement this pulse on the QPU. The whole process is illustrated in Fig. 3.

C. Classification results

After a training of our model, we experimentally obtain an F_1 score of $60.4 \pm 5.1\%$. For comparison purposes, we examine the performances of other kernels on this data set: the graphlet sampling (GS), random walk (RW), shortest path (SP), and SVM ϑ kernels, all these kernels being described in detail in Appendix F. The F_1 scores reached by the various kernels are collected in Table I. The obtained scores range from $49.8 \pm 6.0\%$ up to $58.2 \pm 5.5\%$. These results show that the quantum evolution kernel is competitive with standard classical kernels on this data set. The SVM ϑ kernel is found to be, among the classical kernels tested, the one with the best performance. As described in Appendix F 1, it is defined up to a choice of base kernel between real numbers, which gives it a certain degree of flexibility.

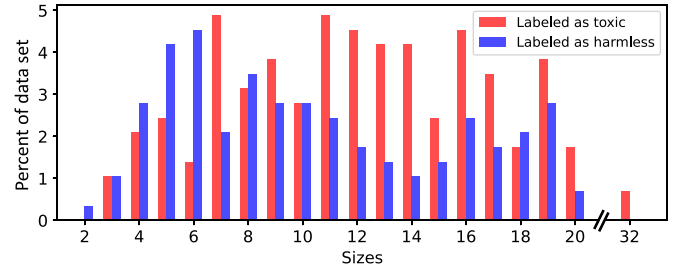


FIG. 5. The PTCFM data set exhibits a strong size imbalance. For a small number of nodes (less than approximately ten) more graphs are labeled as harmless (blue), while it is the opposite for larger graphs, more prone to be labeled as toxic (red).

We show in Fig. 4(a) the kernel matrix associated with QEK, with indices sorted by increasing size of the graphs. Using the same noise model as in the preceding section, we find adequate agreement between the numerically \mathcal{P}^{num} and experimentally \mathcal{P}^{exp} obtained data. Quantitatively, we make use of the Jensen-Shannon divergence to estimate this agreement for any \mathcal{G}_i and observe that $\langle D_{\text{JS}}(\mathcal{P}_i^{\text{num}}, \mathcal{P}_i^{\text{exp}}) \rangle_i \approx 0.03 \pm 0.01$ is one order of magnitude below $\langle D_{\text{JS}}(\mathcal{P}_i^{\text{exp}}, \mathcal{P}_j^{\text{exp}}) \rangle_{i \neq j} \approx 0.33 \pm 0.01$. An interesting feature of both QEK and SVM ϑ [Fig. 4(b)] kernel matrices is the emergence of size-related diagonal blocks, signaling that the models identify the size of the graphs as an important feature for classification. Examining more closely the data set, we indeed remark that the subset of PTCFM that we used is significantly size imbalanced, as illustrated in Fig. 5. Since the graph size seems to be a relevant feature for this particular data set, we define a size kernel as a Gaussian in the size difference, which reaches an F_1 score of $56.7 \pm 5.6\%$. The corresponding kernel matrix is displayed in Fig. 4(c) and exhibits a block-diagonal shape with a Gaussian tail.

It is interesting to note that the quantum model was able to identify size as a relevant parameter for this data set, leading to classification results which are on par with the best classical kernels.

Going forward, we modify the QEK procedure in order to make the kernel insensitive to size. To that end, we compare the measurement distributions obtained for different graph

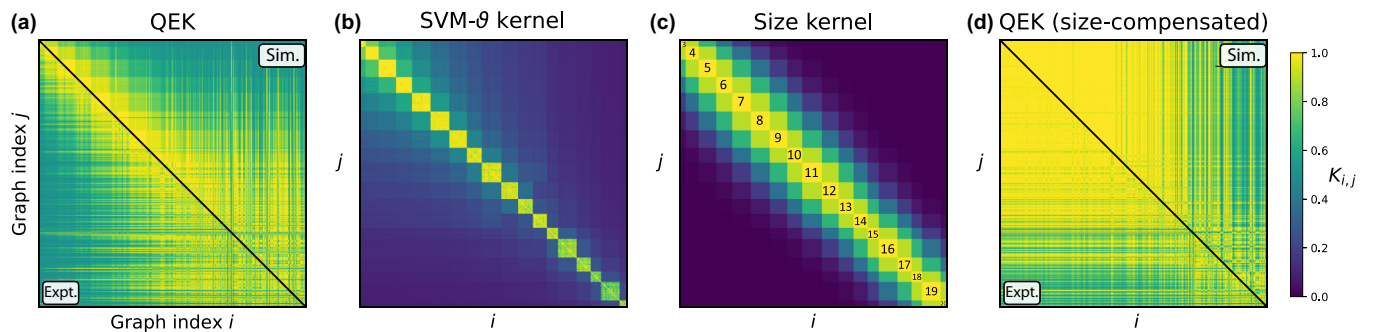


FIG. 4. Each kernel is represented by an $M \times M$ matrix where $K_{i,j} = K(\mathcal{G}_i, \mathcal{G}_j)$ as defined in Eq. (4). The graph indices are sorted by increasing size. A separation (black line) is drawn between numerically simulated (top right) and experimentally measured (bottom left) QEK matrices. (a) QEK kernel obtained using directly the raw distributions \mathcal{P}_i and \mathcal{P}_j . (b) Kernel obtained via the SVM ϑ method. (c) Size kernel obtained for $K^{\text{size}}(\mathcal{G}_i, \mathcal{G}_j) = \exp[-\gamma(|\mathcal{G}_i| - |\mathcal{G}_j|)^2]$ with $\gamma = 0.1$. (d) QEK kernel obtained using modified distributions $\tilde{\mathcal{P}}_i$ and \mathcal{P}_j , where graphs of smaller sizes are convoluted with binomial distributions when compared to larger graphs.

sizes using a convolution operation. Let us consider two graphs \mathcal{G}_i and \mathcal{G}_j of N_i and $N_j = N_i + \Delta N > N_i$ nodes, respectively, and note their respective observable distributions \mathcal{P}_i and \mathcal{P}_j . From \mathcal{P}_i we construct $\tilde{\mathcal{P}}_i = \mathcal{P}_i \star b_{\Delta N}^{(i,j)}$, the convolution of \mathcal{P}_i and a binomial distribution

$$b_{\Delta N}^{(p)}(n) = \binom{\Delta N}{n} p^n (1-p)^{\Delta N-n}. \quad (5)$$

Here $\tilde{\mathcal{P}}_i$ corresponds to the distribution one would get by adding to the graph ΔN noninteracting qubits, submitted to the same laser pulse as the other. Each of these isolated qubits undergoes Rabi oscillations, induced by the applied pulse sequence. They are therefore measured either in $|0\rangle$ with probability p or in $|1\rangle$ with probability $1-p$, where $p = \sin^2(\pi\Omega T)$ (approximately equal to 0.768 here). We finally define the modified graph kernel as

$$K_{\text{conv}}(\mathcal{G}_i, \mathcal{G}_j) = \exp[-D_{\text{JS}}(\tilde{\mathcal{P}}_i, \mathcal{P}_j)]. \quad (6)$$

Using this procedure on the data obtained experimentally, we obtain the kernel matrix shown in Fig. 4(d), with a corresponding F_1 score of $45.1 \pm 3.7\%$. If this size-compensated version of QEK had been implemented without interaction between atoms, its score would be 42%, which is the lowest score reachable by any kernel. We therefore see that this version of QEK cannot capture useful features beyond the graph size, meaning that the presence of interactions by itself is not sufficient to produce an interesting kernel for the task at hand. While the size-compensated QEK does not give results that are comparable to classical kernels, we study in the following section its expressive power and show that the geometry induced by this method is hardly reproducible by a classical kernel.

D. Geometric test with respect to classical kernels

In order to obtain an advantage over classical approaches it is not sufficient to implement a quantum feature map based on quantum dynamics that are hard to simulate classically. As shown in [26], classical ML algorithms can in certain instances learn efficiently from intractable quantum evolutions if they are allowed to be trained on data. The authors consequently proposed another metric between kernels in the form of an asymmetric metric function called the geometric difference g_{12} . It compares two kernels K_1 and K_2 as

$$g_{12} = \sqrt{\|\sqrt{K_2}(K_1)^{-1}\sqrt{K_2}\|_{\infty}}, \quad (7)$$

where $\|\cdot\|_{\infty}$ is the spectral norm. Intuitively, g_{12} measures the difference between how kernels K_1 and K_2 perceive the relation between data. Precisely, it characterizes the disparity regarding how each of them maps data points to their respective feature spaces. In our case, we take K_1 to be the size-compensated QEK K_{conv} and K_2 is selected from a set of classical kernels. If the geometric difference is small, it means that there exists no underlying function mapping the data to the targets for which K_{conv} outperforms the classical kernel. On the other hand, a large geometric difference between a quantum and a classical kernel guarantees that there exists such a function for which the quantum model outperforms the classical one. Estimating the geometric difference is therefore

TABLE II. Order of magnitude of the geometric difference between QEK and various classical kernels.

