Machine learning of high dimensional data on a noisy quantum processor

Evan Peters,[1,2,3,*] João Caldeira,[3] Alan Ho,[4] Stefan Leichenauer,[5] Masoud Mohseni,[4] Hartmut Neven,[4] Panagiotis Spentzouris,[3] Doug Strain,[4] and Gabriel N. Perdue[3]

[1] *Institute for Quantum Computing, University of Waterloo, Waterloo, Ontario, N2L 3G1, Canada*

[2] *Department of Applied Mathematics, University of Waterloo, Waterloo, Ontario, N2L 3G1, Canada*

[3] *Fermi National Accelerator Laboratory, Batavia, IL 60510*

[4] *Google Quantum AI, Venice, CA 90291, United States*

[5] *Sandbox@Alphabet, Mountain View, CA 94043, United States*

(Dated: January 26, 2021)

We present a quantum kernel method for high-dimensional data analysis using Google's universal quantum processor, Sycamore. This method is successfully applied to the cosmological benchmark of supernova classification using real spectral features with no dimensionality reduction and without vanishing kernel elements. Instead of using a synthetic dataset of low dimension or pre-processing the data with a classical machine learning algorithm to reduce the data dimension, this experiment demonstrates that machine learning with real, high dimensional data is possible using a quantum processor; but it requires careful attention to shot statistics and mean kernel element size when constructing a circuit ansatz. Our experiment utilizes 17 qubits to classify 67 dimensional data - significantly higher dimensionality than the largest prior quantum kernel experiments - resulting in classification accuracy that is competitive with noiseless simulation and comparable classical techniques.

Keywords: quantum computing, machine learning, kernel methods

I. INTRODUCTION

Quantum kernel methods (QKM) [1, 2] provide techniques for utilizing a quantum co-processor in a machine learning setting. These methods were recently proven to provide a speedup over classical methods for certain specific input data classes [3]. They have also been used to quantify the computational power of data in quantum machine learning algorithms and drive the conditions under which quantum models will be capable of outperforming classical ones [4]. Prior experimental work [1, 5, 6] has focused on artificial or heavily pre-processed data, hardware implementations involving very few qubits, or circuit connectivity unsuitable for NISQ [7] processors; recent experimental results show potential for many-qubit applications of QKM to high energy physics [8].

In this work, we extend the method of machine learning based on quantum kernel methods up to 17 hardware qubits requiring only nearest-neighbor connectivity. We use this circuit structure to prepare a kernel matrix for a classical support vector machine to learn patterns in 67-dimensional supernova data for which competitive classical classifiers fail to achieve 100% accuracy. To extract useful information from a processor without quantum error correction (QEC), we implement error mitigation techniques specific to the QKM algorithm and experimentally demonstrate the algorithm's robustness to some of the device noise. Additionally, we justify our circuit design based on its ability to produce large kernel magnitudes that can be sampled to high statistical certainty with relatively short experimental runs.

We implement this algorithm on the Google Sycamore processor which we accessed through Google's Quantum Computing Service. This machine is similar to the quantum supremacy demonstration Sycamore chip [9], but with only 23 qubits active. We achieve competitive results on a nontrivial classical dataset, and find intriguing classifier robustness in the face of moderate circuit fidelity. Our results motivate further theoretical work on noisy kernel methods and on techniques for operating on real, high-dimensional data without additional classical pre-processing or dimensionality reduction.

II. QUANTUM KERNEL SUPPORT VECTOR MACHINES

A common task in machine learning is *supervised learning*, wherein an algorithm consumes datum-label pairs $(x, y) \in \mathcal{X} \times \{0, 1\}$ and outputs a function $f : \mathcal{X} \to \{0, 1\}$ that ideally predicts labels for seen (training) input data and generalizes well to unseen (test) data. A popular supervised learning algorithm is the Support Vector Machine (SVM) [10, 11] which is trained on inner products $\langle x_i, x_j \rangle$ in the input space to find a robust linear classification boundary that best separates the data. An important technique for generalizing SVM classifiers to non-linearly separable data is the so-called "kernel trick" which replaces $\langle x_i, x_j \rangle$ in the SVM formulation by a symmetric positive definite kernel function $k(x_i, x_j)$ [12]. Since every kernel function corresponds to an inner product on input data mapped into a feature Hilbert space [13], linear classification boundaries found by an SVM trained on a high-dimensional mapping correspond to complex, non-linear functions in the input space.

This manuscript has been authored by Fermi Research Alliance, LLC under Contract No. DE-AC02-07CH11359 with the U.S. Department of Energy, Office of Science, Office of High Energy Physics.

