Quantum fidelity kernel with a trapped-ion simulation platform

Rodrigo Martínez-Peña,[1, 2, *] Miguel C. Soriano,[1] and Roberta Zambrini[1]

[1] *Instituto de Física Interdisciplinar y Sistemas Complejos (IFISC, UIB-CSIC), Campus Universitat de les Illes Balears E-07122, Palma de Mallorca, Spain*

[2] *Donostia International Physics Center, Paseo Manuel de Lardizabal 4, E-20018 San Sebastián, Spain*

(Dated: April 16, 2024)

Quantum kernel methods leverage a kernel function computed by embedding input information into the Hilbert space of a quantum system. However, large Hilbert spaces can hinder generalization capability, and the scalability of quantum kernels becomes an issue. To overcome these challenges, various strategies under the concept of inductive bias have been proposed. Bandwidth optimization is a promising approach that can be implemented using quantum simulation platforms. We propose trapped-ion simulation platforms as a means to compute quantum kernels, filling a gap in the previous literature and demonstrating their effectiveness for binary classification tasks. We compare the performance of the proposed method with an optimized classical kernel and evaluate the robustness of the quantum kernel against noise. The results show that ion trap platforms are well-suited for quantum kernel computation and can achieve high accuracy with only a few qubits.

I. INTRODUCTION

The availability of noisy intermediate-scale quantum (NISQ) devices [1] has catalyzed a surge in several research areas such as machine learning, many-body physics, chemistry, and combinatorial optimization [2]. In particular, quantum machine learning (QML) is a promising route toward quantum advantage, where some theoretical studies [3–6] and experimental evidence [7–9] have already indicated positive progress in this direction. While circuit-based NISQ computing implementations are developed worldwide [2], an alternative approach exploits NISQ analog or hybrid designs, with examples ranging from quantum annealing [10], boson sampling [11], and QML itself [12, 13]. In this work we will focus on the QML implementation of a kernel method in an analog simulation platform.

Kernel methods refer to classification or regression algorithms that leverage a kernel function, denoted as $K(\boldsymbol{x}, \boldsymbol{y})$, with $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{X}$, being $\mathcal{X}$ the input space. The kernel function transforms data points residing in the input space into a higher-dimensional space, called feature space. This transformation usually facilitates the distinction between different classes of data when they are not linearly separable. A quantum kernel method is generally understood as the case where a kernel function is computed by embedding the input information into the Hilbert space of a quantum system (the feature space), offloading the optimization process for the ML algorithm to a classical computer [14]. The most common approach to computing quantum kernels is based on the quantum state fidelity, i.e., the inner product of input-dependent quantum states. Quantum advantage has been claimed as possible in realistic problems if the embedding of inputs into these quantum states is classically intractable [14, 15].

However, finding a kernel function that could provide a quantum advantage remains an open problem. One plausible strategy that could come to our minds is to propose a very large quantum Hilbert space as the feature space since, in principle, it would be much harder to simulate classically. But very large Hilbert spaces can hinder the generalization capability of quantum models [16], performing similarly or even worse than classical ML models [4]. Indeed, training a kernel-based model ensures the discovery of the optimal model parameters as the training landscape is convex. However, this assurance is based on the assumption that the quantum kernel can be efficiently obtained from quantum hardware. With an increasing number of qubits, the number of measurements required to evaluate the kernels to sufficiently high precision might scale exponentially, hindering trainability [16–18].

Several alternatives have been proposed to avoid this exponential scaling, all of them under the notion of what is called inductive bias [8, 16]. An inductive bias is a restriction over the set of functions a given model can reproduce. They can be implemented in very diverse ways, such as projected quantum kernels [4], quantum Fisher kernels [18], exploiting structured-data encoding [19], and bandwidth optimization [20, 21]. While the former inductive biases can be problem-dependent, bandwidth optimization is a general strategy in which we simply tune the hyperparameters of our model. In particular, for classical data the decrease in performance with respect to the number of qubits can be compensated by the tuning of hyperparameters [20, 21], although it might be possible to find classical kernels that perform as well as them in some cases [22].

A promising choice for bandwidth optimization is quantum simulation platforms. Quantum simulations are believed to be a classically hard problem together with instantaneous quantum polynomial (IQP) random circuits or boson sampling [23]. Some analog simulation platforms have already been experimentally exploited for quantum kernels, such as Rydberg atoms [19] and nu-

* rodrigo.martinez@dipc.org

clear magnetic resonance (NMR) platforms [24], while more platforms have been numerically studied, such as driven-dissipative spin chains [2], a single Kerr nonlinear mode [25] and quantum annealers [26]. Each physical platform has its own advantages and challenges, which not only depend on experimental details but also on the QML technique. For example, NMR experiments allow the expectation value of observables to be obtained with a single shot, thus removing the influence of the statistical noise on the experimental results. The need to deal with statistical noise can be a source of severe overhead in some QML techniques such as quantum reservoir computing [13]. However, NMR platforms are difficult to initialise, whereas systems such as Rydberg atoms and trapped ions have very high fidelity [27, 28].

In this line of research, we propose to compute quantum kernels through trapped-ion simulation platforms, filling a gap in the previous literature. Trapped and laser-cooled atomic ions offer an excellent benchmark for simulating quantum spin models with interactions [29–31]. Optical fields can control the ions' Coulomb interaction, enabling customizable, long-range spin-spin interactions [28]. Ion trap quantum computers with logical quantum gates have already been used for classification tasks [32, 33], even with quantum kernel methods [34, 35].

In this work, we will introduce an inductive bias (i.e. a change in expressivity) in a transverse-field Ising model by applying bandwidth optimization, i.e., by tuning some of the model's hyperparameters. We show that this approach is suitable for solving binary classification tasks, comparing the performance with an optimized classical kernel. Our goal is to show via numerical simulations that an already available experimental platform can be harnessed for quantum kernel methods, demonstrating its robustness against noise. We introduce noise into the system in two different ways: decoherence with a depolarizing channel after the dynamics, and statistical noise with white noise at the entries of the kernel. For this, we require to use an error mitigation technique to keep the kernel as a positive semi-definite matrix, applying the shift method (see Sect. II B).

Our results show that ion trap platforms are perfectly suited for the computation of quantum kernels, being robust against different sources of noise. We evaluate the classification tasks for an increasing Hilbert space size, showing that only a few qubits are needed to obtain the best possible accuracy.