Kernel	Order of magnitude
SVM ϑ	10^3
size	10^5
graphlet sampling	10^4
random walk	10^5
shortest path	10^5

a sanity check to stating that the encoding of data to the feature space through the quantum evolution kernel could not be closely replicated by a classical model.

We compute the geometric difference between QEK and various classical kernels over the PTCFM data set and report the results in Table II. The threshold for a large geometric difference is typically taken to be \sqrt{M} , where M is the size of the data set. Here the obtained g_{12} is always far beyond $\sqrt{M} \sim 10^1$, indicating that the embedding of data through our quantum-enhanced kernel is not trivial and cannot be replicated by a classical machine learning algorithm.

To summarize, while the F_1 score on PTCFM is rather similar using quantum or classical models, we see nonetheless that the geometry created by our quantum model is nontrivial. A possible interpretation of the nonsuperiority of quantum approaches on PTCFM would be that the relationship between the data and the targets is not better captured by our quantum model, although its feature space is not reproducible by classical means. To further confirm this understanding, we find a function that increases and even maximizes the utility of our rich quantum feature space. We build such a function by artificially relabeling the targets according to a procedure presented in [26] and outlined in Appendix G. We observe that QEK, without retraining, retains an F_1 score of around 99% on the relabeled data set, while the closest classical kernel reaches a score of at most 82% even after retraining it on the new labels. The results are summarized in Table III, where the difference in F_1 score between QEK and the various classical kernels is shown.

In light of the geometric difference assessment and the observed gap of F_1 score between QEK and classical kernels on an artificial function, it remains an open question to generally characterize which types of data set naturally offer a structure that better exploits the geometry offered by our quantum model, without requiring artificial tweaking of the labels. In the following section, we present a synthetic data

TABLE III. Gap in F_1 score between QEK and various classical kernels after relabeling the data set.

Kernel	Gap
SVM ϑ	17.2 ± 4.5
size	17.8 ± 4.2
graphlet sampling	20.1 ± 4.5
random walk	17.3 ± 4.3
shortest path	18.2 ± 4.4

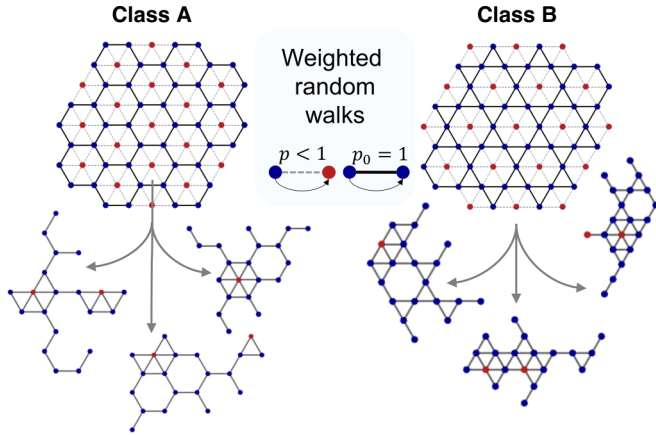


FIG. 6. Graphs in class A contain honeycomb sites (blue) with inclusions of nonhoneycomb sites (red) with probability p . Graphs in Class B contain kagome sites (blue) with inclusions of nonkagome sites (red) with probability p . We show examples of graphs generated with the aforementioned process.

set on which QEK is able to outperform classical methods without any relabeling.

E. Synthetic data set

This binary classification data set is created by sampling weighted random walks on a triangular lattice. In class A, sites belonging to a honeycomb-type sublattice are favored. They are explored with a weight $p_0 = 1$, while the rest of the triangular lattice sites are explored with a weight $p < 1$. Class B is constructed in a similar fashion, but taking a kagome instead of a honeycomb sublattice. The construction of this artificial data set is illustrated in Fig. 6. In the case where $p = 0$, the differences in their local structure make the two classes easily distinguishable. However, with increasing p , their local structure becomes more and more similar, as additional triangular lattice sites are incorporated. When p is large enough, many triangular local substructures are shared by the two classes, rendering them potentially hard to distinguish by classical methods. At $p = 1$, the underlying triangular lattice is explored uniformly, rendering the data sets indistinguishable.

Building on our ability to distinguish between graphs with similar local structure but globally distinct, we apply QEK on this synthetic data set. We expect our method to be hardly affected by the presence of sparse defects and therefore be able to outperform classical approaches. We investigate numerically this assumption, for several values of p . In each case, we create 200 graphs of 20 nodes each, 100 in each class. The graphs are mapped to a triangular lattice with 5- μm spacing. Here we consider two alternative schemes of pulse sequences. The first one remains almost the same as the experimentally implemented one, i.e., a unique resonant pulse of $\Omega/2\pi = 2$ MHz with parametrized duration up to 8 μs . The second one is an alternate layer scheme with four parameters as described in [17], where we evaluate 500 random values of the parameters and select the best one. The procedure is designed such that it would be directly implementable on the hardware, as we did for the PTCFM data set. We then compare the F_1 score reached by QEK to those

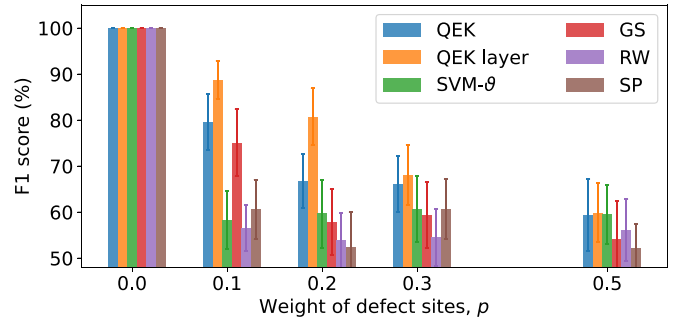


FIG. 7. The F_1 score (%) reached on the synthetic data set for different probabilities p of including nonsublattice sites, by the quantum evolution kernel (the alternate scheme is labeled the QEK layer) as well as by the best SVM ϑ , GS, RW, and SP kernels. The values reported are the average over a fivefold cross validation repeated ten times. Each kernel reaches an F_1 score of 100% when $p = 0$.

reached by other classical kernels, namely, SVM ϑ , GS, RW, and SP. The results are summarized in Fig. 7. With decreasing proportion of defects, all methods perform increasingly better, as expected. Overall, regarding the mean F_1 score reached, the two QEK schemes outperform the four other classical kernels tested for all $p \leq 0.5$. Noticeably, at $p = 0.1$ ($p = 0.2$), the mean gap in F_1 score between the QEK scheme and the best classical scheme is 4.5% (7.1%), while the mean gap obtained with the alternate QEK scheme is even larger with 13.7% (21%), thus showing that QEK can significantly surpass classical approaches on certain types of data sets. When adding too many defects, i.e., $p = 0.5$, our quantum evolution kernel exhibits similar performance to the SVM ϑ .

V. CONCLUSION

In this paper we reported the implementation of a quantum feature map for graph-structured data on a neutral atom quantum processor with up to 32 qubits. We experimentally showed that this embedding not only was sensitive to local graph properties but also was able to probe more global structures such as cycles. This property offers a promising way to expand the capabilities of standard GNN architectures, which have been shown to have the same expressiveness as the Weisfeiler-Lehman (WL) isomorphism test in terms of distinguishing nonisomorphic graphs [56,57]. For example, a standard GNN architecture will treat \mathcal{G}_1 and \mathcal{G}_2 shown in Fig. 2(a) in the same way, as they have the same local structure. Some properties of quantum-enhanced version of GNNs have been explored in [50] by some of the authors of this paper.

We then used the quantum graph feature map for a toxicity screening procedure on a standard biochemistry data set comprising 286 graphs of sizes ranging from 2 to 32 nodes. This procedure achieved an F_1 score of $60.4 \pm 5.1\%$, on par with the best classical kernels. We intentionally did not include GNNs in the benchmark, as they belong to another distinct family of models. Beyond this pure performance assessment, we showcased the potential advantage of using a quantum feature map through the computation of geometric differences with respect to said classical kernels, which are metrics evaluating the degree of similarity between the kernels' feature spaces. We showed that the quantum evolution

kernel captured features that are invisible to the classical kernels we considered. An artificial relabeling of the data enabled us to create a synthetic data set for which the performances of the quantum evolution kernel could not be matched. We also identified another data set made of bipartite 2D lattices, for which the quantum procedure exhibited superior performances.

