Covariant quantum kernels for data with group structure

Jennifer R. Glick,[1] Tanvi P. Gujarati,[2] Antonio D. Córcoles,[1] Youngseok Kim,[1]
Abhinav Kandala,[1] Jay M. Gambetta,[1] and Kristan Temme[1, *]

[1] *IBM Quantum, IBM T.J. Watson Research Center, Yorktown Heights, NY 10598, USA*
[2] *IBM Quantum, Almaden Research Center, San Jose, California 95120, USA*

(Dated: March 23, 2022)

The use of kernel functions is a common technique to extract important features from data sets. A quantum computer can be used to estimate kernel entries as transition amplitudes of unitary circuits. Quantum kernels exist that, subject to computational hardness assumptions, cannot be computed classically. It is an important challenge to find quantum kernels that provide an advantage in the classification of real-world data. We introduce a class of quantum kernels that can be used for data with a group structure. The kernel is defined in terms of a unitary representation of the group and a fiducial state that can be optimized using a technique called kernel alignment. We apply this method to a learning problem on a coset-space that embodies the structure of many essential learning problems on groups. We implement the learning algorithm with 27 qubits on a superconducting processor.

The core tenet of the kernel method in machine learning is that it allows one to apply linear statistical methods to data sets that are complex and non-linear in nature. The kernel function K corresponds to an inner product of vectors in a (potentially) high-dimensional Euclidean space referred to as the feature space [1]. The datum $\boldsymbol{x} \in \mathcal{X}$ is mapped to this high-dimensional space by means of a non-linear feature map $\Phi(\boldsymbol{x})$. This feature map has to be chosen in such a way that the data, initially not tractable by linear methods, can be linearly separated in the higher-dimensional space. The choice of the non-linear map is therefore central to this approach. The use of the kernel trick allows one to process the data in the high-dimensional space without explicitly computing the feature vector. This trick has found its way into many machine learning tasks such as classification [1], regression [2, 3], clustering [4, 5], correlation analysis [6] and filtering [7]. The most prominent application of the kernel method is arguably for binary classification in the use of the support vector machine (SVM) [1, 8]. This is also the learning problem we consider in the experimental implementation here. After seeing m training samples $\boldsymbol{x}_i \in \mathcal{X}$ with labels $y_i = \pm 1$, we train a classifier f that accurately predicts the label $y = f(\boldsymbol{x})$ of a previously unseen datum $\boldsymbol{x}$.

A quantum computer can be used to perform the feature mapping into a quantum-enhanced feature space and estimate the kernel matrix for the training data [9, 10]. This quantum kernel can then be used in most machine learning algorithms that use the kernel method. In fact, it was observed in [9] that many of the recently introduced variational quantum algorithms [11–15] reduce to a quantum kernel method, since these algorithms are only linear methods in the quantum feature space. Following [9], a datum $\boldsymbol{x} \in \mathcal{X}$ is mapped to an n-qubit quantum feature state $\Phi(\boldsymbol{x}) = U(\boldsymbol{x}) |0^n\rangle \langle 0^n| U^\dagger(\boldsymbol{x})$ through a unitary circuit family $U(\boldsymbol{x})$. The unitary depends non-linearly on the datum and needs to have an efficient implementation. The kernel entry for two samples $\boldsymbol{x}, \tilde{\boldsymbol{x}}$ is obtained as the Hilbert-Schmidt inner product $K(\boldsymbol{x}, \tilde{\boldsymbol{x}}) = \mathrm{tr}\left[\Phi^\dagger(\boldsymbol{x})\Phi(\tilde{\boldsymbol{x}})\right]$ of the two quantum feature states, and can be understood as the transition amplitude

$$K(\boldsymbol{x}, \tilde{\boldsymbol{x}}) = |\langle 0^n| U^\dagger(\boldsymbol{x}) U(\tilde{\boldsymbol{x}}) |0^n\rangle|^2. \tag{1}$$

The kernel entry can be estimated on a quantum computer by evolving the initial state $|0^n\rangle$ with $U^\dagger(\boldsymbol{x})U(\tilde{\boldsymbol{x}})$ and recording the frequency of the all-zero outcome 0^n. This procedure (c.f. Appendix A.1) is referred to as quantum kernel estimation (QKE).

Learning algorithms that use QKE have a proven advantage over all classical learners for specifically constructed learning problems [16]. A core challenge is to establish this advantage in practically relevant settings. We take steps to address this challenge by identifying a class of learning problems that provide a natural fit for QKE and generalize the result in [16]. What these learning problems have in common is that the data space is a subset of a group $\mathcal{X} \subseteq G$. The study of data with group structure has a long tradition in statistics [17]. Important learning problems such as ranking [17, 18] can be expressed as the learning of permutations [19, 20]. Other examples are learning problems in coset spaces such as partial rankings, Q-sort data, error correcting codes and homogeneous spaces [17]. We consider a general class of feature map circuits that we call *covariant feature maps* and that can be used for data space with a group structure. The corresponding quantum feature states are intimately related to covariant measurements [21]. The covariant feature map is defined relative to a unitary representation [22] $D_{\boldsymbol{x}}$ for the group G with $\boldsymbol{x} \in G$, and a fiducial state $|\psi\rangle \in \mathbb{C}^{2^n}$ on n qubits as $\Phi(\boldsymbol{x}) = D_{\boldsymbol{x}} |\psi\rangle \langle\psi| D_{\boldsymbol{x}}^\dagger$. The *covariant quantum kernel* is then estimated as the fidelity [see Eq. (1)]

$$K(\boldsymbol{x}, \tilde{\boldsymbol{x}}) = |\langle\psi| D_{\boldsymbol{x}}^\dagger D_{\tilde{\boldsymbol{x}}} |\psi\rangle|^2. \tag{2}$$

[*] kptemme@ibm.com

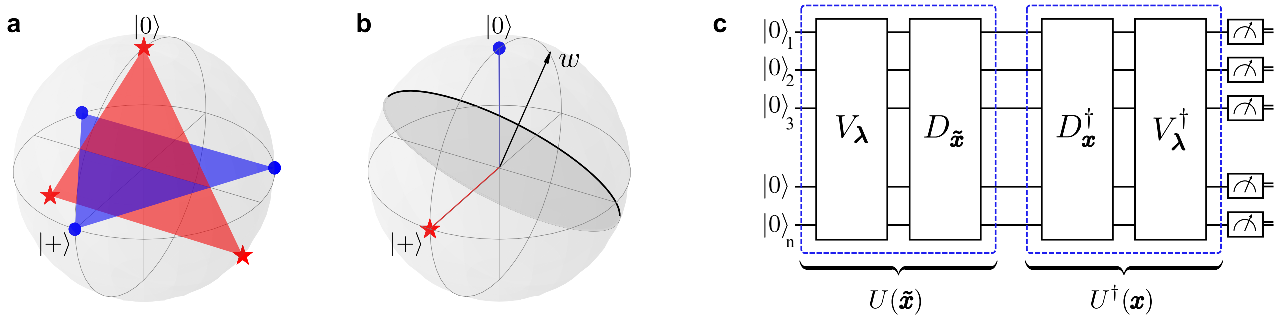
a
$|0\rangle$
$|+\rangle$
b
$|0\rangle$
w
$|+\rangle$
c
$|0\rangle_1$
$|0\rangle_2$
$|0\rangle_3$
$|0\rangle$
$|0\rangle_n$
$V_{\boldsymbol{\lambda}}$
$D_{\tilde{\boldsymbol{x}}}$
$D^{\dagger}_{\boldsymbol{x}}$
$V^{\dagger}_{\boldsymbol{\lambda}}$
$U(\tilde{\boldsymbol{x}})$
$U^{\dagger}(\boldsymbol{x})$

FIG. 1. **Labeling cosets**. (a), (b) Two covariant feature maps for a single-qubit example of the *labeling cosets* learning problem introduced in the text. We take $S = \{\mathbb{1}, A, A^2\}$ as subgroup of $G = SU(2)$, where $A = \exp(i(2\pi/3)X)$. Choosing two elements $\boldsymbol{c}_+, \boldsymbol{c}_- \in SU(2)$, with their representation $D_{\boldsymbol{c}_+} = \mathbb{1}$, $D_{\boldsymbol{c}_-} = H$, we form two left-cosets: $C_+ = D_{\boldsymbol{c}_+} S = S$ and $C_- = D_{\boldsymbol{c}_-} S = \{H, HA, HA^2\}$. (a) Non-linearly-separable case with fiducial state $|\psi\rangle = |0\rangle$. Data points $\boldsymbol{x} \in C_\pm$, for which $\boldsymbol{x} = \boldsymbol{c}_\pm \boldsymbol{s}$, are mapped to the states $D_{\boldsymbol{c}_+} D_{\boldsymbol{s}} |0\rangle$ and $D_{\boldsymbol{c}_-} D_{\boldsymbol{s}} |0\rangle$, for all $\boldsymbol{s} \in S$, which correspond to points on the Bloch sphere marked by three red stars and three blue dots, respectively. As a result, elements from different cosets live on orthogonal planes and cannot be linearly separated. (b) Separable case with fiducial state as a subgroup invariant state $|\psi\rangle = |+\rangle$. Elements from a given coset are mapped to a single state: $D_{\boldsymbol{c}_+} D_{\boldsymbol{s}} |+\rangle = D_{\boldsymbol{c}_+} |+\rangle = |+\rangle$ or $D_{\boldsymbol{c}_-} D_{\boldsymbol{s}} |+\rangle = D_{\boldsymbol{c}_-} |+\rangle = |0\rangle$ for all $\boldsymbol{s} \in S$. A classifier needs only to distinguish between two points $|+\rangle$ and $|0\rangle$, which are linearly separable by an optimal hyperplane tilted at 45° from the zy plane.(c) Quantum circuit for calculating matrix elements of a covariant quantum kernel. The feature map $U(\boldsymbol{x})$ is defined in terms of a unitary representation $D_{\boldsymbol{x}}$ for a group G with $\boldsymbol{x} \in G$ and a circuit $V_{\boldsymbol{\lambda}}$ that prepares the fiducial state $|\psi_{\boldsymbol{\lambda}}\rangle = V_{\boldsymbol{\lambda}} |0^n\rangle$. The frequency of measuring all zeros in the computational basis is an estimate for the kernel matrix element $K_{\boldsymbol{\lambda}}(\boldsymbol{x}, \tilde{\boldsymbol{x}})$

We assume the fiducial state $|\psi\rangle = V\,|0^n\rangle$ can be prepared by applying an efficient quantum circuit V. Likewise, it is important to also assume that the representation D of G can be implemented efficiently on a quantum computer. In this case, the QKE routine reduces to estimating the transition amplitude $K(\boldsymbol{x}, \tilde{\boldsymbol{x}}) = |\,\langle 0^n|\, V^\dagger D^\dagger_{\boldsymbol{x}} D_{\tilde{\boldsymbol{x}}} V\,|0^n\rangle\,|^2$, and the feature map circuit becomes $U(\boldsymbol{x}) = D_{\boldsymbol{x}} V$; cf. Fig. 1(c). The kernel as defined here is left-invariant under the group action. A right-invariant definition (c.f. Appendix B) is immediate.

Covariant quantum kernels can lead to a provable separation between quantum and classical learners [16] for specific problems. The learning problem in [16] is a binary classification problem for data from the group $\mathbb{Z}^*_p$ (integer multiplication modular p) and reduces to the discrete logarithm problem (DLOG) [23]. The kernel in [16] is a special case of the covariant quantum kernel introduced here, with the regular representation of $\mathbb{Z}^*_p$ and a fiducial state that is the uniform superposition of group elements obtained from applications of the generator (c.f. Appendix B.1). In this example, it becomes apparent that different fiducial states can lead to vastly different kernels. While the aforementioned fiducial state produces a kernel that can lead to an efficient learning problem for the DLOG classification problem, a fiducial state that is given by one of computational basis states would lead to an identity kernel, matrix which is well-known to have extremely poor performance. This illustrates that the choice of $|\psi\rangle$ is essential for the performance of the quantum kernel. If sufficient structural knowledge about the problem is present a suitable fiducial state can be chosen *a priori*. However, we also want a method to optimize the fiducial state subject to the available data, if no prior knowledge is available. The objective of this optimization will depend on the learning problem. We consider a binary classification problem with SVMs. For other types of kernel functions, objectives have been proposed [24, 25] that are motivated by quantum information theoretic insights. Here, we will follow a method commonly used in the classical literature referred to as *kernel alignment* [26, 27].