II. METHODS

A. Kernel methods and support vector machines

In machine learning, kernel methods are algorithms used for classification or regression. They utilize a kernel function, denoted as $K(\boldsymbol{x}, \boldsymbol{y})$, where $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{X}$ and $\mathcal{X}$ represents the input space, that transforms data points $\mathcal{X}$ to a higher-dimensional space, i.e. the feature space, usually facilitating the distinction between different classes of data. A function $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is defined as a kernel function if it fulfills two conditions: 1) it is symmetric, such that $K(\boldsymbol{x}, \boldsymbol{y}) = K(\boldsymbol{y}, \boldsymbol{x})$; and 2) it is a positive-semidefinite function. The latter is known as the Mercer condition [36], which can be expressed as:

$$\sum_{i=1}^{M} \sum_{j=1}^{M} K(\boldsymbol{x}_i, \boldsymbol{x}_j) c_i c_j \geq 0, \tag{1}$$

where M is the number of training samples and $\{c_i\}$ is the set of all possible real coefficients. A kernel function evaluated over pairs of data points defines a symmetric and positive semidefinite matrix whose elements are given by $K_{ij} := K(\boldsymbol{x}_i, \boldsymbol{x}_j)$.

Here we will consider Support Vector Machines (SVM) as our kernel method [37]. Our starting point will be to introduce the simplest case: linear binary classification. This problem consists on drawing a straight line that separates two classes by assigning two different labels to them, like $y = \pm 1$, on opposite sites of the line. The decision function, i.e. the function that assigns a label for new data, can be determined as:

$$D(\boldsymbol{x}) = \boldsymbol{w}^{\top} \cdot \boldsymbol{x} + b, \tag{2}$$

where $\boldsymbol{w} \in \mathbb{R}^m$ is a vector with the same dimension as the input and b is an offset. The labeling is introduced by using the sign function:

$$y(\boldsymbol{x}) = \mathrm{sgn}(\boldsymbol{w}^{\top} \cdot \boldsymbol{x} + b). \tag{3}$$

Let us assume we have M training inputs $\boldsymbol{x}_i$ belonging either to Class 1 with $y_i = 1$ or Class 2 with $y_i = -1$. Since we assume the training data are linearly separable, the following inequality is fulfilled:

$$y_i D(\boldsymbol{x}) = y_i(\boldsymbol{w}^{\top} \cdot \boldsymbol{x}_i + b) \geq 1, \quad i = 1, \ldots, M. \tag{4}$$

That is, the decision function yields $D(\boldsymbol{x}_i) \geq 1$ for $y_i = 1$ and $D(\boldsymbol{x}_i) \leq -1$ for $y_i = -1$, separating both classes by the hyperplanes $D(\boldsymbol{x}) = 1$ and $D(\boldsymbol{x}) = -1$. We can define a separating hyperplane between these two: $\boldsymbol{w}^{\top} \cdot \boldsymbol{x} + b = c$, for $-1 < c < 1$. The distance between the separating hyperplane and the training data sample nearest to the hyperplane is called the margin. The generalization ability of this model depends on the location of the separating hyperplane, and the hyperplane with the maximum margin is denoted as the optimal one.

In fact, $\boldsymbol{w}$ represents the orthogonal vector of the optimal separating hyperplane. If input data is bidimensional $\boldsymbol{w}$ defines a separation line, and in higher dimensions, it defines a separating hyperplane. However, a linear classifier will not suffice if the classification problem is nonlinear. Here comes the concept of the feature map $\phi(\boldsymbol{x})$, which embeds data points into a higher dimensional space (feature space) such that the modified inputs may become linearly separable. The new decision function becomes

$$D(\boldsymbol{x}) = \boldsymbol{w'}^{\top} \cdot \phi(\boldsymbol{x}) + b, \tag{5}$$

where $\boldsymbol{w'}$ is now the vector of the optimal separating hyperplane in the new feature space. The representer theorem [38] states that the vector $\boldsymbol{w'}$ can be written as a sum over a subset S of the training data points with real coefficients α_i:

$$\boldsymbol{w'} = \sum_{i\in S} \alpha_i y_i \phi(\boldsymbol{x}_i). \tag{6}$$

In the case of SVM, the subset S corresponds to the support vectors, that is, only the closest points to the decision boundary contribute. We write again the decision function with this new information,

$$D(\boldsymbol{x}) = \sum_{i\in S} \alpha_i y_i \phi(\boldsymbol{x}_i)^\top \cdot \phi(\boldsymbol{x}) + b, \tag{7}$$

finally arriving at the introduction of the kernel, defined as an inner product between feature vectors:

$$K(\boldsymbol{x}, \boldsymbol{x'}) = \phi(\boldsymbol{x})^\top \cdot \phi(\boldsymbol{x'}). \tag{8}$$

Under this definition, the kernel fulfills all the requirements exposed above. The kernel trick consists then on substituting the product of feature vectors by the kernel function, exploiting the fact that it might be easier to compute the inner product than the vectors themselves. This is especially relevant when the embedding happens in large feature spaces, as it usually happens with quantum kernels. We will use this fact later to define our quantum kernel.

The parameters α_i can be efficiently computed by solving the Lagrangian optimization problem for the soft margin SVM in its dual form:

$$\arg\max_{\alpha_i} \left(\sum_{i=1}^{M} \alpha_i - \frac{1}{2} \sum_{i=1}^{M}\sum_{j=1}^{M} \alpha_i \alpha_j y_i y_j K(\boldsymbol{x}_i, \boldsymbol{x}_j) \right), \tag{9}$$

subject to $\sum_{i=1}^{M} \alpha_i y_i = 0$ with $0 \leq \alpha_i \leq C$ for $i = 1, \ldots, M$. The penalty term $C > 0$ determines the balance between minimizing the training error and maximizing the margin. When C is small, it results in a large margin but potentially high training error.

For this paper, we will make use of the soft margin SVM method from the library Scikit-Learn [39]. In order to have a competitive benchmark to compare with, we take the classical kernel known as the Radial Basis Function (RBF). This kernel is defined as:

$$K(\boldsymbol{x}, \boldsymbol{y}) = e^{-\gamma ||\boldsymbol{x}-\boldsymbol{y}||^2}, \tag{10}$$

where γ controls the precision of the kernel.

B. Quantum fidelity kernel

Once the SVM framework is introduced, we define the quantum kernel that we will consider in this work, based on an analog implementation in an ion trap platform. We start by specifying the dynamical system that will process the input information. The Hamiltonian of our system is a Transverse-field Ising model:

$$H = \sum_{i>j=1}^{N} \frac{J}{|i-j|^\alpha} \sigma_i^x \sigma_j^x + \sum_{i=1}^{N} h_i \sigma_i^z, \tag{11}$$

where N is the number of spins, h_i is the onsite magnetic field for each spin, σ_i^a $(a = x, y, z)$ are the Pauli matrices and α is a hyperparameter that controls the decay of the spin-spin couplings, with maximum strength J for the nearest-neighbors in the chain. This model can be generated using a combination of an effective magnetic field and Ising interactions in ion trap platforms, with appropriate settings of the spin phases [28]. Spin-spin interactions are mediated by global laser beams that couple spin and motion according to the Mølmer-Sørensen scheme [40]. The coupling decay is approximated by a power-law, where α can be tuned between 0 and 3 by varying parameters of the trap [28, 41]. Here we will fix $\alpha = 1.13$ and $J = 1$, following the values adopted by experimentalists to obtain a medium-length interaction range [29]. Our proposal is to use this ion trap to implement a kernel, by encoding the inputs in the local magnetic fields, where site-dependent laser-induced Stark shifts can be used to prepare them [42], see [29, 43, 44] for some examples of experimental implementations. In particular, we introduce the input into the external magnetic field of some or all the spins as $h_i = hx_i$, depending on the dimension of the input vector. In general:

$$H(\boldsymbol{x}) = \sum_{i>j=1}^{N} \frac{J}{|i-j|^\alpha} \sigma_i^x \sigma_j^x + h \sum_{i=1}^{N} x_i \sigma_i^z, \tag{12}$$

where h is the strength of the magnetic field for all spins and $\boldsymbol{x} = \{x_1, x_2, ...\}$ is the input vector for each input instance with values scaled in the interval $x_i \in [-1, 1]$. The chain length N is set to be equal or larger than the number of input features. For less features than the chain length we will set the magnetic field in the remaining spins to zero: for instance, for an input of only two features (and $N \geq 2$), $\boldsymbol{x} = \{x_1, x_2, 0, ..., 0\}$. If N is a multiple of the number of input features, one could redundantly encode the inputs in more than one spin. In this input protocol, the number of qubits must scale linearly with respect to the number of input features.

We will now move on to the definition of the quantum kernel. The system is initialized with all spins in the state $|0\rangle$. Then, we apply the unitary operator of the dynamics:

$$|\psi(\boldsymbol{x})\rangle = e^{-iH(\boldsymbol{x})\Delta t} |0\rangle, \tag{13}$$

where Δt indicates the time of the simulation. This operation of encoding the input vector into a new Hilbert space is the feature mapping. To construct the kernel, we need a dot product between two vectors in the new feature space. This is expressed as

$$K(\boldsymbol{x},\boldsymbol{y}) = |\langle\psi(\boldsymbol{y})|\psi(\boldsymbol{x})\rangle|^2 = |\langle 0|e^{iH(\boldsymbol{y})\Delta t}e^{-iH(\boldsymbol{x})\Delta t}|0\rangle|^2, \tag{14}$$

where we introduced the squared absolute value for experimental purposes. In order to compute the kernel matrix element for inputs $\boldsymbol{x}$ and $\boldsymbol{y}$, we need to measure the overlap between states $|\psi(\boldsymbol{x})\rangle$ and $|\psi(\boldsymbol{y})\rangle$. There are several ways of tackling this problem. The most popular one with quantum computers is the quantum kernel estimation method [15]. This method would require applying the following chain of operations in our system:

1. Initialize the system at $|0\rangle$ through optical pumping.

2. Apply the sequence of unitary operators $e^{iH(\boldsymbol{y})\Delta t}e^{-iH(\boldsymbol{x})\Delta t}$, where each unitary is simulated as described after Eq. (11). Examples of programmed sequences of unitaries in trapped ions can be seen, for instance, in gate-based quantum computing [45], Floquet systems [43] and preparation of thermofield states [46].

3. Measure the state of all qubits in the z direction through e.g. fluorescence collected into a CCD camera.

4. Repeat the whole process to estimate the frequency of obtaining the state $|0\rangle$ in all qubits after measuring, which indeed approximates Eq. (14).

Let us now describe the different sources of noise that we will study in the system. There are common sources of decoherence for simulations and gate-based computations in trapped ions, such as stray magnetic and electric fields, mode frequency drifts, off-resonant motional excitation, and spontaneous emission [47]. The main point of decoherence is that it can set a time limit for quantum simulations. Therefore, we model the experimental noise produced by decoherence with a depolarizing channel after the unitary dynamics [48, 49], which is a simple model that can be easily studied in the context of quantum kernels:

$$\tilde{\rho}(\boldsymbol{x}) = (1-p)\rho(\boldsymbol{x}) + p\frac{I}{2^N}, \tag{15}$$

where p is the probability of replacing the quantum state by the maximally mixed state $I/2^N$ and $\rho(\boldsymbol{x}) = |\psi(\boldsymbol{x})\rangle\langle\psi(\boldsymbol{x})|$. We assume that p is the same for all inputs $\boldsymbol{x}$ to simplify the setting, but this is not necessarily true in general. The quantum kernel is given now by the Hilbert-Schmidt product of the feature vectors in the space of density matrices:

$$\tilde{K}(\boldsymbol{x},\boldsymbol{y}) = \mathrm{tr}(\tilde{\rho}(\boldsymbol{x})\tilde{\rho}(\boldsymbol{y})) = (1-p)^2K(\boldsymbol{x},\boldsymbol{y}) + p(2-p)/2^N. \tag{16}$$

Besides, we can add a Gaussian random number to the kernel matrix elements in order to simulate statistical noise due to finite sampling (justified by the central limit theorem):

$$\tilde{K}'(\boldsymbol{x},\boldsymbol{y}) = \tilde{K}(\boldsymbol{x},\boldsymbol{y}) + \xi(\boldsymbol{x},\boldsymbol{y}), \tag{17}$$

where $\xi(\boldsymbol{x},\boldsymbol{y})$ is generated from a normal distribution of zero mean and standard deviation s for each pair of inputs $(\boldsymbol{x},\boldsymbol{y})$. This can be related to the number of measurements V through the relation $s \sim 1/\sqrt{V}$. Since the kernel matrix has to be symmetric, we set $\xi(\boldsymbol{x},\boldsymbol{y}) = \xi(\boldsymbol{y},\boldsymbol{x})$. In this work, we consider $p = 0.01, 0.1$ and $s = 0.01, 0.1$.

In order to use a kernel matrix for classification tasks, it must be a positive-semidefinite matrix. The two sources of noise we previously introduced might hinder this feature, so some mitigation techniques should be applied. We use the Tikhonov regularization [48], also known as the shifting technique in this context [49]. It consists on shifting all the eigenvalues by the minimum non-positive eigenvalue of $\tilde{K}'$. This is equivalent to subtracting it from the diagonal:

$$\overline{K} = \begin{cases} \tilde{K}' - \lambda_{\min}I & \text{if } \lambda_{\min} < 0 \\ \tilde{K}' & \text{else} \end{cases}, \tag{18}$$

where $\overline{K}$ is now a symmetric positive-semidefinite matrix.

C. Optimization

As was mentioned in the Introduction, we will introduce an inductive bias in our models by applying bandwidth optimization. To this end, we perform a grid search that will help to visualize the performance for all explored values of hyperparameters. This can be numerically expensive for many hyperparameters (and hard to visualize), so we limit ourselves to grid searches that vary two hyperparameters simultaneously. For the classical case, we vary the scaling factor of the RBF kernel γ in Eq. (10) and the regularization parameter C (introduced after Eq. (9)). For the quantum case, we will vary Δt and h while fixing $C = 1$ (recall we already fixed $J = 1$ and $\alpha = 1.13$). We will also have a look at the performance as the number of spins increases for $N = 2, 4, 6$.