This proof of concept illustrates the potential of quantum-enhanced methods for graph machine learning tasks. Our study paves the way for the incorporation of quantum-enhanced algorithms with standard ML solutions, aiming at constructing better tools for graph data analysis and prediction. Further work on more diverse data sets will be required to assess the viability of the approach compared to powerful state-of-the-art GNN architectures [58–61]. Additionally, our results showcase the power and versatility of neutral atom QPUs, with their ability to change the register geometry from run to run. Going forward, the implementation of similar methods on nonlocal graphs could be envisaged by embedding them into three-dimensional registers [45] or moving the qubits throughout the course of the computation [62].

ACKNOWLEDGMENTS

We thank Jacob Bamberger, Lucas Beguin, Antoine Browaeys, Luc Couturier, Romain Fouilland, Thierry Lahaye, Hsin-Yuan Huang, Christophe Jurczak, and Georges-Olivier Raymond for fruitful discussions.

APPENDIX A: EXPERIMENTAL SETUP

The experimental setup is shown schematically in Fig. 8. It features a magneto-optical trap (MOT), able to cool down and confine a cloud of ^{87}Rb atoms in order to load an array of optical tweezers. They are created by shining a 849-nm laser beam on a spatial light modulator (SLM) and then focusing the beam to a small waist of approximately $1\ \mu\text{m}$ with a high-numerical-aperture optical system inside the vacuum chamber. The loading of the optical tweezers is stochastic with a probability $\eta \approx 0.55$ of obtaining one atom per trap. Hence, at each repetition cycle of the experiment, we use a dynamical optical tweezer to move the atoms one by one in order to generate the targeted graph.

The atoms are embedded into a 10 G magnetic field that sets the quantization axis. The qubits are encoded into the ground state $|0\rangle = |5S_{1/2}, F = 2, m_F = 2\rangle$ and a Rydberg state $|1\rangle = |60S_{1/2}, m_J = 1/2\rangle$ of the atoms. They are initialized in the ground state by optical pumping. The qubit transition is then addressed by a two-photon laser excitation, via an intermediate state $6P_{3/2}$. The first (second) photon excitation is generated by a 420-nm (1013-nm) σ^+ -polarized (σ^- -polarized) laser beam with a $1/e^2$ waist radius of $260\ \mu\text{m}$ ($180\ \mu\text{m}$). The two lasers being far detuned from the intermediate state by 700 MHz, we avoid spurious populating of this state and the three-level system can be faithfully approximated by an effective two-level system. The qubits state is read out in a single step by fluorescence imaging close to resonance at 780 nm, using an electron multiplying charge-coupled device (EMCCD) camera with an integration time of 20 ms.

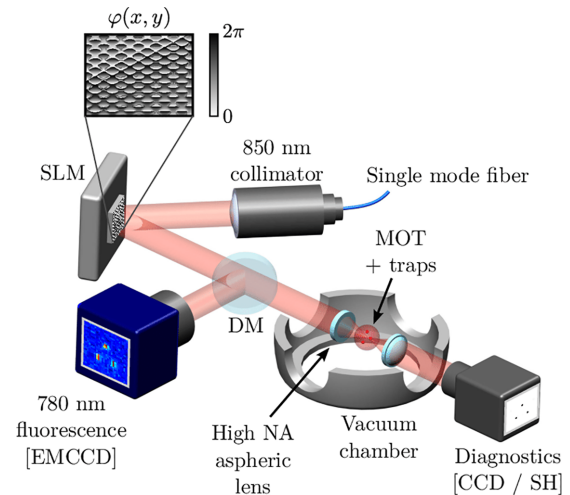


FIG. 8. Microtraps for capturing single atoms are generated using a SLM. A calculated phase pattern is printed on the 849-nm laser beam and then focused by the first of two high numerical aperture lenses on the middle of a MOT. The atomic fluorescence at 780 nm is reflected by a dichroic mirror (DM) and detected with an EMCCD camera. A second aspheric lens (identical to the first) collects the 849-nm light for three kind of images: layout loading (tweezers loading quality), register validation (rearrangement successfulness), and register readout (Rydberg excitation discrimination results). The transmitted beam is used for trap diagnostics via a CCD camera or a Shack-Hartmann wavefront sensor (SH). (Figure has been reproduced from [28].)

A set of eight electrodes in an octupole configuration provides active control of the electric-field environment around the Rydberg atoms. The durations and shapes of the Rydberg pulses are defined using acousto- and electro-optic modulators, in order to ensure the correct pulse length used on the measurements.

APPENDIX B: NOISE MODEL

Despite the precise calibration of the control devices which enable us to monitor quantities such as the SLM pattern spacing or the pulse shapes, several experimental imperfections may alter the data measured on the experiment. All experimental data obtained during this study, including those presented in Figs. 2 and 4, are uncorrected and thus need to be benchmarked with respect to their simulated counterpart, taking into account the following main sources of noise.

First and foremost, due to the nature of the quantum state and the limited budget of shots, measurements are subject to sampling noise. For instance, on average, each of the 25 experimental points in Fig. 2 is obtained using 600 shots and the uncertainty related to this effect (vertical error bars) can be estimated using the jackknife resampling method [63].

The finite sampling is also inherently flawed by several physical processes such as atom thermal motion, background-gas collisions, or Rydberg state finite lifetime, whose effects can all be encompassed as a first approximation into two detection error terms ε and ε' . The ε (ε') yield the probability to get a false positive (negative), i.e., measure an atom in $|0\rangle$ ($|1\rangle$) as being in $|0\rangle$ ($|1\rangle$). The ε can be measured with

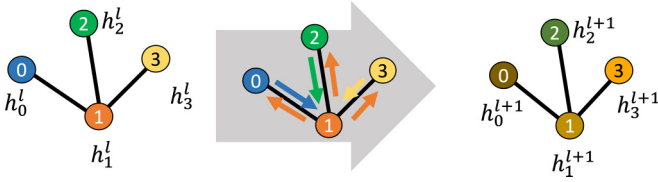


FIG. 9. In message passing neural networks, node feature vectors h_i^l are iteratively updated, from one layer l to the next $l+1$, using only neighboring nodes, similarly to what is done in the WL test.

a regular release-and-recapture experiment and ε' with a more advanced method [64] involving π and pushout pulses. To replicate the probabilistic effect of detection errors, the simulated distributions of bit strings are altered using the following rule to compute the probability of measuring j instead of i :

$$P_{j|i} = \prod_k (1 - |i - j|_k) - (-1)^{|i-j|_k} [(1 - i_k)\varepsilon + i_k\varepsilon']. \quad (\text{B1})$$

Here $i, j \in \mathbb{B}^N$ and $i_k = 0$ (1) if atom k is in $|0\rangle$ ($|1\rangle$). On our device, we measure $\varepsilon \approx 3\%$ and $\varepsilon' \approx 8\%$; thus, as an example, we can compute $P_{1001|0101} = \varepsilon\varepsilon'(1 - \varepsilon)(1 - \varepsilon') \approx 0.2\%$. Those detection errors can deeply modify the measured excitation distributions, with a noticeable effect shown in Fig. 2(b) at $t = 0$ where the simulated $\langle n_j \rangle$ does not start at 0 despite $|\psi(t=0)\rangle = |0 \dots 0\rangle$.

Additional errors can also lead to decoherence in the system [65], affecting the atom dynamics in ways costly to emulate. For instance, since the Rydberg transition used is addressed by a two-photon process, misalignments and power fluctuations of the two lasers are twice as likely to occur. Atoms are subject to positional disorder between each shot and their finite velocities make them sensitive to the Doppler effect. Since taking all those effects into consideration becomes quickly intractable, they were only individually simulated in order to assess their limited action on the implemented protocols. However, in order to replicate the experimental data presented in Fig. 2, we resort to an effective decoherence model in the form of solving the master equation with a relaxation rate of $2\pi \times 0.06$ MHz [32]. This value was obtained by fitting with the above model damped Rabi oscillations measured on the same device. Thus, reaching similar behavior within error bars between numerically simulated and experimentally obtained $D_{\text{JS}}(\mathcal{P}_1, \mathcal{P}_2)$ was achieved with no free parameter.

APPENDIX C: MESSAGE PASSING NEURAL NETWORKS

Message passing neural networks [58] are widely used families of graph neural networks. They were one of the first networks to be developed for graph-structured data and are still one of the most successful [66]. They consist of GNNs where the update is made only by aggregating the features of nearest neighbors. In this scheme, the nodes features are multiplied by a trainable weight matrix at each layer and each node aggregates as a message of the features of its neighbors, as illustrated in Fig. 9.

Message passing neural networks are closely related to Weisfeiler-Lehman algorithms. In particular, they have been proven to be at most as powerful in distinguishing graph

structures [57]. In their standard form, they are then also limited to capture only local features of graphs.