To optimize the fiducial state, it is first generated by applying a variational quantum circuit $V_{\boldsymbol{\lambda}}$ with parameters $\boldsymbol{\lambda} \in \Omega \subset \mathbb{R}^q$ to the state $|\psi_{\boldsymbol{\lambda}}\rangle = V_{\boldsymbol{\lambda}}\,|0^n\rangle$. This will lead to parametrized quantum kernels $K_{\boldsymbol{\lambda}}(\boldsymbol{x}, \tilde{\boldsymbol{x}})$ with feature map circuit $D_{\boldsymbol{x}} V_{\boldsymbol{\lambda}}$ as depicted in Fig. 1(c). The parameters $\boldsymbol{\lambda}$ are optimized with kernel alignment. The binary classifier associated with kernel $K_{\boldsymbol{\lambda}}$ is given as a linear threshold function $f(\boldsymbol{x}) = \mathrm{sign}(\sum_{i=1}^m y_i \alpha_i K_{\boldsymbol{\lambda}}(\tilde{\boldsymbol{x}}_i, \boldsymbol{x}))$ with model parameters $\{\alpha_i\}_{i=1\ldots m}$ for a training set of size $|\{\boldsymbol{x}_i\}| = m$ and labels $y_i = \pm 1$. We use a "weighted" version of the alignment [27, 28] to optimize the kernel parameters while the SVM is used to optimize the model parameters. This approach can be seen as optimizing the SVM upper bound on the generalization error directly (c.f. Appendix A). The cost function

$$F(\boldsymbol{\alpha}, \boldsymbol{\lambda}) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j K_{\boldsymbol{\lambda}}(\boldsymbol{x}_i, \boldsymbol{x}_j), \qquad (3)$$

is related to an upper bound to the generalization error when maximized over $\boldsymbol{\alpha}$. The weighted kernel alignment minimizes this upper bound with respect to $\boldsymbol{\lambda}$. The procedure is expressed as the optimization,

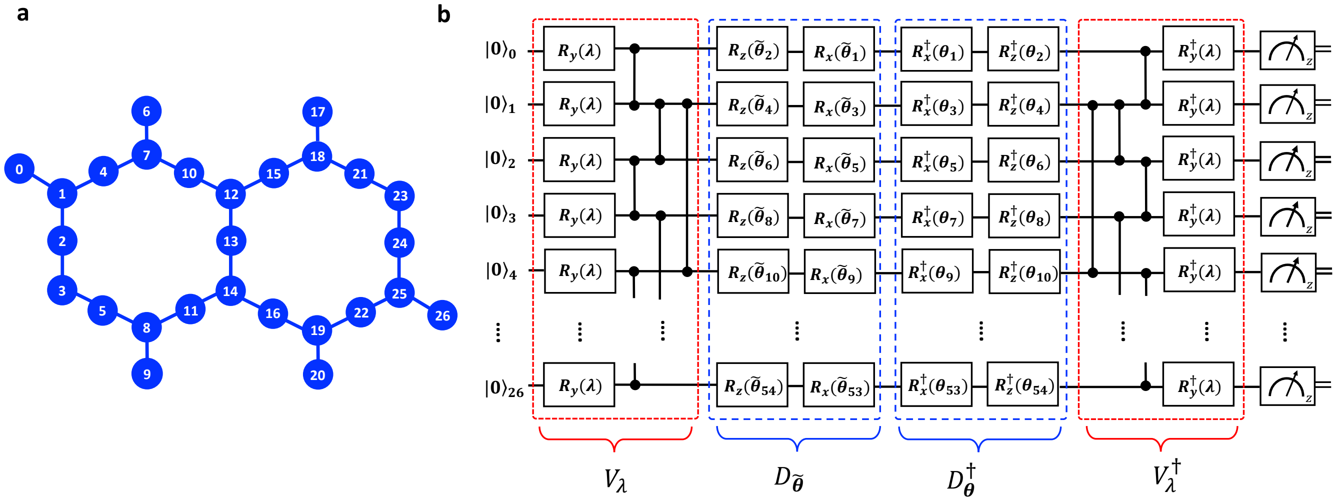

FIG. 2. **Device layout and circuit mapping.** (a) The connectivity of the 27-qubit device *ibmq_kolkata*. (b) The quantum circuit used to evaluate the kernel matrix elements for the learning problem *labeling cosets with errors*. Here, we define the single-qubit rotations as $R_P(\phi) = \exp(-i(\phi/2)P)$ for $P \in \{X, Y, Z\}$. We choose the representation of the group $G = SU(2)^{\otimes 27}$ to be $D_{\boldsymbol{\theta}} = \bigotimes_{k=1}^{27} R_X(\theta_{2k-1}) R_Z(\theta_{2k})$. The fiducial state (5) is prepared by the circuit V_λ for a single parameter λ. The entanglers CZ, which are the controlled phase gates diag$(1, 1, 1, -1)$, match the connectivity of the quantum device in (a). The frequency of measuring all zeros in the computational basis is an estimate for a kernel matrix element.

$\min_{\boldsymbol{\lambda}} \max_{\boldsymbol{\alpha}} F(\boldsymbol{\alpha}, \boldsymbol{\lambda})$, subject to the constraints of the feasible set $0 \leq \alpha_i \leq C$, where C is the box parameter, $\sum_i y_i \alpha_i = 0$, and $\boldsymbol{\lambda} \in \Omega$. In the Supplementary Material (c.f. Appendix D.2), we present a stochastic algorithm for this optimization problem, which is an iterative algorithm with kernel matrices evaluated on a quantum processor and continuous parameters updated with classical optimization routines.

In the quantum experiment we want to benchmark both the accuracy of a learner with access to covariant quantum kernels, as well as the performance of the physical hardware. It is therefore important to have a data set that allows for zero classification error. We therefore construct an artificial benchmark data set. We introduce a learning problem *labeling cosets with error* (LCE) that serves as an abstraction of common learning problems on coset spaces (c.f. Appendix C). Learning problems on coset spaces are frequently considered in the literature [17, 19], for example when considering partial rankings [18] or for manifolds that arise as homogeneous spaces [17]. In LCE we are given a group G and subgroup $S < G$ and generate data from two left-cosets, $c_\pm S \subset G$ determined by representatives $c_+, c_- \in G$. See Fig. 1(a) for a single-qubit example. Every datum taken from cosets is perturbed with a small error ϵ so that the data is not part of the coset any longer. After seeing sufficient data, the learner is asked to classify to which coset a previously unseen datum belongs. We implement LCE with a group G motivated by our physical hardware. We implement our kernel alignment, training, and classification test experiments on a $n = 27$-qubit superconducting quantum processor with heavy-hexagon topology, c.f. Fig. 2(a) [29, 30]. We choose $G = \mathsf{SU}(2)^{\otimes n}$ with the natural representation D of $\mathsf{SU}(2)$ for each qubit, and a Pauli-stabilizer group $S = \langle s_1, \ldots, s_n \rangle$ [31] as subgroup of G. In particular, we choose the graph-stabilizer [32]

$$S_{\text{graph}} = \left\langle \{X_i \bigotimes_{k:(k,i)\in E} Z_k\}_{i \in V} \right\rangle \tag{4}$$

associated with the coupling graph (E, V) given by the device connectivity. To generate the data we represent the rotations $D(\theta_1, \theta_2, \theta_3) \in \mathsf{SU}(2)$ by their Euler angles $D(\theta_1, \theta_2, \theta_3) = \exp(-i(\theta_1/2)X)\exp(-i(\theta_2/2)Z)\exp(-i(\theta_3/2)X)$. For simplicity we set all $\theta_3 = 0$ and randomly draw two $\boldsymbol{c}_\pm \in [-\pi/2, \pi/2]^{2n}$ and define each class by the cosets $C_\pm = \boldsymbol{c}_\pm S_{\text{graph}}$. The rotations with the representative angles $\boldsymbol{c}_\pm$ can be combined with the elements from the stabilizer group so that each datum in the coset is expressed as the rotation $D_{\boldsymbol{\theta}} = \bigotimes_{k=1}^{n} D(\theta_{2k-1}, \theta_{2k}, 0)$. We express every element in the coset in terms of the Euler angles $\boldsymbol{\theta}$. To generate data for training and testing we uniformly sample elements from $C_\pm$ and perturb each set of Euler angles with a normal random error of variance $\epsilon = 0.01$.

After having fixed the unitary representation, we need to choose a fiducial state family. A good a priori choice is a state that is invariant under the sub-group action $D_{\boldsymbol{s}} |\psi\rangle = |\psi\rangle$ for all $\boldsymbol{s} \in S$, illustrated in Fig. 1(b) in the form of a single-qubit example. For S as in (4) this state is given by a stabilizer graph state [32]. Our benchmark for kernel alignment asks to recover this graph state and a perfect classification accuracy from the optimization of

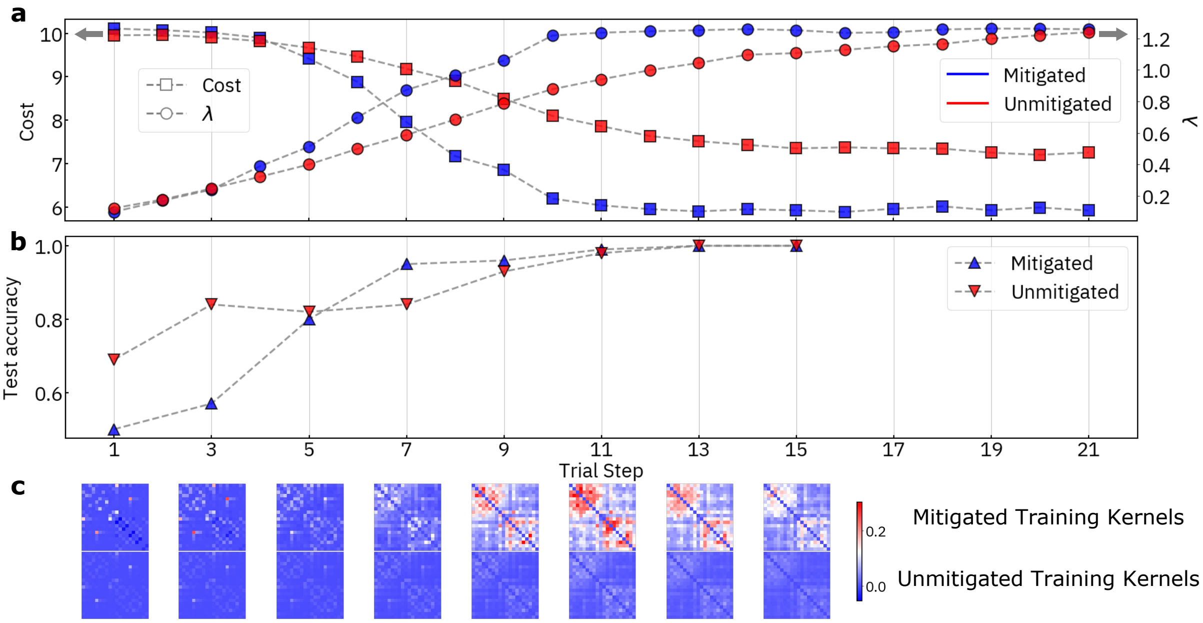

FIG. 3. **Kernel alignment.** (a) Kernel alignment cost function (3) and fiducial state parameter λ for each SPSA trial step, for the error mitigated (blue symbols) and unmitigated (red) experimental realizations. The ideal λ is $\pi/2$. The training data set during this optimization contains 10 data points per label. (b) Accuracy tests results at odd trial steps. For these tests, a training set kernel is estimated with the λ parameter at that step, from which a separating hyperplane is obtained and used to classify a test set consisting of 50 data points per label. The training set kernels used for each classification are shown in (c) for both the mitigated and the unmitigated cases. The identity matrix has been subtracted from these kernels for clarity.

the variational state family

$$|\psi_\lambda\rangle = \prod_{(k,t)\in E} \mathrm{CZ}_{k,t} \prod_{k\in V} R_{Y_k}(\lambda)\,|0^n\rangle\,, \tag{5}$$

where $R_Y(\lambda) = \exp(-i(\lambda/2)Y)$ for a uniform $\lambda \in [0, 2\pi]$. This state interpolates between a trivial initial state $|\psi_0\rangle = |0^n\rangle$ and a graph state stabilized by the subgroup. The resulting covariant quantum kernel, cf. Fig. 2(b), is given as $K_\lambda(\boldsymbol{\theta}, \tilde{\boldsymbol{\theta}}) = |\langle\psi_\lambda| D^\dagger_{\boldsymbol{\theta}} D_{\tilde{\boldsymbol{\theta}}} |\psi_\lambda\rangle|^2$.