To measure the performance of our kernel methods, we use the classification accuracy. It is defined as the number of correct predictions (CP) divided by the total number of predictions (TP):

$$A = \frac{\mathrm{CP}}{\mathrm{TP}}. \tag{19}$$

This metric is appropriate since our dataset is well-balanced, with half of each set from each classification class.

In order to find the optimal hyperparameters, we adopted a training, validation, and testing strategy. Training is used to find the optimal parameters of the SVM model for the training set (for a given value of the hyperparameters). The validation set allows us to compute the performance of the model (with the same hyperparameters) for new data that the trained model has not seen. Based on the validation performance, we make the grid search in order to find the optimal hyperparameters. Finally, at the test, we evaluate the final performance of the model, with trained parameters and optimized hyperparameters, for a new set of inputs.

Three binary classification tasks are evaluated. The Circles (C) and Moons (M) tasks were generated from the library ScikitLearn [39]. These two tasks are paradigmatic examples of benchmark classification tasks because they are easy to generate, while not trivially solvable, and are very popular in the quantum kernel literature [14, 26, 50–53], even with experiments [24, 54]. The third task, called Ad Hoc (AH), was extracted from the Qiskit dataset library in Python [55]. This dataset was originally designed to be exactly separable using quantum kernels as proposed in [15]. We evaluate here the capability of the classical RBF and quantum trapped ion kernels to solve these three tasks.

For the Circles and Moons tasks, Gaussian noise is introduced during the generation of the datasets (with standard deviation equal to 0.2 and 0.3 respectively), in such a way that points belonging to one class can enter into the area that corresponds to the other class (See Figs. 1 (a) and (b)). For the Ad Hoc task (Fig. 1 (c)), we set the separation gap as $\Delta = 0.3$ (see [15] for more details). Note that the input data noise is the only noise source we considered for the classical kernel.

The datasets are equally divided into three sets (train, validation, and test) with 333 points per set. We remark that given this number of data points and the resolution of the grid search (we used 100 points per hyperparameter), we may find more than one equal-valued minima in the grid search. We need to select one of them to compute the test accuracy. Our choice is to take the middle element of the ordered list of minima, having checked that the other points work equally well.

Task	γ	C	A_{val}	A_{test}
C	$2.78 \cdot 10^{-3}$	$2.98 \cdot 10^{7}$	0.91	0.89
M	$4.98 \cdot 10^{-2}$	$8.11 \cdot 10^{6}$	0.94	0.89
AH	$2.98 \cdot 10^{0}$	$2.11 \cdot 10^{4}$	0.85	0.84

TABLE I: Optimal hyperparameters for the classical RBF kernel. The tasks are represented as C (Circles), M (Moons), and AH (Ad Hoc). The values of A_{val} and A_{test} represent the accuracy with the optimal hyperparameters for the validation and test sets respectively.

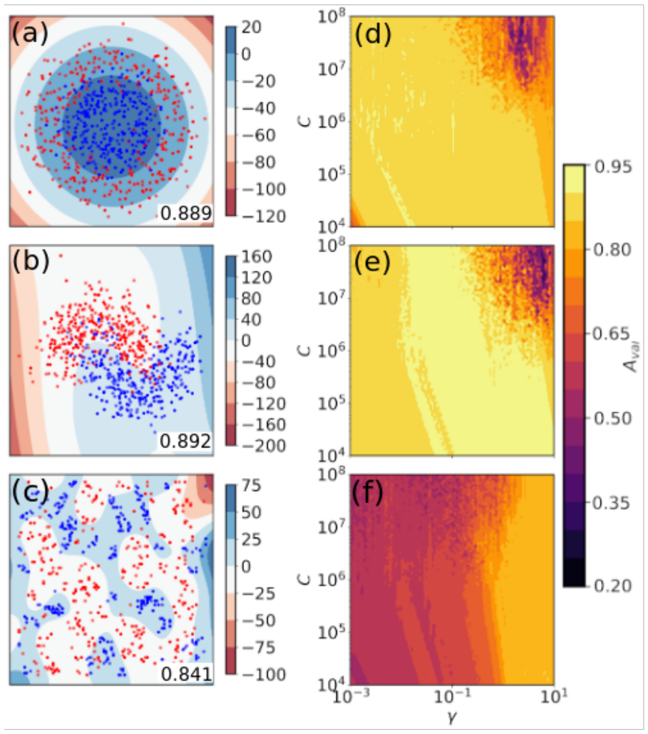

FIG. 1: Decision function ((a)-(c)) and optimization landscape ((d)-(f)) of the RBF classical kernel model. Figures (a)-(c) were obtained for the test sets by extracting the optimal hyperparameters of figures (d)-(f). The three different tasks are referred to as Circles ((a) and (d)), Moons ((b) and (e)), and Ad Hoc ((c) and (f)).

III. NUMERICAL RESULTS

A. Classical kernel

We begin by presenting the results of the classical RBF kernel. Table I shows the values of the optimal hyperparameters for the three different tasks, together with the validation accuracy A_{val} and test accuracy A_{test}. We observe that the values of validation accuracy are larger than those of test accuracy, as expected. The fact that we do not surpass the value 0.9 of test accuracy in any of the tasks is due to the presence of noise during the generation of the datasets, as explained in the previous section.

Figures 1 (d)-(f) represent the classification accuracy during the validation procedure of the classical kernel over the hyperparameter space. The optimal hyperparameters of Table I were obtained by selecting the values that maximize the validation accuracy, as described in Section II C. For the Circles and Moon tasks (Figs. 1 (d) and (e)), we observe that a broad combination of hyperparameters allows us to solve them. However, the Ad Hoc task (Fig. 1 (f)) seems to require a higher non-linear response from the kernel. Using the kernel with a small scale factor γ tends to make the SVM behave like a linear classifier. Conversely, a large scale factor γ makes the output classifier highly sensitive to small input changes, which could lead to overfitting even with margin maximization. Figure 1 (f) shows how this task is best solved with the higher values of γ (higher nonlinearity) but without overfitting, as the test accuracy of Table I demonstrates.

B. Quantum kernel

Now we introduce the results of the quantum kernel. Table II shows the optimal hyperparameters and classification accuracy of the quantum kernel with the Hamiltonian given in Eq. (12) for the same tasks. This table contains only the noiseless case, i.e., the kernel given by Eq. (14). As in the classical case, we compute the validation and test accuracy for the optimal hyperparameters (h and Δt), while exploring the number of qubits N as well.