APPENDIX D: SUPPORT VECTOR MACHINE ALGORITHM

The SVM algorithm aims at splitting a data set into two classes by finding the best hyperplane that separates the data points in the feature space, in which the coordinates of each data point (here each graph) are determined according to the kernel K .

For a training graph data set $\{\mathcal{G}_i\}_{i=1, \dots, M}$ and a set of labels $\mathbf{y} = \{y_i\}_{i=1, \dots, M}$ (where $y_i = \pm 1$ depending on which class the graph \mathcal{G}_i belongs to), the dual formulation of the SVM problem consists in finding $\tilde{\alpha} \in \mathcal{A}_C(\mathbf{y}) = \{\alpha \in [0, C]^M \mid \alpha^T \mathbf{y} = 0\}$ such that

$$\frac{1}{2} \tilde{\alpha}^T Q \tilde{\alpha} - \mathbf{e}^T \tilde{\alpha} = \min_{\alpha \in \mathcal{A}_C(\mathbf{y})} \left\{ \frac{1}{2} \alpha^T Q \alpha - \mathbf{e}^T \alpha \right\}, \quad (\text{D1})$$

where \mathbf{e} is the vector of all ones, Q is a $M \times M$ matrix such that $Q_{ij} = y_i y_j K(\mathcal{G}_i, \mathcal{G}_j)$, and $C > 0$ is the penalty hyperparameter, to be adjusted. Setting C to a large value increases the range of possible values of α and therefore the flexibility of the model; however, it also increases the training time and the risk of overfitting.

The data points for which $\tilde{\alpha}_i > 0$ are called support vectors (SVs). Once the α_i are trained, the class of a new graph \mathcal{G} is predicted by the decision function, given by

$$y(\mathcal{G}) = \text{sgn}\{\langle \phi(\mathcal{G}) | \phi_0 \rangle\} \quad (\text{D2})$$

$$= \text{sgn} \left\{ \sum_{i \in \text{SV}} y_i \tilde{\alpha}_i K(\mathcal{G}, \mathcal{G}_i) \right\}, \quad (\text{D3})$$

with

$$\phi_0 = \sum_{i \in \text{SV}} y_i \tilde{\alpha}_i \phi(\mathcal{G}_i). \quad (\text{D4})$$

In this case, the training of the kernel amounts to finding the optimal feature vector ϕ_0 . It is worth noting that in many cases, Eq. (D3) is evaluated directly, without explicitly computing ϕ_0 .

APPENDIX E: MAPPING AND BATCHING

We present in detail our method to embed the graphs of the PTCFM data set. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a graph of the data set for which we have a layout of the nodes. Embedding the graph amounts to replacing its nodes with atoms, the latter interacting between themselves with the $1/R^6$ dependence. Moving two atoms slightly apart can therefore drastically reduce their interaction strength, but it remains nonzero. In order for the Hamiltonian to reflect the topology of \mathcal{G} , this $1/R^6$ dependence needs to be approximated by the Heaviside function defined as

$$h(r) = \begin{cases} \infty & \text{for } r \leq r_b \\ 0 & \text{otherwise.} \end{cases} \quad (\text{E1})$$

For the Heaviside approximation to be correct, we have to ensure that the largest distance between a pair sharing an edge in the graph is always far less than the shortest distance

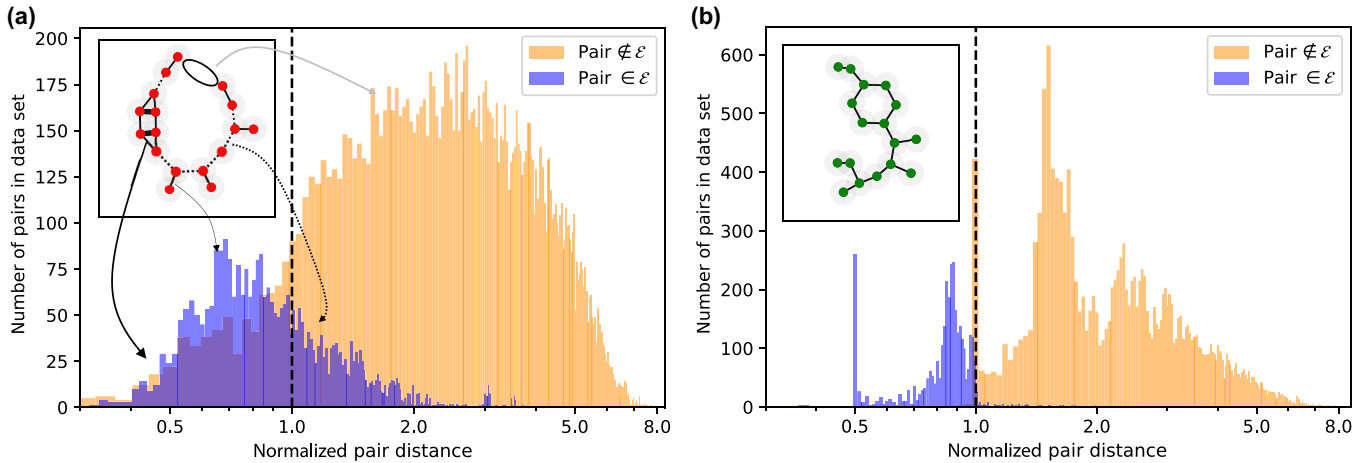


FIG. 10. Histograms of normalized pairwise distances between atoms in the 286 graphs of the truncated data set when performing the embedding (a) with only a Fruchterman-Reingold layout or (b) when adding a local optimization step afterward. For a given graph (inset), two atoms forming a pair in \mathcal{E} (blue) can be close enough to form a bond via interaction (solid line) or too far, creating a missing bond (dotted line). Likewise, two atoms forming a pair not in \mathcal{E} can be placed too close and form a fake edge (thick line).

between a pair not sharing an edge. In other words, in theory, $\min\{U_{ij}, (i, j) \in \mathcal{E}\} / \max\{U_{ij}, (i, j) \notin \mathcal{E}\} \gg 1$.

We use a local optimizer to maximize this ratio and find good solutions in polynomial time. The method optimizes the position of each node in turn, depending on the previously mapped nodes and the presence of cycles in the graph. For the data set used in this study, we achieve a significant increase of the mean ratio up to 16.8, starting from 5.9 with the classical Fruchterman-Reingold layout. We report that more than half the data set exhibits a ratio higher than 10 and less than 5% of the data set is embedded with some defects, i.e., a ratio smaller than 1. We also assess the benefit of this approach in Fig. 10 by comparing the distributions of distance of pairs in \mathcal{E} and pairs not in \mathcal{E} before [Fig. 10(a)] and after [Fig. 10(b)] the optimization. While some defects, such as fake or missing bonds, frequently appear in the preoptimization embedding, the optimized positions are constrained such that a clear-cut is visible between the two distributions, easing the approximation.

In order to further characterize the effect of these defects, we analyze their effect on the measurement histograms. For each graph, we first compute the histogram that would have been obtained with a perfect embedding. We then compute the Jensen-Shannon divergence between this histogram and the one measured in the QPU (see Fig. 11). From these, we can also estimate that if one were to perform the SVM using histograms resulting from ideal embedding, one would expect a slightly worst F1 score of 58.8 ± 4.0 .

In principle, we can program a different SLM pattern for the layout of each graph from the data set. In practice, however, the SLM calibration step can be quite time consuming, i.e., of the order of 1 min. We can compare it to the duration of hundreds of shot, each of which consist of applying a sequence and measuring a quantum state, performed at a frequency of 1 Hz. Then for each graph, calibrating the SLM and obtaining the probability distribution take approximately the same order of time.

We therefore seek to regroup many graphs onto the same SLM pattern, to be able to reduce the number of calibrations

needed for the whole data set. We do so by clustering the graphs according to similarities in their structures. Because the data set consists in representations of organic molecules, many of the graphs share common structures. We thus focus on retrieving the presence and multiplicity of pentagons and hexagons. We then build a similarity measure between the graphs. For the pentagons, for example, the similarity can be written in the form

$$s(\mathcal{G}_1, \mathcal{G}_2) = 1 - \exp(-\alpha |N_1^P - N_2^P|), \quad (\text{E2})$$

where N^P represents the number of pentagons in \mathcal{G} and α is a hyperparameter. We then use a linear combination of similarity measures in order to build a similarity matrix between all graphs of the data set. We then apply a k -means clustering algorithm [67] using the similarity matrix in order to separate the graphs into different batches. Furthermore, since the laser power is distributed over all the traps, we want to reduce the total number of traps in order to maximize the intensity provided to each trap. This ensures that the traps are deep

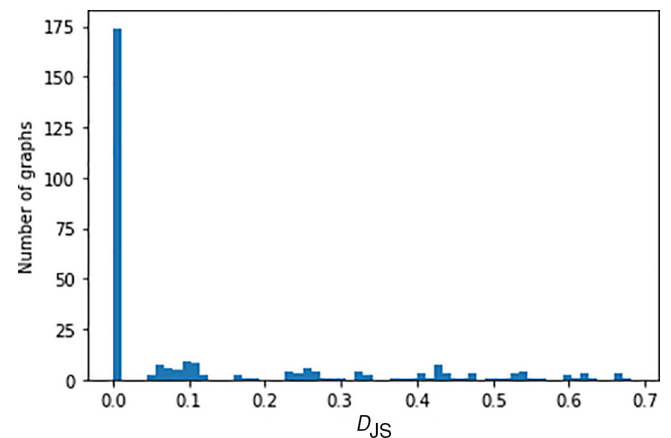


FIG. 11. Distribution of Jensen-Shannon divergences between measured and expected histograms for a pulse of duration of $T = 0.66 \mu\text{s}$. The implementation was done in a regime where this difference can be large for some graphs.