We run kernel alignment with and without error mitigation [33–35]. The stochastic optimization algorithm for λ uses Simultaneous Perturbation Stochastic Approximation (SPSA) in both cases. We cover 21 trial steps, starting from $\lambda = 0.1$ in both experimental realizations, using the same training set consisting of 10 data points per label. As we can see in Fig. 3(a), the cost, Eq. (3), flattens after around 11 SPSA trials both for the mitigated an unmitigated cases, with a lower cost for the mitigated experiments. The kernel parameter λ approaches its ideal value of $\pi/2$ at a faster rate in the mitigated case. Once the training is completed, accuracy tests can be run on different sets of data points. We run these classification tests at each odd SPSA step for the mitigated and the unmitigated protocols, targeting a test set with 50 data points per label. As shown in Fig. 3(b), the mitigated experiments reach high classification success percentages faster than the unmitigated version, although both approaches reach 100% success after SPSA step 13. We show the training set Gram matrices at odd steps of SPSA for the mitigated and unmitigated approaches in Fig. 3(c). Each circuit in all these experiments is sampled 8,192 times.

Conclusions: We have identified a promising class of learning problems that stand to profit from the use of quantum kernels. Learning problems with a group theoretic structure have important practical applications [17, 19] and provide a high degree of structure that can be exploited in quantum algorithm design. The quantum kernel method makes it possible to turn a group theoretic learning problem in to a geometric question. Advanced classical kernels for group-data [19] are known to be computationally expensive to evaluate, which is why they are mostly approximated in terms of their lowest Fourier-weights [18, 36]. A potential advantage of quantum kernels can be that such an approximation may not be necessary. The learning problem we have studied experimentally serves as an abstract representation of many group theoretic problems, i.e. classification of data that is close to cosets of a group. As discussed in the manuscript, we find that an important degree of freedom for the covariant quantum kernels is the fiducial state. We expect that, depending on the problem, different choices have to be made. The data for the problem was generated artificially, so that we could ensure that the classification problem could be solved to arbitrary precision and we have a high fidelity benchmark of the experiment. The specific learning problem was constructed with groups that can be implemented naturally and efficiently on current quantum hardware with, to date, the largest circuit width for a quantum kernel. The considered circuit, although supported on a larger number of qubits, is still sufficiently shallow so that classical simulation methods can be applied [37]. These methods exploit the explicit two-dimensional structure, an advantage that disappears for deeper circuits or other coupling topologies.

The core element of the QKE routine reduces to the estimation of an expectation value. If the circuit remains within the coherence limit set by the device, error mitigation techniques [33–35] can be applied. This has been demonstrated in the 27 qubit experiment presented here. The experiment shows that even for a 27 qubit experiment the kernel matrix can be significantly improved by using error mitigation techniques. Note that no readout error mitigation [38] has been applied here, giving rise to the hope that, using threshold functions of the kernels, this approach is indeed robust against small experimental imperfections. For regression problems [39] or other applications where the actual magnitude of the kernel is more important, error mitigation schemes may be increasingly important.

Acknowledgements: We thank Srinivasan Arunachalam and Vojtech Havlicek for helpful comments and discussions and Doug T. McClure and Neereja Sundaresan for technical assistance with the device. This work was supported by the IBM Research Frontiers Institute and K.T. acknowledges support from the ARO Grant W911NF-20-1-0014.

Appendix A: Support vector machines

Although kernels find a wide range of applications throughout the machine learning and statistics literature [2], we focus in the manuscript specifically on the supervised learning learning problem of binary classification. More precisely we consider binary classification with support vector machines (SVM)s. We only review the most elementary components here, in particular in light of the fact that we use a quantum computer to evaluate the kernel entries in the otherwise classical algorithm. In binary classification one considers two data sets (or a partition into two sets) as the training set $T = \{\boldsymbol{x}_i, y_i\}_{i=1\ldots m} \subset \mathcal{X} \times \mathbb{Z}_2$ for a data space $\mathcal{X}$ and the testing set $S \subset \mathcal{X} \times \mathbb{Z}_2$, where typically $S \cap T = \emptyset$. Support vector machines for data that is not linearly separable [8] usually employ a feature mapping to map $\Phi : \mathcal{X} \to \mathbb{R}^N$ from the original data space $\mathcal{X}$ into a higher dimensional Euclidean vector space $\mathbb{R}^N$ with inner product $\langle \cdot, \cdot \rangle_{\mathbb{R}^N}$, often referred to as feature space. The binary SVM classifier is a linear threshold function in this feature space

$$f(\boldsymbol{z}) = \operatorname{sign}\left(\langle w, \Phi(\boldsymbol{z})\rangle_{\mathbb{R}^N} + b\right), \tag{A1}$$

where the vector $w \in \mathbb{R}^N$ is referred to as the hyperplane normal vector that defines the linear threshold function and $b \in \mathbb{R}$ is an offset referred to as a bias. The training of the classifier occurs when optimizing the upper bound of the generalization error [40] in terms of the soft margin of the binary classifier for the training set T. This is commonly expressed in terms of the following quadratic program

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\|w\|_2^2 + C\sum_{i=1}^{m} \xi_i \\ \text{subject to:} \quad & y_i(\langle w, \Phi(\boldsymbol{x}_i)\rangle_{\mathbb{R}^N}) + b \geq 1 - \xi_i \\ & \xi_i \geq 0 \quad \forall i = 1 \ldots m \end{aligned} \tag{A2}$$

The box parameter $C \geq 0$ determines how strongly data points that are not linearly separable contribute to the cost function, where individual contributions for each data point are expressed in terms of the additional variable ξ_i. This is a convex problem and can be solved efficiently in the dimension N of the feature space. However, it is frequently assumed that this dimension is large so that the problem, although convex, becomes intractable. To deal with this problem the kernel method is applied. The kernel function is just the inner product between two feature maps $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \langle \Phi(\boldsymbol{x}_i), \Phi(\boldsymbol{x}_j)\rangle_{\mathbb{R}^N}$. The kernel appears for more general functions by means of the representer theorem [41]. For support vector machines the kernel appears naturally when considering the dual optimization problem to Eq. (A2). The dual problem [8] for the primal problem with $m = |T|$ constraints is again a quadratic program in the Lagrange multipliers α_i for $i = 1 \ldots m$

$$\begin{aligned} \text{maximize} \quad & \sum_{i=1}^{m} \alpha_i - \frac{1}{2}\sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j K(\boldsymbol{x}_i, \boldsymbol{x}_j) \\ \text{subject to:} \quad & \sum_i y_i \alpha_i = 0 \\ & 0 \leq \alpha_i \leq C \quad \forall i = 1 \ldots m \end{aligned} \tag{A3}$$

The optimal dual variables α_i^* are related to the optimal primal normal $w^* = \sum_{i=1}^m y_i \alpha_i^* \Phi(\boldsymbol{x}_i)$ so that the decision function (A1) can be expressed in terms of the kernel as well. For a new datum $\boldsymbol{z} \notin T$ we write the decision function

$$f(\boldsymbol{z}) = \mathrm{sign}\left(\sum_{i=1}^m y_i \alpha_i K(\boldsymbol{x}_i, \boldsymbol{z}) + b\right). \tag{A4}$$

Both training of the SVM as in Eq. (A3) as well as classification (A4) can be performed efficiently in arbitrary spatial dimensions N, even in infinite-dimensional Hilbert spaces, if the kernel function can be evaluated efficiently.

1. Quantum kernels and quantum feature maps

The motivation to introduce quantum kernels in [9] stems from the observation that many of the frequently considered quantum machine learning models, such as variational quantum neural networks [11, 12, 14] can be understood as simple kernel models. This means that if the data mapping in these models leads to states for which the fidelity can be estimated efficiently up to an additive sampling error on a classical computer, conventional support vector machine methods can be applied and it is not possible to obtain a quantum speedup. The main motivation is therefore to consider quantum feature maps that lead to quantum kernel functions that are classically hard to evaluate but can be approximated on a quantum computer. With such a quantum kernel function, the supervised learning algorithm is the direct application of the SVM algorithm as explained in the previous section. The only difference is the use of a quantum computer in estimating the kernel function. Following [9] this kernel function can be understood as the $|0^n\rangle$ to $|0^n\rangle$ transition probability of a particular unitary quantum circuit on n qubits. The non-linear feature mapping for a datum $\boldsymbol{x} \in \mathcal{X}$ occurs through the application of the datum dependent unitary circuit $U(\boldsymbol{x})$ to a reference state $|0^n\rangle \langle 0^n|$. The resulting feature vector is the quantum state of density matrix

$$\Phi(\boldsymbol{x}) = U(\boldsymbol{x}) |0^n\rangle \langle 0^n| U^\dagger(\boldsymbol{x}). \tag{A5}$$

When fitted with conventional trace inner product $\langle A, B\rangle_{tr} = \mathrm{tr}\left[A^\dagger B\right]$ the space of Hermitian matrices becomes a Euclidean vector space. The space that is obtained by the aforementioned construction is a $N = 4^n - 1$ - dimensional Euclidean space that can be mapped directly on to $\mathbb{R}^N$ when a suitable Hermitian matrix basis is chosen. One such example is the basis of Pauli matrices [42]. The features are the components of the feature vectors, relative to this basis and reduce to the expectation values of the Hermitian matrices with respect to the state $\Phi(\boldsymbol{x})$. The kernel function is then the trace inner product between $K(\boldsymbol{x}, \boldsymbol{z}) = \langle \Phi(\boldsymbol{x}), \Phi(\boldsymbol{z})\rangle_{tr}$ and evaluates to the transition amplitude

$$K(\boldsymbol{x}, \boldsymbol{z}) = |\langle 0^n| U^\dagger(\boldsymbol{x}) U(\boldsymbol{z}) |0^n\rangle|^2. \tag{A6}$$

This can be estimated up to an additive error by first preparing the state $|0^n\rangle$ applying the circuit $U^\dagger(\boldsymbol{x})U(\boldsymbol{z})$ and then counting the number of times the all-zero outcome 0^n is observed. Other ways of estimating this kernel are through the use of a SWAP test [9, 43]. This method may become relevant when considering larger noisy devices.

While it may appear to be only a detail, it is actually important to understand the feature state as the density matrix $\Phi(\boldsymbol{x})$. This ensures that the feature space is a Euclidean vector space and it determines the form of the algorithm to access the feature space [9]. Formally using only $|\phi(x)\rangle = U(\boldsymbol{x}) |0^n\rangle$ and the corresponding canonical inner product $\langle \phi(x)|\phi(z)\rangle$ [10] would lead to a kernel and classifiers that are sensitive to the global phase of the state.

Appendix B: Covariant quantum kernels

In this manuscript we introduce a class of quantum kernels suited for learning tasks with data sets that can be seen as a subset of some group $\mathcal{X} \subseteq G$, where the group G is determined by the learning problem at hand. The kernel is computed from a feature mapping that is inspired by covariant quantum measures [21]. To define the feature map in general, we need two components:

1. **Unitary representation:** We define a quantum circuit $D_{\boldsymbol{x}} \in U(2^n)$ that corresponds to a unitary representation $D : \mathcal{X} \to U(2^n)$ [17, 22] of the group G that will act on n qubits. A quantum circuit is assigned for every datum $\boldsymbol{x} \in \mathcal{X}$ and we require that this quantum circuit be implemented efficiently on the quantum computer.

2. **Fiducial state:** Furthermore, we require a reference state $|\psi\rangle \in \left(\mathbb{C}^2\right)^{\otimes n}$ on which the representation can act. This state is referred to as a fiducial state and we assume that it can be prepared efficiently by a quantum circuit V, so that $|\psi\rangle = V|0^n\rangle$.

Two examples will be presented in Secs. B 1 and C. It will become apparent that the choice of fiducial state is important to achieving good classification accuracy. The quantum feature mapping can be defined in two similar ways. One Φ^l, which leads to a *left-invariant* quantum kernel, and another Φ^r, which leads to a *right-invariant* kernel function as will be discussed shortly.