No noise					
Task	N	h	Δt	A_{val}	A_{test}
C	2	0.40	2.11	0.76	0.72
C	4	0.37	16.30	0.91	0.89
C	6	1.87	2.31	0.91	0.87
M	2	0.42	49.77	0.64	0.57
M	4	2.26	2.78	0.93	0.90
M	6	2.98	2.11	0.93	0.90
AH	2	1.63	2.21	0.74	0.72
AH	4	1.56	6.73	0.84	0.78
AH	6	1.56	6.73	0.87	0.82

TABLE II: Optimal hyperparameters for the quantum kernel model without noise. The tasks are represented as C (Circles), M (Moons), and AH (Ad Hoc). N represents the number of qubits, h is the strength of the external magnetic field and Δt is the unitary evolution's time scale. The values of A_{val} and A_{test} represent the accuracy with the optimal hyperparameters for the validation and test sets respectively.

The first thing that calls our attention from the results presented in Table II is that the performance significantly improves when increasing the number of qubits from $N = 2$ to $N = 4$ in all tasks. This transition is directly related to the size of the quantum feature space where the inputs are introduced, increasing the expressivity of the model. However, the Ad Hoc task is the only one that also displays a significant change when increasing the number of spins from $N = 4$ to $N = 6$. This suggests that the task at hand determines the optimal size of the Hilbert space, depending on the expressiveness required.

Regarding the effect of statistical (s) and depolarization (p) noise, we analyze four different situations where the quantum kernel is given by Eqs. (16)-(18), summarized in Table III. In the first case, we added a small amount of both statistical and depolarizing noise, with $s = 0.01$ and $p = 0.01$. Here the test accuracy is slightly decreased in almost all the cases, but we also observe that it can be also slightly increased, as in the Circles task with $N = 6$. Then, we increase the statistical noise to $s = 0.1$, finding a small negative effect over validation accuracy that was not present with $s = 0.01$. Instead, when increasing the depolarizing noise to $p = 0.1$ while keeping $s = 0.01$, both validation and test accuracy return to values similar to the noiseless case, with even higher performance in the Ad Hoc task for $N = 6$. Finally, the case of larger statistical and depolarizing noise ($s = 0.1$ and $p = 0.1$) shows that test performance is almost identical to the noiseless situation, even larger in some cases. However, the validation accuracy decreases again as in the case of $s = 0.1$ and $p = 0.01$, meaning that this effect is produced by the statistical noise.

To visualize the change in performance with noise, Fig. 2 represents the performance of the quantum kernel for the Ad Hoc task in all the studied situations. The x axis describes the noise parameters and the y axis represents the test accuracy. The color of each bar corresponds to a given number of qubits. Figure 2 illustrates our main findings, namely that test accuracy increases with system size and that statistical noise can be detrimental, although an appropriate amount of depolarizing noise can compensate for this negative effect, even finding the best performance for the Ad Hoc task for $N = 6$ with $s = 0.01$ and $p = 0.1$.

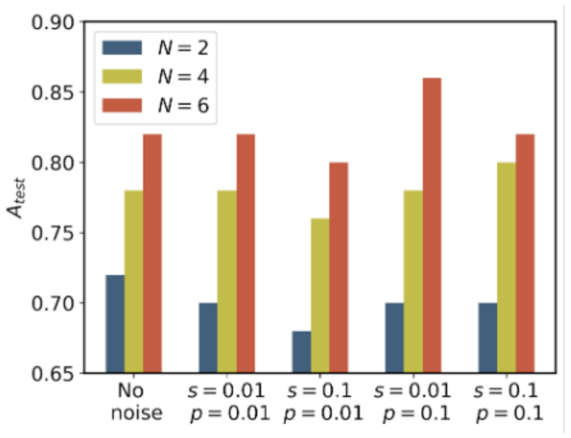

FIG. 2: Test accuracy of the quantum kernel model for the Ad Hoc task. The color of each bar corresponds to a given number of qubits and each group of bars corresponds to a set of noise parameters.

When comparing the optimal quantum kernels with the classical kernel, the quantum models for $N = 4$ and $N = 6$ obtain a very similar performance to the classical case, even under the effect of noise. For the Ad Hoc task, we can also find one situation where the test accuracy of the quantum kernel is larger than in the classical case ($N = 6$ for $s = 0.01$ and $p = 0.1$). On the one hand, the fact that statistical noise (less measurement precision) tends to decrease the performance is an evident negative effect that can have severe consequences, such as the loss of any possible generalization advantage of quantum kernels [49]. However, as shown in [5], quantum kernels

		$s = 0.01$ and $p = 0.01$				$s = 0.1$ and $p = 0.01$				$s = 0.01$ and $p = 0.1$				$s = 0.1$ with $p = 0.1$			
Task	N	h	Δt	A_{val}	A_{test}	h	Δt	A_{val}	A_{test}	h	Δt	A_{val}	A_{test}	h	Δt	A_{val}	A_{test}
C	2	0.18	19.63	0.78	0.72	0.71	7.39	0.76	0.71	0.49	1.92	0.78	0.71	0.71	7.39	0.76	0.71
C	4	1.02	2.54	0.91	0.88	0.28	29.84	0.90	0.88	0.89	2.92	0.91	**0.89**	0.42	25.95	0.90	0.88
C	6	0.56	6.73	0.91	**0.89**	0.74	4.64	0.90	0.88	0.77	5.09	0.91	0.88	0.37	59.95	0.90	**0.89**
M	2	1.42	2.31	0.64	0.58	1.87	2.54	0.62	0.57	1.56	2.11	0.63	0.60	1.29	3.20	0.63	0.54
M	4	1.18	8.90	0.93	0.87	1.79	3.05	0.91	**0.90**	1.96	2.92	0.93	**0.90**	2.26	2.11	0.91	0.89
M	6	2.85	2.21	0.93	**0.90**	1.87	3.35	0.92	**0.90**	2.85	2.21	0.93	**0.90**	1.71	3.20	0.91	**0.90**
AH	2	1.79	2.01	0.74	0.70	3.77	1.00	0.68	0.68	1.79	2.01	0.74	0.70	1.79	2.01	0.68	0.70
AH	4	1.42	7.06	0.82	0.78	2.36	8.50	0.72	0.76	1.56	6.73	0.81	0.78	1.42	7.39	0.72	0.80
AH	6	1.29	8.50	0.87	0.82	1.18	10.24	0.76	0.80	1.56	18.74	0.86	**0.86**	1.42	7.39	0.75	0.82

TABLE III: Optimal hyperparameters for the quantum kernel model with noise. The tasks are represented as C (Circles), M (Moons) and AH (Ad Hoc). N represents the number of qubits, h is the strength of the external magnetic field and Δt is the unitary evolution's time scale. The values of A_{val} and A_{test} represent the accuracy with the optimal hyperparameters for the validation and test sets respectively.

with a controlled additive sampling noise in SVM are robust, where the noisy hyperplane is close to the noiseless hyperplane. On the other hand, depolarizing noise corresponds to a loss of information about the quantum state with probability p, so large values of p would also hinder the performance of quantum devices. But as we observe here, it can act as a regularization parameter for small values. We remark that in any case, techniques can be carried out to mitigate the effect of both sources of noise [34, 48, 49].