Algorithm 1. Creating a triangular SLM pattern by batching M graphs.

Require: Graphs $\{\mathcal{G}_1, \dots, \mathcal{G}_M\}$ in sorted sizes and optimized positions $\{x_1, \dots, x_M\}$

Ensure: Single SLM pattern that embeds M graphs with optimal positions on a triangular lattice

- 1: traps = $\{\}$
- 2: for i in range $1, \dots, M$:
- 3: find $r_{\mathcal{G}_i} = \{r_1, \dots, r_{|\mathcal{G}_i|}\}$ triangular grid points that best conserve the pairwise distances between points in x_i and maximizes overlap with existing traps.
- 4: traps \leftarrow traps + $r_{\mathcal{G}_i}$ traps
- 5: if $|\text{traps}| < 2|\mathcal{G}_M|$, add additional random triangular grid points to guarantee the filling property for rearrangement.

enough to obtain a satisfying filling efficiency (approximately 55%) over the whole pattern. For each batch, we thus apply the following mapping algorithm.

We successfully map the entire data set of 286 graphs into only six SLM patterns. For example, we batch 66 graphs together onto the 71-trap SLM pattern presented in Fig. 12. On average, the six SLM patterns use 70 traps each to encode 48 graphs each.

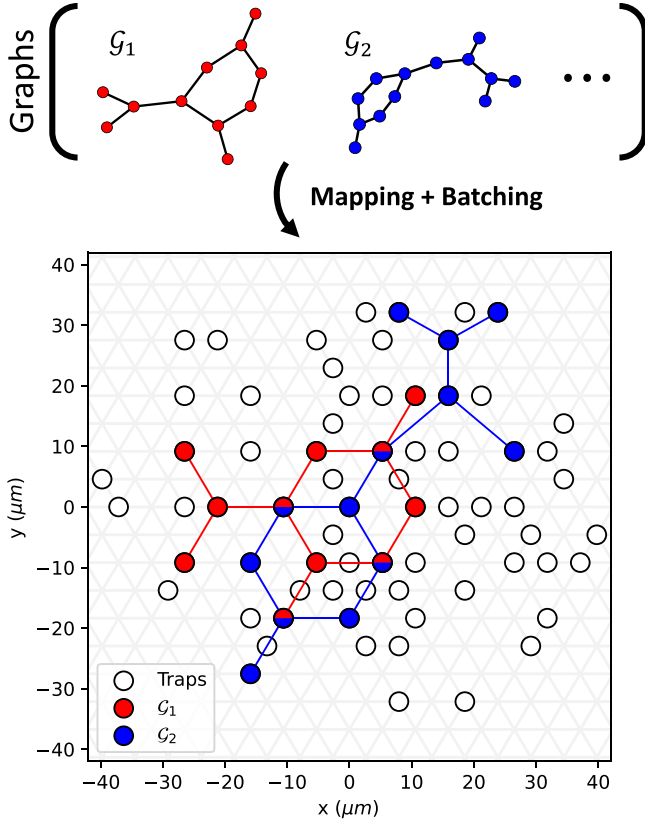


FIG. 12. Family of 66 graphs, ranging in sizes from 4 to 19 nodes, mapped and batched to the same SLM pattern (white circles) over a triangular grid with a spacing of $5.6 \mu\text{m}$. The traps used when implementing \mathcal{G}_1 (\mathcal{G}_2) are colored in red (blue). The bicolored traps are those used for both graphs.

APPENDIX F: CLASSICAL GRAPH KERNELS

A variety of classical kernels that do not require node or edge attributes are used in the main text to compare the performance of QEK on the PTCFM data set. In the following, a brief description of each is given.

1. SVM ϑ kernel

The SVM ϑ kernel was proposed as an alternative to the more computationally intensive Lovász ϑ kernel. Both ϑ kernels leverage the so-called orthogonal representation of a graph. Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, the orthogonal representation is an assignment of unit vectors $\{\mathbf{u}_i\}$ to each node of the graph, subject to the constraint that unit vectors associated with vertices that are not joined by an edge are orthogonal: $\langle \mathbf{u}_i, \mathbf{u}_j \rangle = 0$ if $\{i, j\} \notin \mathcal{E}$.

Orthogonal representations are not unique, but there is a particular representation associated with the ϑ number [68] of a graph. Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with n vertices, denote by $U_{\mathcal{G}}$ an orthogonal representation of \mathcal{G} and by C the space of unit vectors in \mathbb{R}^n . The ϑ number is defined as

$$\vartheta(\mathcal{G}) := \min_{\mathbf{c} \in C} \min_{U_{\mathcal{G}}} \max_{\mathbf{u}_i \in U_{\mathcal{G}}} \frac{1}{\langle \mathbf{c}, \mathbf{u}_i \rangle^2}. \quad (\text{F1})$$

From now on, we will always be referring to the particular orthogonal representation $U_{\mathcal{G}}$ that minimizes (F1).

Now consider a subset of vertices $B \subset \mathcal{V}$ and call $U_{\mathcal{G}|B}$ the orthogonal representation obtained from $U_{\mathcal{G}}$ by removing the vectors that are not in B :

$$U_{\mathcal{G}|B} := \{\mathbf{u}_i \in U_{\mathcal{G}} : i \in B\}. \quad (\text{F2})$$

Note that $U_{\mathcal{G}|B}$ preserves the global properties encoded in $U_{\mathcal{G}}$ through the orthogonal constraint and that $U_{\mathcal{G}|B}$ is not in general the orthogonal representation of the subgraph of \mathcal{G} containing only the vertices in B . Define the ϑ_B number

$$\vartheta_B(\mathcal{G}) := \min_{\mathbf{c} \in C} \max_{\mathbf{u}_i \in U_{\mathcal{G}|B}} \frac{1}{\langle \mathbf{c}, \mathbf{u}_i \rangle^2}. \quad (\text{F3})$$

We are ready now to give the definition of the Lovász ϑ kernel. Given two graphs $\mathcal{G}_1 = (\mathcal{V}_1, \mathcal{E}_1)$ and $\mathcal{G}_2 = (\mathcal{V}_2, \mathcal{E}_2)$, define

$$K_{\text{Lo}}(\mathcal{G}_1, \mathcal{G}_2) := \sum_{B_1 \subset \mathcal{V}_1} \sum_{B_2 \subset \mathcal{V}_2} \delta_{|B_1|, |B_2|} \frac{1}{Z} k(\vartheta_{B_1}, \vartheta_{B_2}), \quad (\text{F4})$$

where $Z = \binom{|\mathcal{V}_1|}{|B_1|} \binom{|\mathcal{V}_2|}{|B_2|}$, δ is the Kronecker delta, and k is a freely specifiable kernel (called the base kernel) from $\mathbb{R} \times \mathbb{R}$ to \mathbb{R} .

The SVM ϑ kernel is defined as (F4), but it uses an approximation for the ϑ numbers. Consider a graph \mathcal{G} with n vertices and adjacency matrix A and let $\rho \geq -\lambda$, where λ is the minimum eigenvalue of A . The matrix

$$\kappa := \frac{1}{\rho} A + I \quad (\text{F5})$$

is positive semidefinite. Define the maximization problem

$$\max_{\alpha_i \geq 0} 2 \sum_{i=1}^n \alpha_i - \sum_{i,j=1}^n \alpha_i \alpha_j \kappa_{ij}. \quad (\text{F6})$$

If $\{\alpha_i^*\}$ are the maximizers of (F6), then it can be proven that on certain families of graphs the quantity $\sum_i \alpha_i^*$ is with high

probability a constant factor approximation to $\vartheta(\mathcal{G})$,

$$\vartheta(\mathcal{G}) \leq \sum_{i=1}^n \alpha_i^* \leq \gamma \vartheta(\mathcal{G}), \quad (\text{F7})$$

for some γ . The SVM ϑ kernel then replaces the ϑ_B numbers on subgraphs with

$$\vartheta_B(\mathcal{G}) \rightarrow \sum_{j \in B} \alpha_j^*. \quad (\text{F8})$$

The SVM ϑ kernel requires a choice of base kernel $k : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$. We choose a translationally invariant universal kernel [69] $k(x, y) = (\beta + \|x - y\|^2)^{-\alpha}$, where α and β are two trainable hyperparameters.