1. *Left-invariant feature state:*

$$\Phi^l : \boldsymbol{x} \to \Phi^l(\boldsymbol{x}) = D_{\boldsymbol{x}} \left|\psi\right\rangle \left\langle\psi\right| D_{\boldsymbol{x}}^\dagger \tag{B1}$$

2. *Right-invariant feature state:*

$$\Phi^r : \boldsymbol{x} \to \Phi^r(\boldsymbol{x}) = D_{\boldsymbol{x}}^\dagger \left|\psi\right\rangle \left\langle\psi\right| D_{\boldsymbol{x}} \tag{B2}$$

In keeping with the notation for quantum feature maps, we identify the quantum feature map circuit as $U(\boldsymbol{x}) = D_{\boldsymbol{x}}V$ and $U(\boldsymbol{x}) = D_{\boldsymbol{x}}^\dagger V$ respectively. This feature mapping can then be used to define the following kernel functions $K^l(\boldsymbol{x},\boldsymbol{z}) = \mathrm{tr}\left[\Phi^l(\boldsymbol{x})\Phi^l(\boldsymbol{z})\right]$ and $K^r(\boldsymbol{x},\boldsymbol{z}) = \mathrm{tr}\left[\Phi^r(\boldsymbol{x})\Phi^r(\boldsymbol{z})\right]$ which are given by

1. *Left-invariant kernel:*

$$K^l(\boldsymbol{x},\boldsymbol{z}) = |\left\langle\psi\right| D_{\boldsymbol{x}}^\dagger D_{\boldsymbol{z}} \left|\psi\right\rangle|^2 = |\left\langle 0^n\right| V^\dagger D_{\boldsymbol{x}}^\dagger D_{\boldsymbol{z}} V \left|0^n\right\rangle|^2. \tag{B3}$$

2. *Right-invariant kernel:*

$$K^r(\boldsymbol{x},\boldsymbol{z}) = |\left\langle\psi\right| D_{\boldsymbol{x}} D_{\boldsymbol{z}}^\dagger \left|\psi\right\rangle|^2 = |\left\langle 0^n\right| V^\dagger D_{\boldsymbol{x}} D_{\boldsymbol{z}}^\dagger V \left|0^n\right\rangle|^2. \tag{B4}$$

The feature maps as well as the kernel functions are referred to as *right-* and *left-invariant* based on their invariance to group multiplication. Consider for any $\boldsymbol{g} \in G$ the functions $K^l(\boldsymbol{g}\boldsymbol{x},\boldsymbol{g}\boldsymbol{z}) = |\left\langle\psi\right| D_{\boldsymbol{x}}^\dagger D_{\boldsymbol{g}}^\dagger D_{\boldsymbol{g}} D_{\boldsymbol{z}} \left|\psi\right\rangle|^2 = |\left\langle\psi\right| D_{\boldsymbol{x}}^\dagger D_{\boldsymbol{z}} \left|\psi\right\rangle|^2 = K^l(\boldsymbol{x},\boldsymbol{z})$ and $K^r(\boldsymbol{x}\boldsymbol{g},\boldsymbol{z}\boldsymbol{g}) = |\left\langle\psi\right| D_{\boldsymbol{x}} D_{\boldsymbol{g}} D_{\boldsymbol{g}}^\dagger D_{\boldsymbol{z}}^\dagger \left|\psi\right\rangle|^2 = |\left\langle\psi\right| D_{\boldsymbol{x}} D_{\boldsymbol{z}}^\dagger \left|\psi\right\rangle|^2 = K^r(\boldsymbol{x},\boldsymbol{z})$ respectively. This means that when $\mathbf{1} \in G$ is the identity element, the kernel functions can always be expressed as $K^r(\boldsymbol{x},\boldsymbol{z}) = K^r(\boldsymbol{x}\boldsymbol{z}^{-1},\mathbf{1})$ and $K^l(\boldsymbol{x},\boldsymbol{z}) = K^l(\boldsymbol{z}^{-1}\boldsymbol{x},\mathbf{1})$ respectively. Hence the kernel function is in either case determined by a single function $f^{r/l}(\boldsymbol{g}) = K^{r/l}(\boldsymbol{g},\mathbf{1})$, and can be recovered from setting either $\boldsymbol{g} = \boldsymbol{z}^{-1}\boldsymbol{x}$ or $\boldsymbol{g} = \boldsymbol{x}\boldsymbol{z}^{-1}$.

For notational simplicity and to be consistent with the main body of the paper, let us only focus on the *left-invariant* kernel functions $f^l(\boldsymbol{g}) = K^l(\boldsymbol{g},\mathbf{1})$. We will from now on simply refer to this function by $l(\boldsymbol{g})$. Recall that a unitary representation D of a group G can always be decomposed in to a direct sum of a set $\mathcal{J}_D$ of irreducible representations D^J by a basis change [17, 22]

$$D_{\boldsymbol{g}} \simeq \bigoplus_{J \in \mathcal{J}_D} D_{\boldsymbol{g}}^J \,. \tag{B5}$$

Given such a decomposition into irreducible representations, general functions on the group, i.e. $f : G \to \mathbb{R}$, can be expressed in terms of the non-Abelian Fourier transform via

$$\hat{f}(J) = \sum_{\boldsymbol{g} \in G} f(\boldsymbol{g}) D_{\boldsymbol{g}}^J. \tag{B6}$$

Recall that $\hat{f}(J) \in M_{\mathrm{d}_J \times \mathrm{d}_J}(\mathbb{C})$ is a matrix-valued function, where the matrix dimension is determined by the dimension of the irreducible representation d_J. The inverse non-Abelian Fourier transform is given by

$$f(\boldsymbol{g}) = \frac{1}{|G|} \sum_J \mathrm{d}_J \mathrm{tr}\left[\hat{f}(J) D^J{}_{\boldsymbol{g}^{-1}}\right]. \tag{B7}$$

Hence, we will be able to represent the covariant quantum kernel $l(\boldsymbol{x})$ in terms of its Fourier coefficients $\hat{l}(J)$. These Fourier coefficients are determined by the fiducial state $\left|\psi\right\rangle$ and the choice of representation $D_{\boldsymbol{x}}$. One can easily verify that

$$K^l(\boldsymbol{x},\mathbf{1}) = |\left\langle\psi\right| D_{\boldsymbol{x}}^\dagger \left|\psi\right\rangle|^2 = \left\langle\psi,\overline{\psi}\right| D_{\boldsymbol{x}^{-1}} \otimes \overline{D}_{\boldsymbol{x}^{-1}} \left|\psi,\overline{\psi}\right\rangle, \tag{B8}$$

where bar indicates complex conjugate. Note that the general tensor product representation $D_{\boldsymbol{x}} \otimes \overline{D}_{\boldsymbol{x}}$ between the original representation and its conjugate can also be decomposed in to irreducible representations and a similar block decomposition exists

$$D_{\boldsymbol{x}} \otimes \overline{D}_{\boldsymbol{x}} \simeq \bigoplus_{J \in \mathcal{J}_{D\overline{D}}} D^J_{\boldsymbol{x}}. \tag{B9}$$

This can be exploited to evaluate the non-Abelian Fourier transform of the quantum kernel as

$$\hat{l}(J) = \frac{|G|}{d_J} \Pi_J \left|\psi, \overline{\psi}\right\rangle \left\langle\psi, \overline{\psi}\right| \Pi_J, \tag{B10}$$

where the projector Π_J is to be read as the projector on the subspace spanned by all the irreducible representations labeled by J. This projector is assumed to act on the space of the representation $D_{\boldsymbol{x}} \otimes \overline{D}_{\boldsymbol{x}}$, which is the originally doubled Hilbert space $\mathbb{C}^{2^{2n}}$ of n qubits and can be constructed by virtue of the characters $\chi_J(g)$ of the irreducible representation J. That is, the projector is computed as

$$\Pi_J = \frac{d_J}{|G|} \sum_{g \in G} \overline{\chi}_J(g)\, D_g \otimes \overline{D}_g. \tag{B11}$$

By inserting the Fourier coefficients, c.f. Eq. (B10) into the inversion formula Eq. (B7), we readily recover the full kernel since we have that $D_{\boldsymbol{x}} \otimes \overline{D}_{\boldsymbol{x}} = \sum_J \Pi_J D^J_{\boldsymbol{x}} \Pi_J$. Note that the Fourier coefficients are given by rank-one projectors induced by the fiducial state $|\psi\rangle$ restricted to the subspace of the irreducible representation J.

1. Example with formal separation

We review an example learning problem for this kernel family introduced in [16] to establish a formal separation between learning with quantum kernels and all other classical learners without access to quantum resources. In this example the group is given by $\mathbb{Z}_p^* = \{1, 2, \ldots, p-1\}$, i.e. the integers with multiplication modulo p. We consider a fixed generator $g \in \mathbb{Z}_p^*$ that generates the full group through modular exponentiation. Given the generator g we will always be able to write any $\boldsymbol{x} = g^{\boldsymbol{v}}$ with $\boldsymbol{v} \in \{0, 1, \ldots, p-1\}$. The inverse of this mapping, i.e. given $\boldsymbol{x}$ compute $\boldsymbol{v} = \mathrm{DLOG}_g(\boldsymbol{x})$ is referred to as the discrete logarithm (DLOG). It is generally assumed that the computation of the discrete logarithm is a computationally hard task for classical computers [23]. That is, any classical algorithm is assumed to scale super-polynomially in $n = \lceil \log_2(p) \rceil$. The learning problem constructed in [16] provides a formal separation between classical and quantum learners with access to quantum kernel functions relative to the hardness assumption of the discrete logarithm problem.

The learning problem is given as follows: To construct a ground truth distribution and labeling rule, draw an element $\boldsymbol{s} \in \mathbb{Z}_p^*$ uniformly at random. Note, there are now $\mathcal{O}(2^n)$ different labeling rules that can be learned. The data to be classified is drawn uniformly at random from $\mathbb{Z}_p^* \subset \{0,1\}^n$. The labeling function then assigns the labels according to the following rule: every sampled element from the training set $\boldsymbol{x} \in T \subset \mathbb{Z}_p^*$ is assigned a label y according to

$$y = \begin{cases} +1, & \text{if } \mathrm{DLOG}_g(\boldsymbol{x}) \in [\boldsymbol{s}, \boldsymbol{s} + \frac{p-3}{2}], \\ -1, & \text{else.} \end{cases} \tag{B12}$$

Here, addition is also taken modulo p. The labeled bit-strings can be generated efficiently classically and are handed to the learner as training and testing sets. It was shown [16] that no classical classifier can assign the correct labels of this decision rule with a probability more than $\frac{1}{2} + \mathrm{poly}(n)^{-1}$, relative to the hardness of the discrete logarithm problem by utilizing a result by Blum and Micali [44]. Conversely it was also proven that near-perfect classification accuracy could be obtained with polynomial effort when the appropriate quantum kernel function was used.

The feature states and kernel for which the quantum advantage has been proven are precisely in the form of the covariant feature map and quantum kernel as discussed in this paper. These feature states are constructed as follows: Let $D_{\boldsymbol{x}}$ denote the regular representation of $\mathbb{Z}_p^* \subset \{0,1\}^n$ defined on n qubits that acts on a computational basis state $|\boldsymbol{z}\rangle \in \mathbb{C}^{2^n}$ through

$$D_{\boldsymbol{x}} \left|\boldsymbol{z}\right\rangle = \left|\boldsymbol{x} \circ \boldsymbol{z} \bmod p\right\rangle. \tag{B13}$$

This group action is implemented by a circuit for modular multiplication as, for example, provided in [45]. We then need to define the fiducial state $|\psi\rangle$. For a generator g we define the following subset state on $\{0,1\}^k \subset \mathbb{Z}_p^*$ as the fiducial state

$$|\psi\rangle = \frac{1}{2^{\frac{k}{2}}} \sum_{\boldsymbol{v}\in\{0,1\}^k} |g^{\boldsymbol{v}}\rangle . \tag{B14}$$

The feature state $\Phi(\boldsymbol{x}) = |\phi(\boldsymbol{x})\rangle\langle\phi(\boldsymbol{x})|$ used in [16] is then obtained as a covariant feature state by writing $|\phi(\boldsymbol{x})\rangle = D_{\boldsymbol{x}}|\psi\rangle = 2^{-\frac{k}{2}} \sum_{\boldsymbol{v}\in\{0,1\}^k} |\boldsymbol{x}g^{\boldsymbol{v}}\rangle$. The kernel is constructed in the canonical form, c.f. Eq. (2).

a. *The importance of the fiducial state choice*

This example highlights the relevance of choosing a good reference state $|\psi\rangle$. To contrast the classification behaviour consider the scenario where one was to modify the kernel and select the computational basis state $|\psi\rangle = |0^n\rangle$ as the fiducial reference state. In this scenario the action of the representation is simply $D_{\boldsymbol{x}}|\psi\rangle = |\boldsymbol{x}\rangle$ and the kernel matrix reduces to

$$K(\boldsymbol{x}, \tilde{\boldsymbol{x}}) = \delta_{\boldsymbol{x},\tilde{\boldsymbol{x}}}. \tag{B15}$$

That is, the kernel becomes the identity matrix. The corresponding distance measure is simply the point-metric that agrees if two data points are identical and is zero if they are not. Apart from the fact that this kernel is trivial to implement classically, it is well known that it will lead to a very poor classifier [8].