We now proceed to visualize the relationship between the optimal hyperparameters and task performance. Figure 3 shows the validation accuracy in terms of the hyperparameters h and Δt for the systems associated with $s = 0.01$ and $p = 0.01$ in Table III. We notice that the optimization landscape is qualitatively similar in all the studied situations, so Fig. 3 is a representative example. The left and right columns correspond to $N = 2$ and $N = 6$ respectively, while each row corresponds to a different task. As already observed in Table III, the validation accuracy reaches higher values for $N = 6$ than for $N = 2$ due to the larger Hilbert space. But Fig. 3 also shows that the region of hyperparameters that could solve the tasks may be broad. The Circles task (Fig. 3 (d)) exhibits a wide range of hyperparameters yielding high accuracy, with a possible linear trend for $h\Delta t \sim$ cte in the center. The Moons task (Fig. 3 (e)) reveals an even wider range of successful hyperparameters. Interestingly, also for the Ad Hoc task (Fig. 3 (f)), optimal hyperparameters consistently cluster in the center of the figures. Our analysis illustrates that these central hyperparameter combinations lead to high kernel expressivity.

One may try to explain the previous results by establishing a connection between the value of the optimal hyperparameters and the underlying dynamical features of the model. However, it seems more complicated than that. Notice that the kernel of Eq. (14) has three hyperparameters related to the dynamics, J, h, and Δt, but in fact, only two of them are independent (when combined as $J\Delta t$ and $h\Delta t$ in the unitary operator). Then the coefficient J/h indicates the relative weight of each of the two terms of the Hamiltonian in Eq. 12. Recalling that we previously fixed $J = 1$, our results suggest that the relative weight of the external magnetic field term to the coupling term would only become a relevant factor in solving some specific tasks. For instance, the Moons task does not strongly depend on the rate J/h: in Fig. 3 (e) we observe that values of the hyperparameters such as $0.3 < h < 3$ with $\Delta t \sim 10$ provide a similar performance. That is, the rates $1/3 < J/h < 3$ work equally well. On the contrary, the Ad Hoc task requires a very specific ratio J/h: Fig. 3 (f) shows a clear optimal region of hyperparameters in the center of the figure, around $\Delta t \sim 10$ and $h \sim 1$.

IV. DISCUSSION

Kernel methods are a promising direction in the search for applications of quantum machine learning techniques. In particular, classically intractable quantum simulation models can be exploited to explore quantum advantages. Our work proposes ion trap platforms as an analog experimental platform to compute quantum kernels, demonstrating their viability through numerical simulations of realistic scenarios with depolarizing and statistical noise.

The computation of quantum kernels proposed here for trapped ions builds on the kernel estimation protocol of Ref. [15]. Computing the unitary dynamics given by $-H(\boldsymbol{y})$ only requires the change of sign of the couplings and the definition of new magnetic fields depending on the input $\boldsymbol{y}$. One potential challenge is whether it is possible to make this sudden change between $H(\boldsymbol{x})$ and $-H(\boldsymbol{y})$ on the fly. There are other possible routes to measure the overlap between two quantum states that might be applied to this quantum simulation platform. For example, the SWAP test and extensions [56, 57] would allow evolving in parallel the states $|\psi(\boldsymbol{x})\rangle$ and $|\psi(\boldsymbol{y})\rangle$ without the sudden quench. Classical machine learning techniques can help here [58, 59], and full tomography allows us to obtain the quantum kernel as well [60]. Finally, a promising direction might be to use randomized mea-

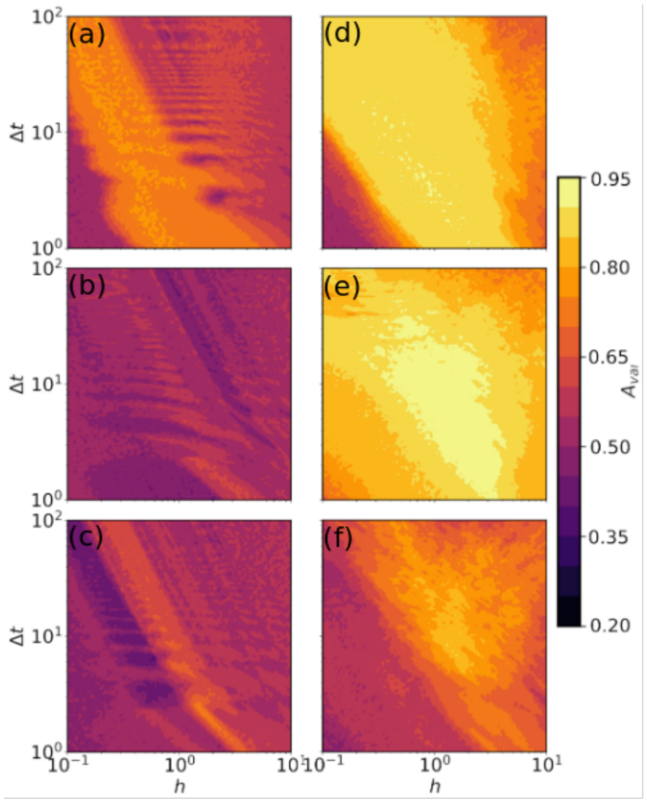

FIG. 3: Optimization landscape for the quantum kernel model with the validation datasets. The statistical noise strength is given by $s = 0.01$ and the depolarizing noise parameter is $p = 0.01$. The left column corresponds to $N = 2$ while the right column corresponds to $N = 6$. The top row is the Circles task, the middle row is the Moons task, and the bottom row is the Ad Hoc task.

surements [61, 62], where the overlap of states generated by independent Hamiltonians can be measured.

We evaluated the performance of the proposed quantum kernel in three different binary classification tasks and compared the results with the well-known RBF classical kernel. We made a grid search of optimal hyperparameters for both classical and quantum models, also exploring the number of qubits in the latter. The bandwidth optimization of the quantum model introduces an inductive bias that restricts the type of functions that can be solved. We find that both classical and quantum models reach a very similar performance, closely saturating the maximum possible test accuracy for the given test sets without overfitting. This means that the proposed quantum model is operational for classification tasks, being robust under the presence of different sources of noise.

The search for optimal quantum kernels was carried out for only two hyperparameters with a very specific Hamiltonian (eq. (11)) and input scheme, but more possibilities can be explored. The hyperparameters α and C can be also tuned, the input can be redundantly injected (like $\boldsymbol{x} = \{x_1, x_2, x_1, x_2, ...\}$ in eq. (11)), or we can even add new terms to the Hamiltonian such as a magnetic field in the x axis. We tested these possibilities for the presented tasks, reaching a similar or worse performance. However, we do not discard that a more systematic evaluation of these strategies could bring an improvement. In fact, the quantitative performance in different tasks is expected to vary under different interactions or driving in the Hamiltonian.