2. Size kernel

Given two graphs $\mathcal{G}_1 = (\mathcal{V}_1, \mathcal{E}_1)$ and $\mathcal{G}_2 = (\mathcal{V}_2, \mathcal{E}_2)$, the size kernel is defined as

$$K_{\text{size}}(\mathcal{G}_1, \mathcal{G}_2) := e^{-\gamma(|\mathcal{V}_1| - |\mathcal{V}_2|)^2} \quad (\text{F9})$$

with a choice of hyperparameter $\gamma > 0$.

3. Graphlet sampling kernel

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and $\mathcal{H} = (\mathcal{V}_H, \mathcal{E}_H)$ be two graphs. We say that \mathcal{H} is a subgraph of \mathcal{G} if there exists an injective map $\alpha : \mathcal{V}_H \rightarrow \mathcal{V}$ such that $(u, v) \in \mathcal{E}_H \iff (\alpha(u), \alpha(v)) \in \mathcal{E}$. In general, it might be possible to map \mathcal{H} into \mathcal{G} in several different ways, i.e., the mapping α , if it exists, is not necessarily unique.

Given two graphs $\mathcal{G}_1 = (\mathcal{V}_1, \mathcal{E}_1)$ and $\mathcal{G}_2 = (\mathcal{V}_2, \mathcal{E}_2)$, the idea behind the graphlet kernel is to pick an integer $k < \min\{|\mathcal{V}_1|, |\mathcal{V}_2|\}$, enumerate all possible graphs of size k , and find the number of ways they can be mapped to \mathcal{G}_1 and \mathcal{G}_2 . Denote by $f_{\mathcal{G}_i}^{(k)}$ the vector where each entry counts the way a specific graph of size k can be mapped as a subgraph of \mathcal{G}_i . A kernel can then be defined as the dot product $f_{\mathcal{G}_1}^{(k)} \cdot f_{\mathcal{G}_2}^{(k)}$ between the two vectors.

The complexity of computing such a kernel scales as $O(n^k)$, as there are $\binom{n}{k}$ size- k subgraphs in a graph of size n . For this reason it is preferable to resort to sampling rather than complete enumeration [70]. Given a choice of integer N , graphs g_1, \dots, g_N of size between 3 and k are randomly sampled. The number of ways each g_i can be mapped as a subgraph of \mathcal{G}_j is computed and stored in a vector $f_{\mathcal{G}_j}$, and the graphlet sampling kernel is defined as the dot product

$$K_{\text{GS}}(\mathcal{G}_1, \mathcal{G}_2) := f_{\mathcal{G}_1} \cdot f_{\mathcal{G}_2}. \quad (\text{F10})$$

To account for the different size of \mathcal{G}_1 and \mathcal{G}_2 , each vector can be normalized by the total number of its subgraphs.

4. Random walk kernel

The random walk kernel is one of the oldest and most studied graph kernels [71]. Given two graphs $\mathcal{G}_1 = (\mathcal{V}_1, \mathcal{E}_1)$ and $\mathcal{G}_2 = (\mathcal{V}_2, \mathcal{E}_2)$, the idea is to measure the probability of simultaneous random walks of a certain length between two vertices in \mathcal{G}_1 and \mathcal{G}_2 .

Simultaneous random walks can be conveniently encoded in powers of the adjacency matrix on the product graph. The product graph $\mathcal{G}_1 \times \mathcal{G}_2 = \mathcal{G}_\times = (\mathcal{V}_\times, \mathcal{E}_\times)$ is defined as follows:

$$\mathcal{V}_\times := \{(u_i, u_r) \mid u_i \in \mathcal{V}_1, u_r \in \mathcal{V}_2\}, \quad (\text{F11})$$

$$\mathcal{E}_\times := \{((u_i, u_r), (v_j, v_s)) \mid (u_i, v_j) \in \mathcal{E}_1, (u_r, v_s) \in \mathcal{E}_2\}. \quad (\text{F12})$$

In other words, an edge in the product graph indicates that an edge exists between the end points in both \mathcal{G}_1 and \mathcal{G}_2 . If A_\times is the adjacency matrix of the product graph, then the entries of A_\times^k indicate the probability of a simultaneous random walk of length k between two vertices $u_i, v_j \in \mathcal{V}_1$ and $u_r, v_s \in \mathcal{V}_2$.

If $p, q \in \mathbb{R}^{|\mathcal{V}_\times|}$ are vectors representing the probability distribution of respectively starting or stopping the walk at a certain node of \mathcal{V}_\times , the first idea for a kernel would be to compute the sum $\sum_k q^T A_\times^k p$, which however may fail to converge. A simple modification to make the sum convergent is to choose an appropriate length-dependent weight $\mu(k)$:

$$K(\mathcal{G}_1, \mathcal{G}_2) := \sum_{k=0}^{\infty} \mu(k) q^T A_\times^k p. \quad (\text{F13})$$

The geometric random walk kernel is obtained by choosing the weights to be the coefficients of a geometric series $\mu(k) = \lambda^k$ and p and q to be uniform. If λ is tuned in such a way as to make the series convergent, the kernel reads

$$K_{\text{RW}}(\mathcal{G}_1, \mathcal{G}_2) := \sum_{k=0}^{\infty} \lambda^k e^T A_\times^k e = e^T (I - \lambda A_\times)^{-1} e, \quad (\text{F14})$$

where e denote vectors with all the entries equal to 1.

The cost of matrix inversion scales as the cube of the matrix size. If $|\mathcal{V}_1| = |\mathcal{V}_2| = n$, then the cost of the algorithm scales as $O(n^6)$, as it involves the inversion of an adjacency matrix of size $n^2 \times n^2$. Several methods were proposed in [72] to make the computation faster. The spectral decomposition method in particular allows one to reduce the complexity for unlabeled graphs to $O(n^3)$. Essentially, one exploits the fact that the adjacency matrix of the product graph can be decomposed in the tensor product of the individual adjacency matrices

$$A_\times = A_1 \otimes A_2, \quad (\text{F15})$$

which allows one to diagonalize each $n \times n$ adjacency matrix in $O(n^3)$ time and perform the inversion only on the diagonal components.

5. Shortest path kernel

Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, an edge path between two vertices $u, v \in \mathcal{V}$ is a sequence of edges (e_1, \dots, e_n) such that $u \in e_1, v \in e_n, e_i$ and e_{i+1} are contiguous (i.e., they have one of the end points in common), and $e_i \neq e_j$ for $i \neq j$. Computing the shortest edge path between any two nodes of a graph can be done in polynomial time with the Dijkstra [73] or Floyd-Warshall [74] algorithms, which makes it a viable feature to be probed by a graph kernel.

The first step of the shortest path kernel is to transform the graphs into shortest path graphs. Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, the shortest path graph $\mathcal{G}^S = (\mathcal{V}^S, \mathcal{E}^S)$ associated with \mathcal{G} is

defined as

$$\mathcal{V}^S = \mathcal{V}, \quad (\text{F16})$$

$$\mathcal{E}^S = \{(u, v) \mid \exists \text{ an edge path } (e_1, \dots, e_n) \text{ between } u \text{ and } v \text{ in } \mathcal{G}\}. \quad (\text{F17})$$

In addition, to each edge $e \in \mathcal{E}^S$ a label $l(e)$ is assigned given by the length of the shortest path in \mathcal{G} between its end points. The shortest path kernel is then defined as

$$K_{\text{SP}}(\mathcal{G}_1, \mathcal{G}_2) := \sum_{e \in \mathcal{E}_1^S} \sum_{p \in \mathcal{E}_2^S} k(e, p), \quad (\text{F18})$$

with k a kernel between edge paths such as the Brownian bridge kernel

$$k(e, p) := \max\{0, c - |l(e) - l(p)|\} \quad (\text{F19})$$

for a choice of c .

APPENDIX G: GEOMETRIC DIFFERENCE AND MAXIMUM QUANTUM-CLASSICAL SEPARATION

Given two kernel functions K_1 and K_2 , the geometric difference $g(K_1||K_2) = g_{12}$ described in [26] is an asymmetric distance function that quantifies whether or not the kernel K_2 has the potential to resolve data better than K_1 on some data set. In its simplest form, the geometric difference is defined as

$$g_{12} = \sqrt{\|\sqrt{K_2}(K_1)^{-1}\sqrt{K_2}\|_\infty}, \quad (\text{G1})$$

where $\|\cdot\|_\infty$ denotes the spectral norm.