This example stresses the importance of the choice of $|\psi\rangle$ and demonstrates that an advantage can only be obtained when a suitable state is chosen. Moreover, the example highlights that the choice of $|\psi\rangle$ may strongly depend on the particular form of the learning problem and the ground truth distribution. This poses a challenge we seek to address in the next section.

Appendix C: The learning problem

Learning problems on data with group structure have a long tradition in statistics [17–19]. In this paper we benchmark the accuracy of the quantum kernel based binary classifier in a quantum computing experiment. The learning problem and data set we construct for the benchmark correspond to the generic classification problem on homogeneous spaces, c.f. [17], Chapter 6. The data problem is chosen to ensure that we can reach zero error. We refer to the particular learning problem considered in the manuscript as labeling cosets with error.

labeling cosets with error: Let us define the general problem for an arbitrary group G and the homogeneous space induced by some subgroup S. We can define two distinct left-cosets for two group elements $\boldsymbol{c}_+, \boldsymbol{c}_- \in G$ by

$$C_+ = \boldsymbol{c}_+ S \qquad \text{and} \qquad C_- = \boldsymbol{c}_- S. \tag{C1}$$

The group elements $\boldsymbol{c}_\pm$ can be drawn at random, for example by the Haar measure of G, to generate different instances of the learning problem. Once two cosets have been determined, the ground truth for the binary classification is as follows: We need two distributions $Q_\pm^\epsilon : G \to \mathbb{R}_0^+$ that have most of their mass on their respective coset $C_\pm$ but can deviate from the coset by a perturbation measured with respect to a small parameter ϵ. To assign a meaning to this deviation it becomes necessary to establish metrics on the group space [17]. We will be working with the matrix norm approach for faithful representations, i.e. $D_{\boldsymbol{x}} \neq D_{\boldsymbol{y}}$ when $\boldsymbol{x} \neq \boldsymbol{y}$, and a unitarily invariant norm $\|\cdot\|$ on the defining space of D. The representation dependent distance is

$$d_D(\boldsymbol{x}, \boldsymbol{y}) = \|D_{\boldsymbol{x}} - D_{\boldsymbol{y}}\| \tag{C2}$$

We ask that the ground truth distribution $Q_\pm^\epsilon(\boldsymbol{x})$, has a constant amount of its mass within an ϵ window $\min_{\boldsymbol{g}\in C_\pm} d_D(\boldsymbol{g}, \boldsymbol{x}) \leq \epsilon$ around each coset.

We will generate a set T of $|T| = m$ data samples for training and testing from such a ground truth distribution by the following steps. For each of the $i = 1 \ldots m$ pick a label $y_i = \pm 1$ and chose the corresponding group element $\boldsymbol{q}_i \in C_\pm$ uniformly at random. Then, we randomly choose a perturbation $\boldsymbol{e}_i \in G$ that is close to the identity element $\mathbf{1}$ so that it satisfies $d_D(\boldsymbol{e}_i, \mathbf{1}) \leq \epsilon$. The distribution of the random perturbation will depend on the group G and the representation D. Each datum $\boldsymbol{x}_i$ is then generated from $D_{\boldsymbol{x}_i} = D_{\boldsymbol{e}_i} D_{\boldsymbol{q}_i}$. This datum will be added to the set T so that the final set we use for training and testing is then $T = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_m, y_m)\}$.

The addition of the perturbation makes this problem different from a purely group-theoretic question. In the error-free limit $\epsilon = 0$ the question amounts to asking whether a given element $\boldsymbol{x}_i$ belongs to a particular coset. Such group theoretic problems are known to be classically hard and have been investigated in the quantum setting to prove formal separations. A particularly relevant example is the subgroup non-membership problem [46]. However, once we add a random perturbation that puts the labeled elements outside of the coset, the problem is no longer purely group theoretic on its own. The problem becomes a geometric learning problem in a Euclidean feature space where the mapping is motivated by group theoretic considerations.

A particularly natural fit for the quantum kernel and fiducial state can be motivated from the error-free setting. The error-free setting motivates a natural candidate for the reference state $|\psi\rangle$. If this state can be prepared efficiently, we would want it to remain invariant under the action of the subgroup S:

$$D_{\boldsymbol{s}} |\psi\rangle = |\psi\rangle \quad \forall\, \boldsymbol{s} \in S. \tag{C3}$$

For such a state, the data samples are mapped to a unique representing state for each coset, since all elements $\boldsymbol{x} \in C_\pm$ are of the form $\boldsymbol{x} = \boldsymbol{c}_\pm \boldsymbol{s}$ with $\boldsymbol{s} \in S$. We therefore have that

$$D_{\boldsymbol{c}_\pm \boldsymbol{s}} |\psi\rangle = D_{\boldsymbol{c}_\pm} D_{\boldsymbol{s}} |\psi\rangle = D_{\boldsymbol{c}_\pm} |\psi\rangle\,. \tag{C4}$$

In this case, the classifier effectively only needs to distinguish between the two states $D_{\boldsymbol{c}_\pm} |\psi\rangle \langle\psi| D^\dagger_{\boldsymbol{c}_\pm}$. This situation changes, however, when we learn cosets with error. The added error is expected to act as a perturbation on each of the two states. If the perturbation is small, e.g. $\epsilon \ll 1$, one would expect that the states will still be classified correctly. Such a covariant quantum feature map turns the initially group theoretic question into a geometric question in quantum feature space and is able to address the perturbations.

The data set for the accuracy benchmark is motivated by the considerations above. In particular, recall to perform the benchmark of the classifier with covariant quantum kernel we need to ensure that we have to work with data that in principle permits an arbitrarily small classification error. For the experiment described in the main section of the paper we choose a natural and easy to implement group. We focus on the group of single-qubit rotations for n qubits, so that $G = SU(2)^{\otimes n}$. A simple subgroup of this group is the Pauli group on n qubits and, by extension, any stabilizer group [47, 48]. We choose the graph-stabilizer of the heavy hex lattice as said subgroup S. This stabilizer group fixes a graph state that follows the chip topology as its invariant state [31].

Appendix D: Quantum kernel alignment algorithm

1. Interpretation of kernel alignment

We use the kernel alignment procedure to chose the best fiducial state in the kernel function from a family of states $|\psi_{\boldsymbol{\lambda}}\rangle = V_{\boldsymbol{\lambda}} |0^n\rangle$. Here, we optimize over different variational quantum circuits $V_{\boldsymbol{\lambda}}$ parametrized by some $V_{\boldsymbol{\lambda}} \in \Omega$. The resulting kernel is then $K_{\boldsymbol{\lambda}}(\boldsymbol{x}, \boldsymbol{z}) = |\langle 0^n| V^\dagger_{\boldsymbol{\lambda}} D^\dagger_{\boldsymbol{x}} D_{\boldsymbol{z}} V_{\boldsymbol{\lambda}} |0^n\rangle|^2$. The kernel alignment procedure for obtaining the optimal hyperplane for data classification solves the problem

$$\min_{\boldsymbol{\lambda}} \max_{\boldsymbol{\alpha}} F(\boldsymbol{\alpha}, \boldsymbol{\lambda}), \tag{D1}$$

subject to the usual constraints for $\boldsymbol{\alpha}$ given in Eq. (A3). The objective we optimize in both the kernel parameters $\boldsymbol{\lambda}$ and the Lagrange multipliers $\boldsymbol{\alpha}$ is

$$F(\boldsymbol{\alpha}, \boldsymbol{\lambda}) = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j K_{\boldsymbol{\lambda}}(\boldsymbol{x}_i, \boldsymbol{x}_j). \tag{D2}$$

This min-max problem has an interesting interpretation as choosing a kernel among all the available $K_{\boldsymbol{\lambda}}$ that minimizes the SVM bound on the classification error. The classification error $\mathbb{P}(y \neq f(\boldsymbol{z}))$ of f is the probability that the classifier fails to predict the correct label y subject to $(\boldsymbol{z}, y)$ being drawn from the true data distribution. This error was upper bounded by Shawe-Taylor and Cristianini [40] in terms of the fat-shattering dimension for linear threshold functions. The fat-shattering bound is data dependent and encodes the generalization error of the classifier. The bound is, up to polylogarithmic factors, given by the primal cost function [8] of the SVM classifier. The optimization of the primal SVM cost function or its Wolfe dual in kernel space, cf. Eq. (A3), considered here can be interpreted as minimizing this upper bound over all admissible threshold functions. For all $\boldsymbol{\lambda}$ the maximum $F^*(\boldsymbol{\lambda}) = \max_{\boldsymbol{\alpha}} F(\boldsymbol{\alpha}, \boldsymbol{\lambda})$ yields the upper bound to the classification error:

$$\mathbb{P}(y \neq f(\boldsymbol{z})) \leq \tilde{\mathcal{O}}(m^{-1} F^*(\boldsymbol{\lambda})). \tag{D3}$$

The notation $\tilde{\mathcal{O}}$ suppresses polylogarithmic factors. The optimization problem in Eq. (D1) thus optimizes the parameters in $K_{\boldsymbol{\lambda}}$ to yield the smallest upper bound on the classification error. The kernel alignment procedure for fiducial states $|\psi_{\boldsymbol{\lambda}}\rangle$ can be interpreted as the search for the fiducial state that gives rise to the best bound on the data-dependent generalization as measured in terms of the fat-shattering dimension for a linear threshold function with the given kernel family.

2. Stochastic algorithm

Our approach to solving the optimization problem with a dataset T as stated in Eq. (D1) is described in Algorithm 1. This iterative classical-quantum algorithm evaluates kernel matrices on a quantum processor and updates the parameters $\boldsymbol{\lambda}$ and $\boldsymbol{\alpha}$ with a classical optimizer. At each iteration, kernel matrices are computed using the QKE routine [9] on quantum circuits parametrized by kernel parameters $\boldsymbol{\lambda}$. The optimal values $\boldsymbol{\lambda}^*$ and $\boldsymbol{\alpha}^*$ obtained can then be used in a standard SVM program to predict labels for a test set.

The parametrization of quantum circuits for the kernel matrices leads, in general, to a highly non-convex objective function (D2) in the kernel parameters. We use simultaneous perturbation stochastic approximation (SPSA) [49] for the minimization over the kernel parameters $\boldsymbol{\lambda}$ in (D1). This method approximates the gradient of (D2) with respect to $\boldsymbol{\lambda}$ using only two objective function evaluations $F(\boldsymbol{\alpha}_\pm, \boldsymbol{\lambda}_\pm)$ independent of the dimension of $\boldsymbol{\lambda}$ and is suitable when measurements of the objective function are subject to stochastic fluctuations, which is the case for kernels evaluated on noisy quantum hardware. For the concave optimization over $\boldsymbol{\alpha}$ in (D1), we use a standard classical solver `CVXOPT` [50], which yields a unique solution for $\boldsymbol{\alpha}$.