Future work will be devoted to exploring the effect of increasing the number of qubits to solve tasks with a larger number of input features. On the one hand, one can expand the number of features that can be codified in a single qubit by feeding them over time with an input-dependent driving field, as presented in Ref. [60]. On the other hand, projected quantum kernels can be explored in this setting [4], limiting the effective Hilbert space dimension to maximize the generalization ability of the model.

ACKNOWLEDGMENTS

We acknowledge the Spanish State Research Agency, through the María de Maeztu project CEX2021-001164-M funded by the MCIN/AEI/10.13039/501100011033 and through the COQUSY project PID2022-140506NB-C21 and -C22 funded by MCIN/AEI/10.13039/501100011033, MINECO, through the QUANTUM SPAIN project, and EU through the RTRP - NextGenerationEU within the framework of the Digital Spain 2025 Agenda. The CSIC Interdisciplinary Thematic Platform (PTI) on Quantum Technologies in Spain is also acknowledged. RMP acknowledges the QCDI project funded by the Spanish Government and part of this work was also funded by MICINN/AEI/FEDER and the University of the Balearic Islands through a pre-doctoral fellowship (Grant No. MDM-2017-0711-18-1).

[1] J. Preskill, Quantum computing in the nisq era and beyond, Quantum **2**, 79 (2018).

[2] K. Bharti, A. Cervera-Lierta, T. H. Kyaw, T. Haug, S. Alperin-Lea, A. Anand, M. Degroote, H. Heimonen, J. S. Kottmann, T. Menke, *et al.*, Noisy intermediate-scale quantum algorithms, Reviews of Modern Physics **94**, 015004 (2022).

[3] H.-Y. Huang, R. Kueng, and J. Preskill, Information-theoretic bounds on quantum advantage in machine learning, Physical Review Letters **126**, 190505 (2021).

[4] H.-Y. Huang, M. Broughton, M. Mohseni, R. Babbush, S. Boixo, H. Neven, and J. R. McClean, Power of data in

quantum machine learning, Nature communications **12**, 1 (2021).

[5] Y. Liu, S. Arunachalam, and K. Temme, A rigorous and robust quantum speed-up in supervised machine learning, Nature Physics **17**, 1013 (2021).

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

[7] H.-Y. Huang, M. Broughton, J. Cotler, S. Chen, J. Li, M. Mohseni, H. Neven, R. Babbush, R. Kueng, J. Preskill, *et al.*, Quantum advantage in learning from experiments, Science **376**, 1182 (2022).

[8] M. Cerezo, G. Verdon, H.-Y. Huang, L. Cincio, and P. J. Coles, Challenges and opportunities in quantum machine learning, Nature Computational Science **2**, 567 (2022).

[9] A. Melnikov, M. Kordzanganeh, A. Alodjants, and R.-K. Lee, Quantum machine learning: From physics to software engineering, Advances in Physics: X **8**, 2165452 (2023).

[10] A. Das and B. K. Chakrabarti, Colloquium: Quantum annealing and analog quantum computation, Reviews of Modern Physics **80**, 1061 (2008).

[11] D. J. Brod, E. F. Galvão, A. Crespi, R. Osellame, N. Spagnolo, and F. Sciarrino, Photonic implementation of boson sampling: a review, Advanced Photonics **1**, 034001 (2019).

[12] D. Marković and J. Grollier, Quantum neuromorphic computing, Applied physics letters **117** (2020).

[13] P. Mujal, R. Martínez-Peña, J. Nokkala, J. García-Beni, G. L. Giorgi, M. C. Soriano, and R. Zambrini, Opportunities in quantum reservoir computing and extreme learning machines, Advanced Quantum Technologies **4**, 2100027 (2021).

[14] M. Schuld and N. Killoran, Quantum machine learning in feature hilbert spaces, Physical review letters **122**, 040504 (2019).

[15] 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 **567**, 209 (2019).

[16] J. Kübler, S. Buchholz, and B. Schölkopf, The inductive bias of quantum kernels, Advances in Neural Information Processing Systems **34**, 12661 (2021).

[17] S. Thanasilp, S. Wang, M. Cerezo, and Z. Holmes, Exponential concentration and untrainability in quantum kernel methods, arXiv preprint arXiv:2208.11060 (2022).

[18] Y. Suzuki, H. Kawaguchi, and N. Yamamoto, Quantum fisher kernel for mitigating the vanishing similarity issue, arXiv preprint arXiv:2210.16581 (2022).

[19] B. Albrecht, C. Dalyac, L. Leclerc, L. Ortiz-Gutiérrez, S. Thabet, M. D'Arcangelo, J. R. Cline, V. E. Elfving, L. Lassablière, H. Silvério, *et al.*, Quantum feature maps for graph machine learning on a neutral atom quantum processor, Physical Review A **107**, 042615 (2023).

[20] R. Shaydulin and S. M. Wild, Importance of kernel bandwidth in quantum machine learning, Physical Review A **106**, 042407 (2022).

[21] A. Canatar, E. Peters, C. Pehlevan, S. M. Wild, and R. Shaydulin, Bandwidth enables generalization in quantum kernel models, arXiv preprint arXiv:2206.06686 (2022).

[22] L. Slattery, R. Shaydulin, S. Chakrabarti, M. Pistoia, S. Khairy, and S. M. Wild, Numerical evidence against advantage with quantum fidelity kernels on classical data, Physical Review A **107**, 062417 (2023).

[23] A. W. Harrow and A. Montanaro, Quantum computational supremacy, Nature **549**, 203 (2017).

[24] T. Kusumoto, K. Mitarai, K. Fujii, M. Kitagawa, and M. Negoro, Experimental quantum kernel trick with nuclear spins in a solid, npj Quantum Information **7**, 1 (2021).

[25] J. Liu, C. Zhong, M. Otten, A. Chandra, C. L. Cortes, C. Ti, S. K. Gray, and X. Han, Quantum kerr learning, Machine Learning: Science and Technology **4**, 025003 (2023).

[26] R. Yang, S. Bosch, B. Kiani, S. Lloyd, and A. Lupascu, Analog quantum variational embedding classifier, Physical Review Applied **19**, 054023 (2023).

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

[28] C. Monroe, W. C. Campbell, L.-M. Duan, Z.-X. Gong, A. V. Gorshkov, P. Hess, R. Islam, K. Kim, N. M. Linke, G. Pagano, *et al.*, Programmable quantum simulations of spin systems with trapped ions, Reviews of Modern Physics **93**, 025001 (2021).

[29] J. Smith, A. Lee, P. Richerme, B. Neyenhuis, P. W. Hess, P. Hauke, M. Heyl, D. A. Huse, and C. Monroe, Many-body localization in a quantum simulator with programmable random disorder, Nature Physics **12**, 907 (2016).

[30] J. Zhang, G. Pagano, P. W. Hess, A. Kyprianidis, P. Becker, H. Kaplan, A. V. Gorshkov, Z.-X. Gong, and C. Monroe, Observation of a many-body dynamical phase transition with a 53-qubit quantum simulator, Nature **551**, 601 (2017).