The geometric difference becomes an especially useful metric when $K_1 = K_C$ is a classical kernel and $K_2 = K_Q$ is a quantum kernel. If N is the size of the data set, a value of

g_{CQ} of order \sqrt{N} or greater indicates that the geometry of the feature space induced by the quantum kernel is rich enough to be hard to learn classically, and the quantum kernel can potentially perform better than classical kernels. In that case, it is possible to artificially relabel the data set in order to maximally separate the kernels' performance. Such a relabeling process is a constructive proof of the existence of a certain data set on which one kernel performs much better than the other. If v is the eigenvector of $\sqrt{K_2}(K_1)^{-1}\sqrt{K_2}$ corresponding to the eigenvalue g_{12}^2 , the vector of new labels is given by $y_{\text{new}} = \sqrt{K_2}v$.

When dealing with a finite amount of training data, Eq. (34) should be regularized in order to stabilize the inversion of K_1 . The regularized expression reads

$$g_{12}(\lambda) = \sqrt{\|\sqrt{K_2}\sqrt{K_1}(K_1 + \lambda I)^{-2}\sqrt{K_1}\sqrt{K_2}\|_\infty}, \quad (\text{G2})$$

where λ is the regularization parameter. The geometric difference $g_{12}(\lambda)$ has a plateau for small λ , when the regularization parameter becomes smaller than the smallest eigenvalue of K_1 and decreases for increasing λ . The effect of λ is to introduce a certain amount of training error. The training error can be upper bounded by a quantity proportional to

$$g_{\text{tra}}(\lambda)^2 = \lambda^2 \|\sqrt{K_2}(K_1 + \lambda I)^{-2}\sqrt{K_2}\|_\infty. \quad (\text{G3})$$

Practically, one should look at the regime where g_{12} has not plateaued but the training error is still small enough.

A regularization should be introduced also in the relabeling procedure. The new labels are taken to be $y_{\text{new}} = \sqrt{K_Q}v$, where v is the eigenvector of the regularized matrix

$$\sqrt{K_Q}\sqrt{K_C}(K_C + \lambda I)^{-2}\sqrt{K_C}\sqrt{K_Q}$$

corresponding to the eigenvalue $g_{12}(\lambda)^2$.

-
- [1] L. C. Freeman, Visualizing social networks, *J. Soc. Struct.* **1**, 4 (2000).
- [2] A. Theocharidis, S. van Dongen, A. J. Enright, and T. C. Freeman, Network visualization and analysis of gene expression data using BioLayout *Express*^{3D}, *Nat. Protoc.* **4**, 1535 (2009).
- [3] R. Ferrer i Cancho and R. V. Solé, The small world of human language, *Proc. Biol. Sci.* **268**, 2261 (2001).
- [4] J. B. Schafer, J. A. Konstan, and J. Riedl, E-commerce recommendation applications, *Data Min. Knowl. Discov.* **5**, 115 (2001).
- [5] T. Pourhabibi, K.-L. Ong, B. H. Kam, and Y. L. Boo, Fraud detection: A systematic literature review of graph-based anomaly detection approaches, *Decis. Support Syst.* **133**, 113303 (2020).
- [6] G. Muzio, L. O'Bray, and K. Borgwardt, Biological network analysis with deep learning, *Brief. Bioinform.* **22**, 1515 (2020).
- [7] P. Goyal and E. Ferrara, Graph embedding techniques, applications, and performance: A survey, *Knowledge-Based Syst.* **151**, 78 (2018).
- [8] S. Bhagat, G. Cormode, and S. Muthukrishnan, *Node Classification in Social Networks* (Springer, Boston, 2011), pp. 115–148.
- [9] D. Liben-Nowell and J. Kleinberg, in *Proceedings of the 12th International Conference on Information and Knowledge Management* (ACM Press, New York, 2003), pp. 556–559.
- [10] M. Schuld and N. Killoran, Quantum Machine Learning in Feature Hilbert Spaces, *Phys. Rev. Lett.* **122**, 040504 (2019).
- [11] V. Havlíček, A. D. Córcoles, K. Temme, A. W. Harrow, A. Kandala, J. M. Chow, and J. M. Gambetta, Supervised learning with quantum-enhanced feature spaces, *Nature (London)* **567**, 209 (2019).
- [12] M. Schuld, K. Brádler, R. Israel, D. Su, and B. Gupt, Measuring the similarity of graphs with a Gaussian boson sampler, *Phys. Rev. A* **101**, 032314 (2020).
- [13] K. Kishi, T. Satoh, R. Raymond, N. Yamamoto, and Y. Sakakibara, Graph kernels encoding features of all subgraphs by quantum superposition, *IEEE J. Emerg. Sel. Top. Circuits Syst.* **12**, 602 (2022).
- [14] I. Cong, S. Choi, and M. D. Lukin, Quantum convolutional neural networks, *Nat. Phys.* **15**, 1273 (2019).
- [15] J. Zheng, Q. Gao, and Y. Lü, in *Proceedings of the 2021 40th Chinese Control Conference* (IEEE, Piscataway, 2021), pp. 6335–6340.