Algorithm 1 Quantum Kernel Alignment: learning the maximum-margin kernel

```
Input Training set $T = \{\boldsymbol{x}_i \in \mathbb{R}^n\}_{i=1}^m$ with labels $y \in \{-1, 1\}^m$, quadratic program solver qp (e.g., CVXOPT).
Parameters Number of measurement shots $R$, box constraint $C > 0$ (the SVM regularization parameter), initial kernel parameters $\boldsymbol{\lambda}_0 \in \mathbb{R}^q$, and SPSA steps $P$.
Calibrate the quantum hardware to generate short depth quantum kernel circuits.
Set initial values of the kernel parameters $\boldsymbol{\lambda} = \boldsymbol{\lambda}_0$ for the quantum kernel circuits.
for $i = 0$ to $P$ do
    Generate random vector $\Delta \in \{-1, 1\}^q$.
    Evaluate $\boldsymbol{\lambda}_{\pm,i} = \boldsymbol{\lambda}_i \pm c_i \Delta$, where $c_i = c/(i+1)^\gamma$ for constants $c$, $\gamma$.
    Evaluate the kernel matrices $K_\pm = K(\boldsymbol{\lambda}_{\pm,i}, T)$ on quantum device with $R$ measurement shots per [9].
    Maximize the SVM objective (D2) over $\boldsymbol{\alpha}$ via qp solver $F(\boldsymbol{\alpha}_{\pm,i}, \boldsymbol{\lambda}_{\pm,i}) \leftarrow \mathtt{qp}(K_\pm, y, C)$ subject to $0 \leq \boldsymbol{\alpha}_\pm \leq C$, $y^T \boldsymbol{\alpha}_\pm = 0$.
    Update $\boldsymbol{\lambda}_{i+1} \leftarrow \boldsymbol{\lambda}_i - \frac{a_i}{2c_i} [F(\boldsymbol{\alpha}_{+,i}, \boldsymbol{\lambda}_{+,i}) - F(\boldsymbol{\alpha}_{-,i}, \boldsymbol{\lambda}_{-,i})]$ via SPSA, where $a_i = a/(i+1+A)^\sigma$ for constants $A$, $\sigma$.
end for
return the final kernel parameters $\boldsymbol{\lambda}^*$.
Evaluate aligned kernel matrix $K(\boldsymbol{\lambda}^*, T)$ on the quantum device with $R$ measurement shots per [9].
```

While Algorithm 1 is based on a simple implementation SPSA, other, more sophisticated optimizers can be used [51–53]. However, this choice is suitable to demonstrate the ideas we've presented here on an instance of the learning problem *labelling cosets with errors*. The structural insights we have on samples drawn from the cosets are sufficient to inform a good choice of the parameterized fiducial state. As shown in Fig. 3 from the main text, the parameter converges towards the expected value and the model reaches 100% test accuracy on a 27-qubit problem instance on noisy hardware. In general, it important to have some information about the structure of the learning problem. Such insight can be used to inform the choice of parameterized fiducial state and increase the utility of QKA in practice.

3. Additional considerations

Here, we've considered weighted kernel alignment in our algorithm for learning kernels using the given data. A similar algorithm can be envisioned by replacing the weighted kernel alignment with alternatives like the unweighted kernel alignment [26] and the centered kernel alignment [27, 54, 55]. Such a form of kernel alignment has recently been implemented for quantum kernels by [56]. Optimizing unweighted kernel alignment, $\hat{A}$, between the ideal kernel and the desired kernel has a simple interpretation, which we can see from its definition:

$$\hat{A} \propto \sum_{i,j\in T} K(x_i, x_j) y_i y_j = \sum_{\substack{i\in\{1,2,..,m\} \\ j\in\{j|y_i=y_j, i\neq j\}}} K(x_i, x_j) - \sum_{\substack{i\in\{1,2,..,m\} \\ j\in\{j|y_i\neq y_j\}}} K(x_i, x_j) \tag{D4}$$

where m is the number of training data points in set T. Optimizing the unweighted kernel alignment implies finding kernels that maximize the intra-class overlap and minimize the inter-class overlap for all points included in the training set T. Whereas, weighted kernel alignment Eq. (D1) aims at finding a kernel using the given data that also maximizes the gap margin of the SVM classifier. This means that only the training points that are support vectors contribute towards learning the kernel.

Previous studies suggest that kernel alignment based on kernels that have been normalized and centered in the feature space provide a more useful metric than uncentered kernel alignment [27]. As defined in [27], for a training set with m data points, a centered kernel K_c can be obtained from uncentered kernel K via:

$$K_c = \left[\boldsymbol{I} - \frac{\boldsymbol{1}\boldsymbol{1}^T}{m}\right] K \left[\boldsymbol{I} - \frac{\boldsymbol{1}\boldsymbol{1}^T}{m}\right] \tag{D5}$$

Centered kernel alignment then is defined exactly as the uncentered kernel alignment, using centered kernels instead. In general, the choice of optimizing centered kernel alignment leads to different alignment values when compared to uncentered kernel alignment.

In the light of these choices, further studies are required to assess the performance of weighted and unweighted kernel alignment procedures with or without centered kernels. In addition, geometric differences between kernels and many other techniques developed for learning kernels from data in the classical machine literature may be used [39, 54, 57, 58]. A natural follow up question would be a more exhaustive performance analysis of different kernel learning techniques applied in previous scenarios in the setting of covariant quantum kernels.

The algorithm outlined in this paper, involves optimization of free parameters in a parameterized quantum circuit. Variational methods that optimize over parameterized quantum circuits tend to suffer from the barren plateau problem [59–61] under certain conditions. The problem of barren plateaus is known to worsen when the cost function of the optimization problem relies on measurement of global, as opposed to local, observables [60]. We expect the barren plateau problem to be present in the implementation of this algorithm as the system size increases if the fiducial state is not chosen carefully. This can potentially be prevented by choosing a fiducial state that is well motivated with respect to the structure of the circuit, as well as the initial values of the parameters to be optimized.

Appendix E: Experimental data

1. Device characterization

The device (*ibmq_kolkata*, with the same topology as the device in Ref. [30]), consists of 27 fixed-frequency transmons with fundamental transition frequencies near 5 GHz and anharmonicities around -340 MHz, coupled by co-planar waveguide bus resonators on a top chip die, bump-bonded to a bottom die for readout and signal delivery. Single qubit gates are implemented via microwave pulses with Gaussian envelope with variance σ equal to a quarter of the total pulse length and with DRAG correction [62] and two-qubit gates use the cross-resonance interaction [63] with target rotary pulses [64]. The experiments described in this work take place over four consecutive days, and use a unique calibration for all the gates and readouts. The median qubit lifetime T_1 for the entire device is 132 μs, the median coherence time T_2^{echo} is 148 μs, and the median readout error (defined as $[P(0|1) + P(1|0)]/2$, where $P(i|j)$ represents the probability of measuring i when the state is j) across all 27 qubits is 9.6e-3. During idle times of length T_{idle} for any qubit in our system we use a dynamical decoupling protocol whenever that idle time is longer than twice our single qubit gate. This dynamical decoupling consists of the following sequence: $\tau/2 - X_p - \tau - X_m - \tau/2$ with $\tau = (T_{\text{idle}} - 2T_{X_{p/m}})/2$. We exploit error mitigation techniques [33–35, 65] in our experiments, using stretches $c = 1$ and $c = 1.3$, from which we apply a first-order zero-noise extrapolation to estimate the kernel entries $K_\lambda(\boldsymbol{\theta}, \tilde{\boldsymbol{\theta}})$. For

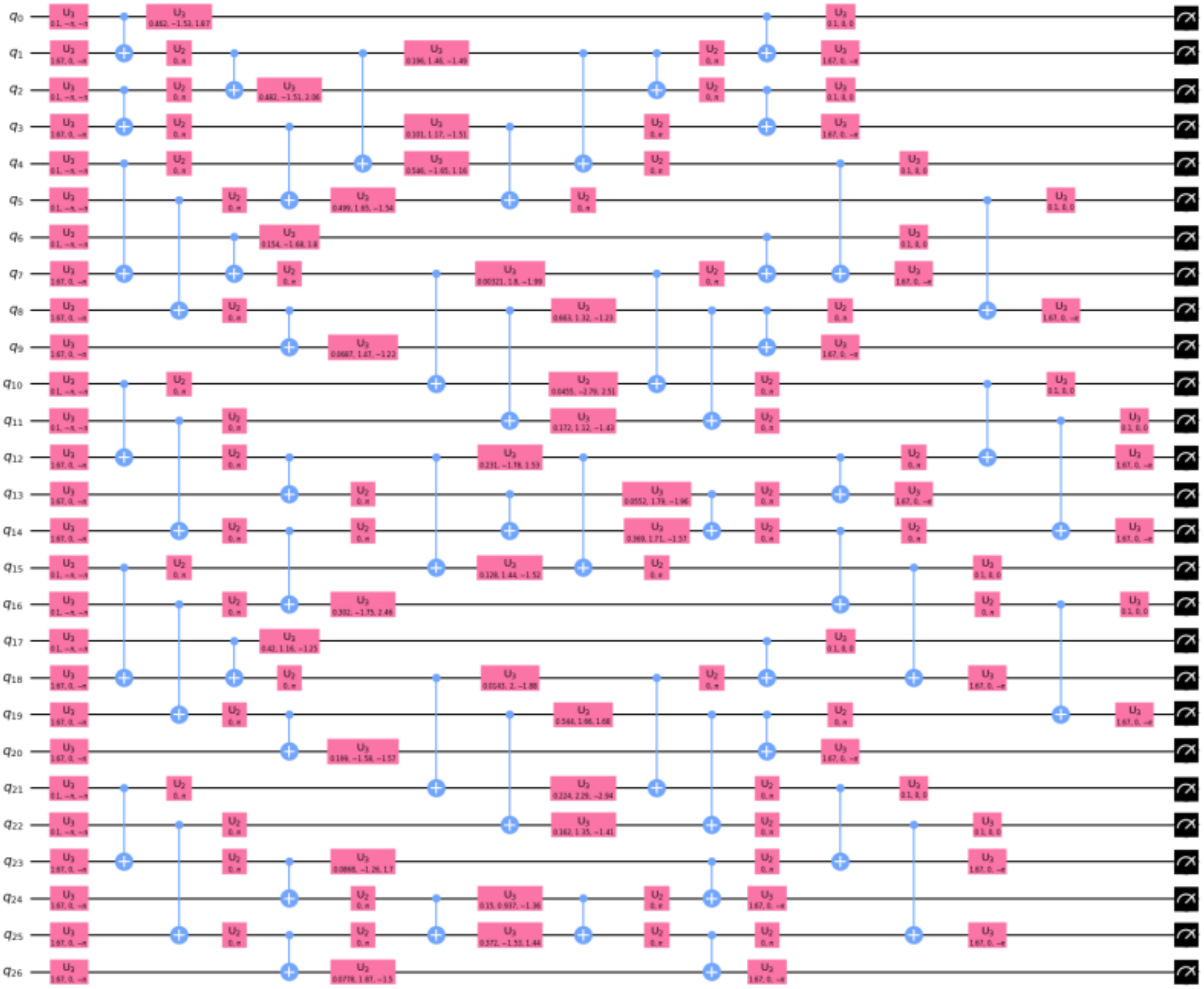

FIG. 4. **Quantum circuit for labeling cosets.** Physical gate representation of the circuit family in Fig. 2(b) for a set of parameters λ, $\boldsymbol{\theta}$, and $\tilde{\boldsymbol{\theta}}$.

stretch $c = 1$ ($c = 1.3$) the single qubit gates are 35.6 (46.2) ns long and the median CNOT length across the device is 476.4 (618.6) ns, with standard deviation of 134 (174) ns. The cross-resonance pulses have Gaussian turn-on and -off envelopes with a variance of 28.4 ns for all the stretches. The median CNOT error across the device is 7.33e-3 (9.27e-3) for the $c = 1$ ($c = 1.3$) stretch and the median single qubit gate error is 2.97e-4 (with negligible difference for both stretches) as measured by randomized benchmarking [66].

Fig. 4 shows a physical representation of one of the circuits used to evaluate kernel entries, as shown in Fig. 2(b) in the main text. The data points are entered via the parameters in the $U_3(\theta, \phi, \lambda) = R_z(\phi)R_y(\theta)R_z(\lambda)$ and $U_2(\phi, \lambda) = U_3(\pi/2, \phi, \lambda)$ unitaries, where $R_y(\alpha) = \exp(-i\alpha Y/2)$ and the equivalent form holds for R_z. The total circuit length is 4.6615 μs, including the 340 ns measurement pulse, for the $c = 1$ stretch, and 5.9580 μs for the $c = 1.3$ stretch. Note that the measurement pulse is not scaled in length when implementing zero-noise extrapolation.

2. Training data

We offer here more insight into the details of the kernel alignment process with the training dataset. Fig. 3 in the main text shows experimental kernels at odd steps between 1 and 15 during the kernel alignment process.