[31] H. B. Kaplan, L. Guo, W. L. Tan, A. De, F. Marquardt, G. Pagano, and C. Monroe, Many-body dephasing in a trapped-ion quantum simulator, Physical Review Letters **125**, 120605 (2020).

[32] M. S. Rudolph, N. B. Toussaint, A. Katabarwa, S. Johri, B. Peropadre, and A. Perdomo-Ortiz, Generation of high-resolution handwritten digits with an ion-trap quantum computer, Physical Review X **12**, 031010 (2022).

[33] S. Johri, S. Debnath, A. Mocherla, A. Singk, A. Prakash, J. Kim, and I. Kerenidis, Nearest centroid classification on a trapped ion quantum computer, npj Quantum Information **7**, 122 (2021).

[34] S. Moradi, C. Brandner, M. Coggins, R. Wille, W. Drexler, and L. Papp, Error mitigation for quantum kernel based machine learning methods on ionq and ibm quantum computers, arXiv preprint arXiv:2206.01573 (2022).

[35] T. Suzuki, T. Hasebe, and T. Miyazaki, Quantum support vector machines for classification and regression on a trapped-ion quantum computer, arXiv preprint arXiv:2307.02091 (2023).

[36] J. Mercer, Xvi. functions of positive and negative type, and their connection the theory of integral equations, Philosophical transactions of the royal society of London. Series A **209**, 415 (1909).

[37] V. N. Vapnik, The nature of statistical learning theory (1995).

[38] B. Schölkopf and A. J. Smola, *Learning with kernels: support vector machines, regularization, optimization, and beyond* (MIT press, 2002).

[39] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer,

R. Weiss, V. Dubourg, *et al.*, Scikit-learn: Machine learning in python, the Journal of machine Learning research **12**, 2825 (2011).

[40] K. Mølmer and A. Sørensen, Multiparticle entanglement of hot trapped ions, Physical Review Letters **82**, 1835 (1999).

[41] R. Islam, C. Senko, W. C. Campbell, S. Korenblit, J. Smith, A. Lee, E. Edwards, C.-C. Wang, J. Freericks, and C. Monroe, Emergence and frustration of magnetism with variable-range interactions in a quantum simulator, science **340**, 583 (2013).

[42] A. C. Lee, J. Smith, P. Richerme, B. Neyenhuis, P. W. Hess, J. Zhang, and C. Monroe, Engineering large stark shifts for control of individual clock state qubits, Physical Review A **94**, 042308 (2016).

[43] J. Zhang, P. W. Hess, A. Kyprianidis, P. Becker, A. Lee, J. Smith, G. Pagano, I.-D. Potirniche, A. C. Potter, A. Vishwanath, *et al.*, Observation of a discrete time crystal, Nature **543**, 217 (2017).

[44] W. Morong, F. Liu, P. Becker, K. Collins, L. Feng, A. Kyprianidis, G. Pagano, T. You, A. Gorshkov, and C. Monroe, Observation of stark many-body localization without disorder, Nature **599**, 393 (2021).

[45] C. D. Bruzewicz, J. Chiaverini, R. McConnell, and J. M. Sage, Trapped-ion quantum computing: Progress and challenges, Applied Physics Reviews **6** (2019).

[46] D. Zhu, S. Johri, N. M. Linke, K. Landsman, C. Huerta Alderete, N. H. Nguyen, A. Matsuura, T. Hsieh, and C. Monroe, Generation of thermofield double states and critical ground states with a quantum computer, Proceedings of the National Academy of Sciences **117**, 25402 (2020).

[47] D. J. Wineland, C. Monroe, W. M. Itano, D. Leibfried, B. E. King, and D. M. Meekhof, Experimental issues in coherent quantum-state manipulation of trapped atomic ions, Journal of research of the National Institute of Standards and Technology **103**, 259 (1998).

[48] T. Hubregtsen, D. Wierichs, E. Gil-Fuster, P.-J. H. Derks, P. K. Faehrmann, and J. J. Meyer, Training quantum embedding kernels on near-term quantum computers, Physical Review A **106**, 042431 (2022).

[49] X. Wang, Y. Du, Y. Luo, and D. Tao, Towards understanding the power of quantum kernels in the nisq era, Quantum **5**, 531 (2021).

[50] S. S. Vedaie, M. Noori, J. S. Oberoi, B. C. Sanders, and E. Zahedinejad, Quantum multiple kernel learning, arXiv preprint arXiv:2011.09694 (2020).

[51] S. Lloyd, M. Schuld, A. Ijaz, J. Izaac, and N. Killoran, Quantum embeddings for machine learning, arXiv preprint arXiv:2001.03622 (2020).

[52] L. H. Li, D.-B. Zhang, and Z. Wang, Quantum kernels with gaussian state encoding for machine learning, Physics Letters A **436**, 128088 (2022).

[53] V. Rastunkov, J.-E. Park, A. Mitra, B. Quanz, S. Wood, C. Codella, H. Higgins, and J. Broz, Boosting method for automated feature space discovery in supervised quantum machine learning models, arXiv preprint arXiv:2205.12199 (2022).

[54] K. Bartkiewicz, C. Gneiting, A. Černoch, K. Jiráková, K. Lemr, and F. Nori, Experimental kernel-based quantum machine learning in finite feature space, Scientific Reports **10**, 12356 (2020).

[55] Qiskit contributors, Qiskit: An open-source framework for quantum computing, 10.5281/zenodo.2573505 (2023).

[56] H. Buhrman, R. Cleve, J. Watrous, and R. De Wolf, Quantum fingerprinting, Physical Review Letters **87**, 167902 (2001).

[57] M. Fanizza, M. Rosati, M. Skotiniotis, J. Calsamiglia, and V. Giovannetti, Beyond the swap test: optimal estimation of quantum state overlap, Physical review letters **124**, 060503 (2020).

[58] L. Cincio, Y. Subaşı, A. T. Sornborger, and P. J. Coles, Learning the quantum algorithm for state overlap, New Journal of Physics **20**, 113022 (2018).

[59] L. Cincio, K. Rudinger, M. Sarovar, and P. J. Coles, Machine learning of noise-resilient quantum circuits, PRX Quantum **2**, 010324 (2021).

[60] V. Heyraud, Z. Li, Z. Denis, A. Le Boité, and C. Ciuti, Noisy quantum kernel machines, Physical Review A **106**, 052421 (2022).

[61] A. Elben, B. Vermersch, R. van Bijnen, C. Kokail, T. Brydges, C. Maier, M. K. Joshi, R. Blatt, C. F. Roos, and P. Zoller, Cross-platform verification of intermediate scale quantum devices, Physical review letters **124**, 010504 (2020).

[62] T. Haug, C. N. Self, and M. Kim, Quantum machine learning of large datasets using randomized measurements, Machine Learning: Science and Technology (2021).