- [16] G. Verdon, T. McCourt, E. Luzhnica, V. Singh, S. Leichenauer, and J. Hidary, Quantum graph neural networks, [arXiv:1909.12264](#).
- [17] L.-P. Henry, S. Thabet, C. Dalyac, and L. Henriët, Quantum evolution kernel: Machine learning on graphs with programmable arrays of qubits, *Phys. Rev. A* **104**, 032416 (2021).
- [18] P. Mernyei, K. Meichanetzidis, and I. I. Ceylan, *Proceedings of the 39th International Conference on Machine Learning, Baltimore, 2022* (PMLR, Cambridge, 2022).
- [19] M. Larocca, F. Sauvage, F. M. Sbahi, G. Verdon, P. J. Coles, and M. Cerezo, Group-invariant quantum machine learning, *PRX Quantum* **3**, 030341 (2022).
- [20] A. Skolik, M. Cattelan, S. Yarkoni, T. Bäck, and V. Dunjko, Equivariant quantum circuits for learning on weighted graphs, [arXiv:2205.06109](#).
- [21] M. Ragone, P. Braccia, Q. T. Nguyen, L. Schatzki, P. J. Coles, F. Sauvage, M. Larocca, and M. Cerezo, Representation theory for geometric quantum machine learning, [arXiv:2210.07980](#).
- [22] Q. T. Nguyen, L. Schatzki, P. Braccia, M. Ragone, P. J. Coles, F. Sauvage, M. Larocca, and M. Cerezo, Theory for equivariant quantum neural networks, [arXiv:2210.08566](#).
- [23] L. Schatzki, M. Larocca, F. Sauvage, and M. Cerezo, Theoretical guarantees for permutation-equivariant quantum neural networks, [arXiv:2210.09974](#).
- [24] B. Y. Weisfeiler and A. A. Leman, The reduction of a graph to canonical form and the algebra which appears therein, *Nauchn.-Tech. Inf.* **229**, 12 (1968).
- [25] C. Helma, R. D. King, S. Kramer, and A. Srinivasan, The predictive toxicology challenge 2000–2001, *Bioinformatics* **17**, 107 (2001).
- [26] H.-Y. Huang, M. Broughton, M. Mohseni, R. Babbush, S. Boixo, H. Neven, and J. R. McClean, Power of data in quantum machine learning, *Nat. Commun.* **12**, 2631 (2021).
- [27] D. Barredo, V. Lienhard, S. de Léséleuc, T. Lahaye, and A. Browaeys, Synthetic three-dimensional atomic structures assembled atom by atom, *Nature (London)* **561**, 79 (2018).
- [28] 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, *Phys. Rev. X* **4**, 021034 (2014).
- [29] A. Browaeys and T. Lahaye, Many-body physics with individually controlled Rydberg atoms, *Nat. Phys.* **16**, 132 (2020).
- [30] L. Henriët, L. Béguin, A. Signoles, T. Lahaye, A. Browaeys, G.-O. Raymond, and C. Jurczak, Quantum computing with neutral atoms, *Quantum* **4**, 327 (2020).
- [31] M. Morgado and S. Whitlock, Quantum simulation and computing with Rydberg-interacting qubits, *AVS Quantum Sci.* **3**, 023501 (2021).
- [32] L. Béguin, A. Vernier, R. Chicireanu, T. Lahaye, and A. Browaeys, Direct Measurement of the van der Waals Interaction between Two Rydberg Atoms, *Phys. Rev. Lett.* **110**, 263201 (2013).
- [33] N. Šibalić, J. Pritchard, C. Adams, and K. Weatherill, ARC: An open-source library for calculating properties of alkali Rydberg atoms, *Comput. Phys. Commun.* **220**, 319 (2017).
- [34] D. Barredo, S. de Léséleuc, V. Lienhard, T. Lahaye, and A. Browaeys, An atom-by-atom assembler of defect-free arbitrary two-dimensional atomic arrays, *Science* **354**, 1021 (2016).
- [35] D. Jaksch, J. I. Cirac, P. Zoller, S. L. Rolston, R. Côté, and M. D. Lukin, Fast Quantum Gates for Neutral Atoms, *Phys. Rev. Lett.* **85**, 2208 (2000).
- [36] A. Gaëtan, Y. Miroshnychenko, T. Wilk, A. Chotia, M. Viteau, D. Comparat, P. Pillet, A. Browaeys, and P. Grangier, Observation of collective excitation of two individual atoms in the Rydberg blockade regime, *Nat. Phys.* **5**, 115 (2009).
- [37] H. Pichler, S.-T. Wang, L. Zhou, S. Choi, and M. D. Lukin, Quantum optimization for maximum independent set using Rydberg atom arrays, [arXiv:1808.10816](#).
- [38] L. Henriët, Robustness to spontaneous emission of a variational quantum algorithm, *Phys. Rev. A* **101**, 012335 (2020).
- [39] C. Dalyac, L. Henriët, E. Jeandel, W. Lechner, S. Perdrix, M. Porcheron, and M. Veshchezerova, Qualifying quantum approaches for hard industrial optimization problems. A case study in the field of smart-charging of electric vehicles, *EPJ Quantum Technol.* **8**, 12 (2021).
- [40] M.-T. Nguyen, J.-G. Liu, J. Wurtz, M. D. Lukin, S.-T. Wang, and H. Pichler, Quantum optimization with arbitrary connectivity using Rydberg atom arrays, *PRX Quantum* **4**, 010316 (2023).
- [41] S. Ebadi, A. Keesling, M. Cain, T. T. Wang, H. Levine, D. Bluvstein, G. Semeghini, A. Omran, J.-G. Liu, R. Samajdar, X.-Z. Luo, B. Nash, X. Gao, B. Barak, E. Farhi, S. Sachdev, N. Gemelke, L. Zhou, S. Choi, H. Pichler *et al.*, Quantum optimization of maximum independent set using Rydberg atom arrays, *Science* **376**, 1209 (2022).
- [42] M. Kim, K. Kim, J. Hwang, E.-G. Moon, and J. Ahn, Rydberg quantum wires for maximum independent set problems, *Nat. Phys.* **18**, 755 (2022).
- [43] A. Byun, M. Kim, and J. Ahn, Finding the maximum independent sets of platonic graphs using Rydberg atoms, *PRX Quantum* **3**, 030305 (2022).
- [44] J. Wurtz, P. L. S. Lopes, N. Gemelke, A. Keesling, and S. Wang, Industry applications of neutral-atom quantum computing solving independent set problems, [arXiv:2205.08500](#).
- [45] C. Dalyac and L. Henriët, Embedding the MIS problem for non-local graphs with bounded degree using 3D arrays of atoms, [arXiv:2209.05164](#).
- [46] E. H. Lieb and D. W. Robinson, The finite group velocity of quantum spin systems, *Commun. Math. Phys.* **28**, 251 (1972).
- [47] M. C. Tran, A. Y. Guo, Y. Su, J. R. Garrison, Z. Eldredge, M. Foss-Feig, A. M. Childs, and A. V. Gorshkov, Locality and Digital Quantum Simulation of Power-Law Interactions, *Phys. Rev. X* **9**, 031006 (2019).
- [48] L. Bai and E. R. Hancock, Graph kernels from the Jensen-Shannon divergence, *J. Math. Imaging Vis.* **47**, 60 (2013).
- [49] G. Verdon, T. McCourt, E. Luzhnica, V. Singh, S. Leichenauer, and J. Hidary, Quantum graph neural networks, [arXiv:1909.12264](#).
- [50] S. Thabet, R. Fouilland, and L. Henriët, Extending graph transformers with quantum computed aggregation, [arXiv:2210.10610](#).
- [51] G. Siglidis, G. Nikolentzos, S. Limnios, C. Giatsidis, K. Skianis, and M. Vazirgiannis, Grakel: A graph kernel library in Python, *J. Mach. Learn. Res.* **21**, 1993 (2020).
- [52] L. Boon, E. Ugarte-Berzal, J. Vandooren, and G. Opendakker, Protease propeptide structures, mechanisms of activation, and functions, *Crit. Rev. Biochem. Mol. Biol.* **55**, 111 (2020).

- [53] R. Shapiro and B. L. Vallee, Interaction of human placental ribonuclease with placental ribonuclease inhibitor, *Biochemistry* **30**, 2246 (1991).
- [54] N. M. Kriege, F. D. Johansson, and C. Morris, A survey on graph kernels, *Appl. Network Sci.* **5**, 6 (2020).
- [55] T. M. J. Fruchterman and E. M. Reingold, Graph drawing by force-directed placement, *Software: Practice Exper.* **21**, 1129 (1991).
- [56] C. Morris, M. Ritzert, M. Fey, W. L. Hamilton, J. E. Lenssen, G. Rattan, and M. Grohe, Weisfeiler and Leman go neural: Higher-order graph neural networks, *Proc. AAAI Conf. Artif. Intell.* **33**, 4602 (2019).
- [57] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, *ICLR 2019: International Conference on Learning Representations, New Orleans, 2019* (ICLR, La Jolla, 2019).
- [58] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, in *Proceedings of the 34th International Conference on Machine Learning*, edited by D. Precup and Y. W. Teh (PMLR, Cambridge, 2017), Vol. 70, pp. 1263–1272.
- [59] C. Ying, T. Cai, S. Luo, S. Zheng, G. Ke, D. He, Y. Shen, and T.-Y. Liu, in *Advances in Neural Information Processing Systems 34*, edited by M. Ranzato, A. Beygelzimer, Y. Dauphin, P. S. Liang, and J. Wortman Vaughan (Curran, Red Hook, 2021), p. 28877.
- [60] L. Rampásek, M. Galkin, V. P. Dwivedi, A. T. Luu, G. Wolf, and D. Beaini, Recipe for a general, powerful, scalable graph transformer, [arXiv:2205.12454](https://arxiv.org/abs/2205.12454).
- [61] D. Kreuzer, D. Beaini, W. Hamilton, V. Létourneau, and P. Tossou, in *Advances in Neural Information Processing Systems 34* (Ref. [59]), p. 21618.
- [62] D. Bluvstein, H. Levine, G. Semeghini, T. T. Wang, S. Ebadi, M. Kalinowski, A. Keesling, N. Maskara, H. Pichler, M. Greiner, V. Vuletic, and M. D. Lukin, A quantum processor based on coherent transport of entangled atom arrays, *Nature (London)* **604**, 451 (2022).
- [63] J. Shao and D. Tu, *The Jackknife and Bootstrap* (Springer, Berlin, 1995).
- [64] S. d. Leseleuc de kerouara, Quantum simulation of spin models with assembled arrays of Rydberg atoms, Ph.D. thesis, Université Paris Saclay, 2018.
- [65] S. de Léséleuc, D. Barredo, V. Lienhard, A. Browaeys, and T. Lahaye, Analysis of imperfections in the coherent optical excitation of single atoms to Rydberg states, *Phys. Rev. A* **97**, 053803 (2018).
- [66] S. Maskey, R. Levie, Y. Lee, and G. Kutyniok, Generalization analysis of message passing neural networks on large random graphs, in *Advances in Neural Information Processing Systems*, edited by A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho (2022).
- [67] M. D. J. MacKay, in *Information Theory, Inference, and Learning Algorithms* (Cambridge University Press, Cambridge, 2004), Chap. 20, pp. 284–292.
- [68] L. Lovasz, On the Shannon capacity of a graph, *IEEE Trans. Inf. Theory* **25**, 1 (1979).
- [69] C. A. Micchelli, Y. Xu, and H. Zhang, Universal kernels, *J. Mach. Learn. Res.* **7**, 2651 (2006).
- [70] N. Shervashidze, S. Vishwanathan, T. Petri, K. Mehlhorn, and K. Borgwardt, in *Proceedings of the JMLR Workshop and Conference* (MIT Press, Cambridge, 2009), Vol. 5, pp. 488–495.
- [71] T. Gärtner, P. Flach, and S. Wrobel, in *Learning Theory and Kernel Machines*, edited by B. Schölkopf and M. K. Warmuth (Springer, Berlin, 2003), pp. 129–143.
- [72] S. Vishwanathan, N. N. Schraudolph, R. Kondor, and K. M. Borgwardt, Graph kernels, *J. Mach. Learn. Res.* **11**, 1201 (2010).
- [73] E. W. Dijkstra, A note on two problems in connexion with graphs, *Numer. Math.* **1**, 269 (1959).
- [74] R. W. Floyd, Algorithm 97: Shortest path, *Commun. ACM* **5**, 345 (1962).