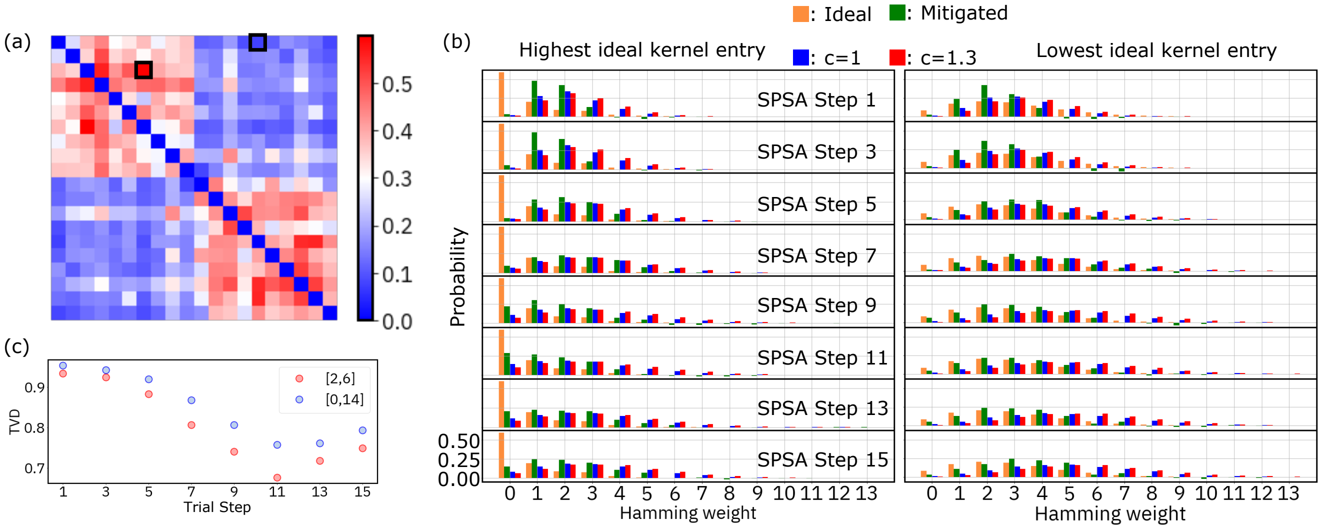

FIG. 5. **Experimental output distributions for training datasets**. (a) Ideal kernel matrix for the training dataset obtained from a noiseless simulation using $\lambda = \pi/2$. The highest (data points 2 and 6) and lowest (data points 0 and 14) entries are highlighted with a black square. (b) Hamming weight distributions of the experimental outputs for the training data points pairs $[2, 6]$ (left) and $[0, 14]$ (right). The yellow bars show the Hamming weight distribution corresponding to the noiseless simulation of those data pairs using $\lambda = \pi/2$. The stretches $c = 1$ (blue bars) and $c = 1.3$ (red bars) are combined to extrapolate to the zero-noise limit (green bars). (c) Total variation distance between the $c = 1$ experimental output distribution for the two kernel entries highlighted in (a) and the corresponding noiseless distribution as a function of SPSA optimization step.

Those kernels capture the overlap between each pair of training data points $(\boldsymbol{\theta}, \tilde{\boldsymbol{\theta}})$ by measuring the observable $K_{\boldsymbol{\lambda}}(\boldsymbol{\theta}, \tilde{\boldsymbol{\theta}}) = |\langle 0^n| V_{\boldsymbol{\lambda}}^{\dagger} D_{\boldsymbol{\theta}}^{\dagger} D_{\tilde{\boldsymbol{\theta}}} V_{\boldsymbol{\lambda}} |0^n\rangle|^2$. Here we look at the output distributions of the quantum computer beyond simply that matrix element for two pairs of data points: the pair $[2, 6]$ and the pair $[0, 14]$. The kernel entries for these two pairs in the ideal case (noiseless simulation for $\lambda = \pi/2$) are marked by black squares in Fig. 5(a). In Fig. 5(b) we show the Hamming weight of the experimental output distributions for both pairs of data points using the mitigated λ parameter at each of the SPSA steps. The bars show both gate stretch factors ($c = 1$ and $c = 1.3$) as well as the error mitigated and ideal (noiseless) expectation values for each Hamming weight. We observe that for both kernel entries, the experimental distributions tend to peak at around Hamming weight 2 and 3. For the training pair $[2, 6]$ (Fig. 5(b) left), however, an increased probability of observing zero Hamming weight progressively develops as the SPSA optimization progresses, peaking at step 11. The pair $[0, 14]$ also shows an increased proximity to the noiseless expectation distribution for the higher SPSA steps measured. Another quantitative approach of the (unmitigated) experimental distribution to the ideal for both pairs of data points can be seen in Fig. 5(c), where the total variation distance $\mathrm{TVD} = 1/2 \sum_i |P_i - Q_i|$ is plotted for both kernel entries as a function of SPSA step.

We can also obtain another view of the convergence of the SPSA optimization by looking at different geometric inspired metrics. By looking at the separation and spread of the training data points in the feature space as the optimization evolves, we can see how the mapping approaches optimal values and how the error mitigation helps with the optimization of the λ parameter. In Fig. 6 we show two such metrics: the Hilbert-Schmidt norm of the distance between the center of mass for each label subset, and the variance of the data points within each label.

For the first metric, we define the center of mass for the positive (negative) label data as $\Phi_+ = \frac{\sum_{i \in T_+} \phi(x_i)}{M}$ $(\Phi_- = \frac{\sum_{i \in T_-} \phi(x_i)}{M})$, and define $HS = ||\Phi_+ - \Phi_-||^2_{\mathrm{HS}}$, where M is the number of data points for each label. The interlabel variance is defined as $\sigma_+ = \frac{\sum_{i \in T_+} ||\phi(x_i) - \Phi_+||^2_{\mathrm{HS}}}{M} = \frac{\sum_{i \in T_+} \sum_{j \in T_+} ||\phi(x_i) - \phi(x_j)||^2_{\mathrm{HS}}}{2M^2}$, with an equivalent definition for the negative label set.

Fig. 6 shows these two metrics, obtained from the experimentally measured kernels, as a function of the trial step. We see that the interlabel variance decreases with increasing trial step, whereas the Hilbert-Schmidt norm of the difference between the center of masses for each label increases with increasing trial step. The ideal values for each metric are shown as gray lines (solid for the center of mass difference, dashed for the interlabel plus set, dotted for the interlabel minus set). The mitigated approach (blue) shows in all cases a measurable advantage versus the non-mitigated data.

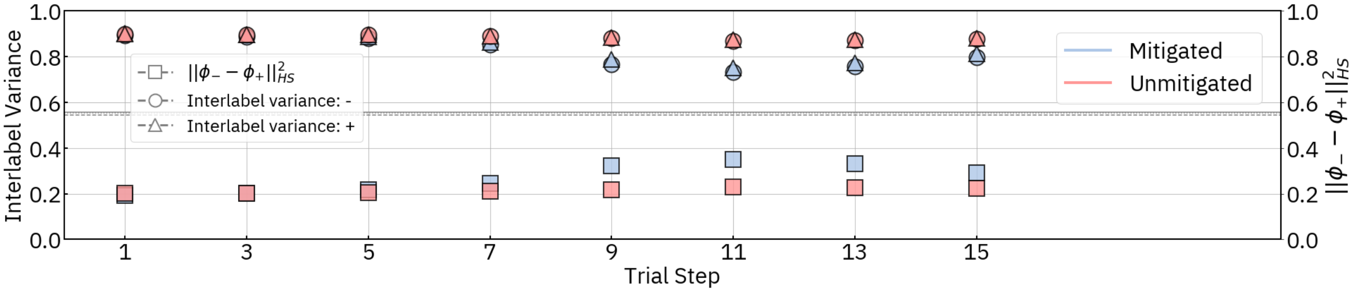

FIG. 6. **Evolution of the mapped training dataset as a function of SPSA trial step**. We look at two metrics related to the data in the feature space as the parameter λ evolves with the SPSA optimization. We see that as the optimization progresses, the Hilbert-Schmidt norm of the difference between the centers of mass for each label increases, showing how the datasets get mapped further from each other (square symbols). We also show the interlabel variance, which shows the overall spread of the datasets for each label in the feature space. The variance for both labels decreases as the optimization progresses. For both metrics the mitigated data does better into the optimization than the unmitigated data. The ideal values for each metric, shown by the gray lines, are limited by the level of noise introduced in our datasets ($\epsilon = 0.01$).

3. Testing data

Similar to the display of the training kernels in the main text, we show in Fig. 7 the Gram matrices for the test data for odd trial steps between 1 and 15. The error mitigated matrices (which also use the error mitigated λ parameter) show a clearer contrast between labels compared to the unmitigated matrices. In this figure, the training data points (10 per label) correspond to the matrices columns and the test data points (50 per label) correspond to the matrices rows.

Beyond simply reporting the classification result of each data point in our test dataset, we can look at the precise value of the decision function in each case, $d(x) = \sum_{i \in SV} y_i \alpha_i K(x_i, x) + b$, where SV is the set of support vectors. This adds valuable information to the classification results and offers interesting insights into the performance of the classifier for the different values of λ and for each particular test data point. We show the decision value for each test data point in Fig. 8(a). The top panel (blue symbols) corresponds to the error mitigated data and the bottom panel (red symbols) corresponds to the unmitigated data. Darker symbol colors in each case correspond to later SPSA trial steps. Misclassified data points are circled for more clarity. We see that the mitigated data show consistently larger decision values at the later stages of the SPSA optimization compared to the unmitigated data. There are some points, however, that remain very close to the decision boundary even deep into the SPSA process, as is the case for data point 42. In Fig. 8(b) we show the overlap between two particular test data points, with indexes 42 and 5, and each of the training data points for the SPSA trial step 11. These are two examples of test data points with reasonably low and high values, respectively, for their decision function. Looking at the mitigated results (green bars) in Fig. 8(b), which represent the corresponding row for each test data point in the mitigated Gram matrix for trial step 11 as shown in Fig. 7, we see that the overlap with the training set for point 42 is quite uniform across the training set, independent of the training label, whereas the overlap for point 5 is remarkably larger with the training points with negative label, which results in an easier classification. Note that, of all the training data points, only index 11 is not a support vector for the kernel corresponding to the mitigated SPSA step 11. Noiseless simulations using the ideal value $\lambda = \pi/2$ (orange bars in Fig. 8(b)) show that the classification of test data point 42 is indeed challenging even in the absence of experimental noise, due to its overlap with support vectors of the opposite label, whereas the same computation for test data point 5 shows this latter point is much easier to classify.

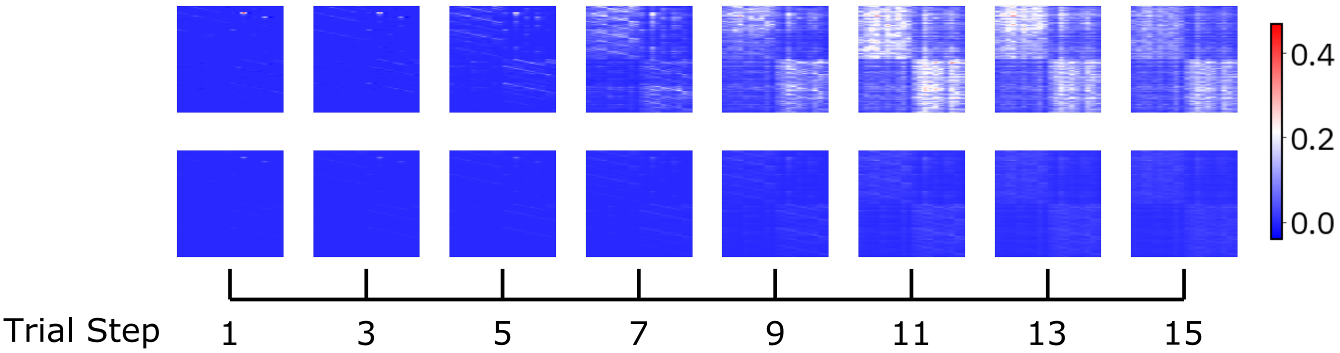

FIG. 7. **Training sets kernels.** Kernel matrices with (upper row) and without (lower row) error mitigation for the test data sets as a function of SPSA step.

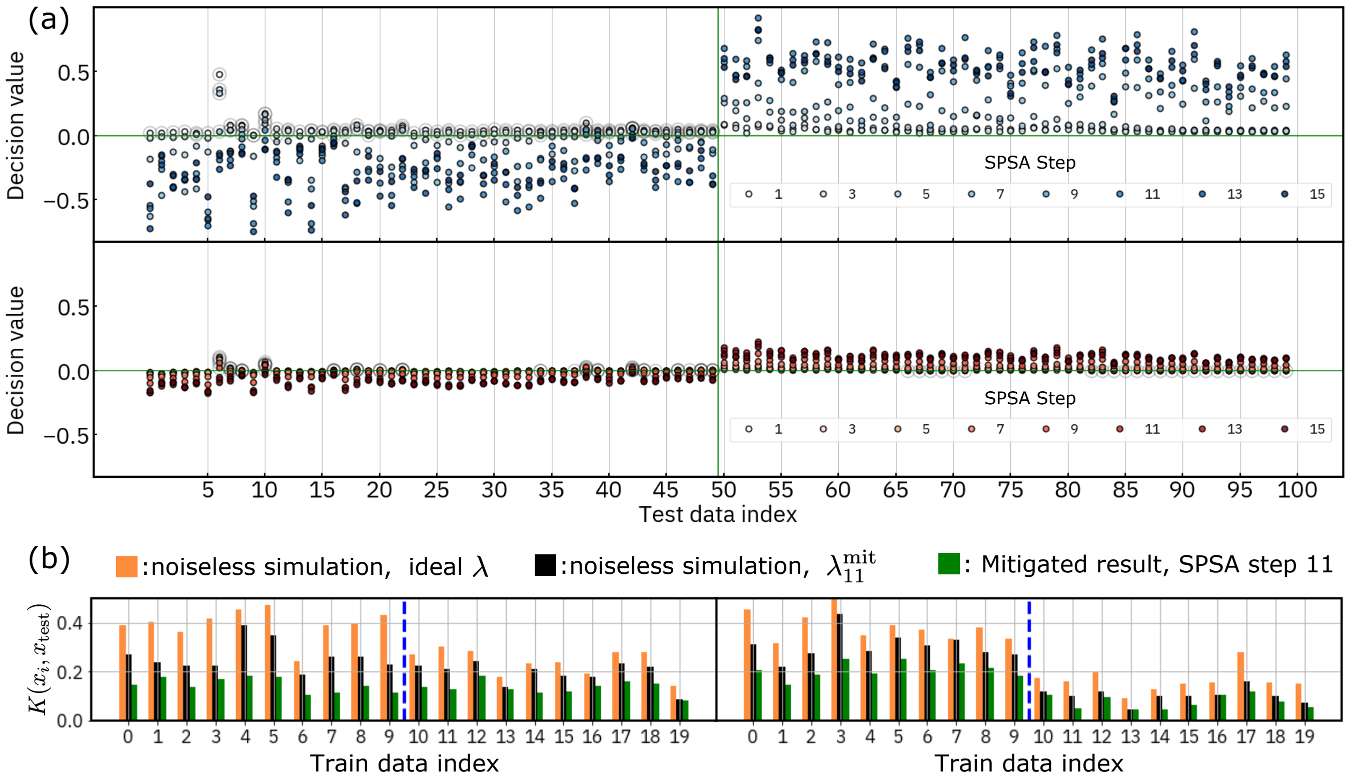

FIG. 8. **A closer look at the classification of the test dataset**.(a) Decision values for each test point classification at each SPSA step for the mitigated (upper panel) and unmitigated (lower panel) approaches. Test data indexes below 50 correspond to negative labels. The increasing margin for the decision values as the SPSA optimization progresses is quite evident for both the mitigated an unmitigated approaches, the former attaining much larger margins as expected. Incorrectly classified data points are highlighted with an outer circle. (b) The overlap of two test data points with negative label (index 42, left; index 5, right) with each of the 20 training data points. Training data point indexes $[0, 9]$ ($[10, 19]$) correspond to negative (positive) label. The dashed blue lines separate the training data points classes. The kernel entries for each training data point are shown for the noiseless simulation for the ideal fiducial state (orange), the noiseless simulation for the experimentally obtained mitigated λ at SPSA step 11 (black), and the error mitigated experimental outcomes at SPSA step 11 (green).

[1] B. E. Boser, I. M. Guyon, and V. N. Vapnik, in *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, COLT '92 (Association for Computing Machinery, New York, NY, USA, 1992) pp. 144–152.

[2] V. Vapnik, S. E. Golowich, and A. Smola, in *Proceedings of the 9th International Conference on Neural Information Processing Systems*, NIPS'96 (MIT Press, Cambridge, MA, USA, 1996) pp. 281–287.

[3] A. J. Smola and B. Schölkopf, Statistics and computing **14**, 199 (2004).

[4] A. Ben-Hur, D. Horn, H. T. Siegelmann, and V. Vapnik, J. Mach. Learn. Res. **2**, 125 (2002).

[5] M. Girolami, IEEE Transactions on Neural Networks **13**, 780 (2002).

[6] P. L. Lai and C. Fyfe, International Journal of Neural Systems **10**, 365 (2000).

[7] W. Liu, J. C. Principe, and S. Haykin, *Kernel adaptive filtering: a comprehensive introduction*, Vol. 57 (John Wiley & Sons, 2011).

[8] V. Vapnik, *The Nature of Statistical Learning Theory*, Information Science and Statistics (Springer New York, 2013).

[9] V. Havlíček, A. D. Córcoles, K. Temme, A. W. Harrow, A. Kandala, J. M. Chow, and J. M. Gambetta, Nature **567**, 209 (2019).

[10] M. Schuld and N. Killoran, Phys. Rev. Lett. **122**, 040504 (2019).

[11] K. Mitarai, M. Negoro, M. Kitagawa, and K. Fujii, Phys. Rev. A **98**, 032309 (2018).

[12] E. Farhi and H. Neven, arXiv:1802.06002 (2018).

[13] E. Grant, M. Benedetti, S. Cao, A. Hallam, J. Lockhart, V. Stojevic, A. G. Green, and S. Severini, npj Quantum Information **4**, 1 (2018).

[14] M. Schuld, A. Bocharov, K. M. Svore, and N. Wiebe, Phys. Rev. A **101**, 032308 (2020).

[15] M. Benedetti, E. Lloyd, S. Sack, and M. Fiorentini, Quantum Science and Technology **4**, 043001 (2019).

[16] Y. Liu, S. Arunachalam, and K. Temme, Nature Physics **17**, 1013 (2021).

[17] P. Diaconis, Lecture notes-monograph series **11**, i (1988).

[18] R. Kondor and M. S. Barbosa, in *COLT* (2010) pp. 451–463.

[19] I. R. Kondor, *Group theoretical methods in machine learning*, Ph.D. thesis (2008).

[20] Y. Jiao and J.-P. Vert, in *International Conference on Machine Learning* (PMLR, 2015) pp. 1935–1944.

[21] A. Holevo, Reports on mathematical physics **16**, 385 (1979).

[22] J.-P. Serre, *Linear representations of finite groups*, Vol. 42 (Springer, 1977).

[23] K. H. Rosen, *Elementary number theory* (Pearson Education London, 2011).

[24] S. Lloyd, M. Schuld, A. Ijaz, J. Izaac, and N. Killoran, arXiv:2001.03622 (2020).

[25] M. Otten, I. R. Goumiri, B. W. Priest, G. F. Chapline, and M. D. Schneider, arXiv:2004.11280 (2020).

[26] N. Cristianini, J. Shawe-Taylor, A. Elisseeff, and J. Kandola, in *Advances in Neural Information Processing Systems 14* (Citeseer, 2001).

[27] C. Cortes, M. Mohri, and A. Rostamizadeh, The Journal of Machine Learning Research **13**, 795 (2012).

[28] B. Bullins, C. Zhang, and Y. Zhang, arXiv:1710.10230 (2017).

[29] C. Chamberland, G. Zhu, T. J. Yoder, J. B. Hertzberg, and A. W. Cross, Phys. Rev. X **10**, 011022 (2020).

[30] P. Jurcevic, A. Javadi-Abhari, L. S. Bishop, I. Lauer, D. F. Bogorin, M. Brink, L. Capelluto, O. Günlük, T. Itoko, N. Kanazawa, *et al.*, Quantum Science and Technology **6**, 025020 (2021).

[31] D. Gottesman, *Stabilizer codes and quantum error correction* (California Institute of Technology, 1997).

[32] M. Hein, W. Dür, J. Eisert, R. Raussendorf, M. Nest, and H.-J. Briegel, quant-ph/0602096 (2006).

[33] K. Temme, S. Bravyi, and J. M. Gambetta, Phys. Rev. Lett. **119**, 180509 (2017).

[34] Y. Li and S. C. Benjamin, Phys. Rev. X **7**, 021050 (2017).

[35] A. Kandala, K. Temme, A. D. Córcoles, A. Mezzacapo, J. M. Chow, and J. M. Gambetta, Nature **567**, 491 (2019).

[36] J. Huang, C. Guestrin, and L. Guibas, Journal of Machine Learning Research **10** (2009).

[37] S. Bravyi, D. Gosset, and R. Movassagh, Nature Physics **17**, 337 (2021).

[38] S. Bravyi, S. Sheldon, A. Kandala, D. C. Mckay, and J. M. Gambetta, Phys. Rev. A **103**, 042605 (2021).

[39] H.-Y. Huang, M. Broughton, M. Mohseni, R. Babbush, S. Boixo, H. Neven, and J. R. McClean, Nature Communications **12** (2021), 10.1038/s41467-021-22539-9.

[40] J. Shawe-Taylor and N. Cristianini, IEEE Transactions on Information Theory **48**, 2721 (2002).

[41] B. Schölkopf, R. Herbrich, and A. J. Smola, in *Computational Learning Theory* (Springer, 2001) pp. 416–426.

[42] M. Nielsen and I. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition* (Cambridge University Press, 2010).

[43] L. Cincio, Y. Subaşı, A. T. Sornborger, and P. J. Coles, New Journal of Physics **20**, 113022 (2018).

[44] M. Blum and S. Micali, SIAM journal on Computing **13**, 850 (1984).

[45] I. L. Markov and M. Saeedi, arXiv:1202.6614 (2012).

[46] J. Watrous, in *Proceedings 41st Annual Symposium on Foundations of Computer Science* (IEEE, 2000) pp. 537–546.

[47] D. Schlingemann and R. F. Werner, Physical Review A **65**, 012308 (2001).

[48] M. Hein, J. Eisert, and H. J. Briegel, Physical Review A **69**, 062311 (2004).

[49] J. Spall, IEEE Transactions on Automatic Control **37**, 332 (1992).

[50] M. Andersen, J. Dahl, and L. Vandenberghe, *CVXOPT: Python Software for Convex Optimization. Version 1.2.6*.

[51] H. Rafique, M. Liu, Q. Lin, and T. Yang, arXiv:1810.02060 (2018).

[52] K. K. Thekumparampil, P. Jain, P. Netrapalli, and S. Oh, in *Advances in Neural Information Processing Systems*, Vol. 32 (Curran Associates, Inc., 2019).

[53] D. Achlioptas, F. McSherry, and B. Schölkopf, Advances in neural information processing systems **14**, 335 (2002).

[54] S.-J. Kim, A. Magnani, and S. Boyd, in *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06 (Association for Computing Machinery, New York, NY, USA, 2006) pp. 465–472.

[55] A. Gretton, O. Bousquet, A. Smola, and B. Schölkopf, in *International conference on algorithmic learning theory* (Springer, 2005) pp. 63–77.

[56] T. Hubregtsen, D. Wierichs, E. Gil-Fuster, P.-J. H. Derks, P. K. Faehrmann, and J. J. Meyer, arXiv:2105.02276 (2021).

[57] N. Srebro and S. Ben-David, in *International Conference on Computational Learning Theory* (Springer, 2006) pp. 169–183.

[58] T. Jebara, Proceedings of the 21st International Conference on Machine Learning (2004), 10.1145/1015330.1015426.

[59] J. Mcclean, S. Boixo, V. Smelyanskiy, R. Babbush, and H. Neven, Nature Communications **9** (2018), 10.1038/s41467-018-07090-4.

[60] M. Cerezo, A. Sone, T. Volkoff, L. Cincio, and P. Coles, arXiv:2001.00550v3 (2020).

[61] C. Marrero, M. Kieferova, and N. Wiebe, PRX Quantum **2** (2021), 10.1103/PRXQuantum.2.040316.

[62] F. Motzoi, J. M. Gambetta, P. Rebentrost, and F. K. Wilhelm, Phys. Rev. Lett. **103**, 110501 (2009).

[63] J. M. Chow, A. D. Córcoles, J. M. Gambetta, C. Rigetti, B. R. Johnson, J. A. Smolin, J. R. Rozen, G. A. Keefe, M. B. Rothwell, M. B. Ketchen, and M. Steffen, Phys. Rev. Lett. **107**, 080502 (2011).

[64] N. Sundaresan, I. Lauer, E. Pritchett, E. Magesan, P. Jurcevic, and J. M. Gambetta, PRX Quantum **1**, 020318 (2020).

[65] Y. Kim, C. J. Wood, T. J. Yoder, S. T. Merkel, J. M. Gambetta, K. Temme, and A. Kandala, arXiv:2108.09197 (2021).

[66] E. Magesan, J. M. Gambetta, and J. Emerson, Phys. Rev. Lett. **106**, 180504 (2011).