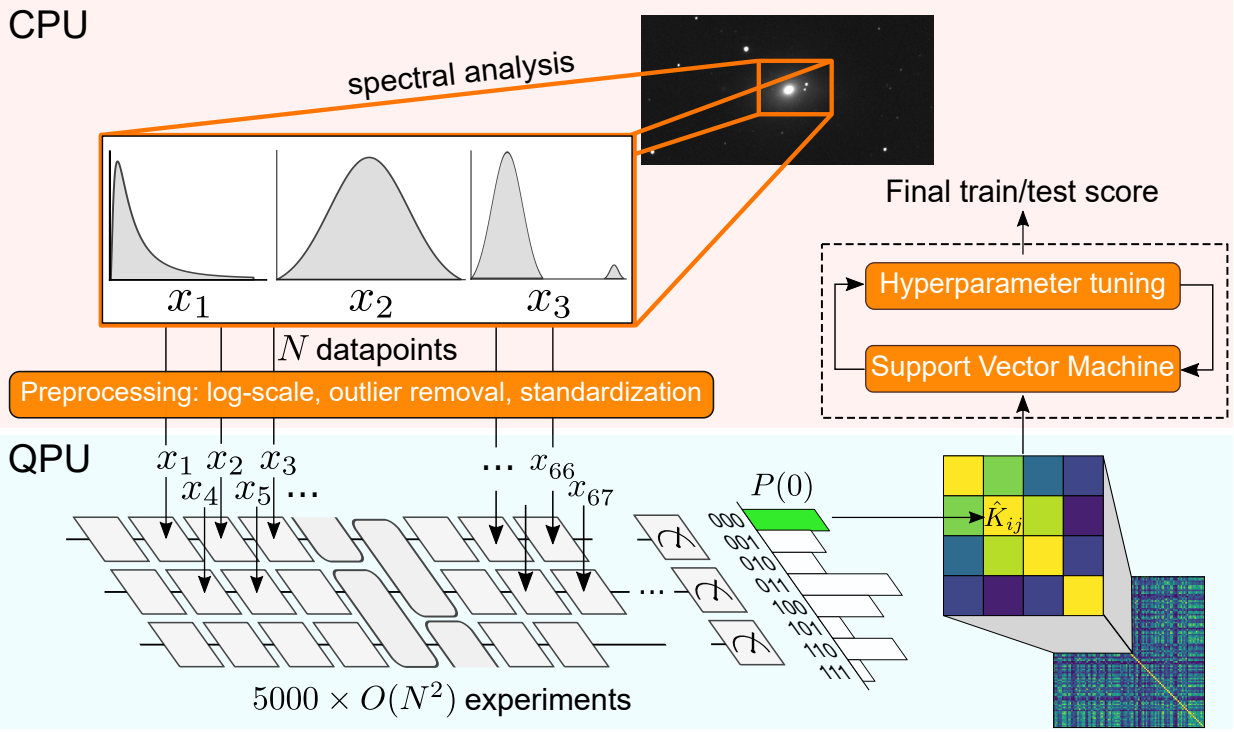


FIG. 1: In this experiment we performed limited data preprocessing that is standard for state-of-the-art classical techniques, before using the quantum processor to estimate the kernel matrix $\hat{K}_{ij}$ for all pairs of encoded datapoints (x_i, x_j) in each dataset. We then passed the kernel matrix back to a classical computer to optimize an SVM using cross validation and hyperparameter tuning before evaluating the SVM to produce a final train/test score.

Quantum kernel methods can potentially improve the performance of classifiers by using a quantum computer to map input data in $\mathcal{X} \subset \mathbb{R}^d$ into a high-dimensional complex Hilbert space, potentially resulting in a kernel function that is expressive and challenging to compute classically. It is difficult to know without sophisticated knowledge of the data generation process whether a given kernel is particularly suited to a dataset, but perhaps families of classically hard kernels may be shown empirically to offer performance improvements. In this work we focus on a non-variational quantum kernel method, which uses a quantum circuit $U(x)$ to map real data into quantum state space according to a map $\phi(x) = U(x)|0\rangle$. The kernel function we employ is then the squared inner product between pairs of mapped input data given by $k(x_i, x_j) = |\langle\phi(x_i)|\phi(x_j)\rangle|^2$, which allows for more expressive models compared to the alternative choice $\langle\phi(x_i)|\phi(x_j)\rangle$ [4].

In the absence of noise, the kernel matrix $K_{ij} = k(x_i, x_j)$ for a fixed dataset can therefore be estimated up to statistical error by using a quantum computer to sample outputs of the circuit $U^\dagger(x_i)U(x_j)$ and then computing the empirical probability of the all-zeros bitstring. However in practice, the kernel matrix $\hat{K}_{ij}$ sampled from the quantum computer may be significantly different from K_{ij} due to device noise and readout error. Once $\hat{K}_{ij}$ is computed for all pairs of input data in the training set, a classical SVM can be trained on the outputs of the quantum computer. An SVM trained on a size-m training set $\mathcal{T} \subset \mathcal{X}$ learns to predict the class $f(x) = \hat{y}$ of an input data point x according to the decision function:

$$f(x) = \text{sign}\left(\sum_{i=1}^{m} \alpha_i y_i k(x_i, x) + b\right) \tag{1}$$

where α_i and b are parameters determined during the training stage of the SVM. Training and evaluating the SVM on $\mathcal{T}$ requires an $m \times m$ kernel matrix, after which each data point z in the testing set $\mathcal{V} \subset \mathcal{X}$ may be classified using an additional m evaluations of $k(x_i, z)$ for $i = 1 \ldots m$. Figure 1 provides a schematic representation of the process used to train an SVM using quantum kernels.

* e6peters@uwaterloo.ca

A. Data and preprocessing

We used the dataset provided in the Photometric LSST Astronomical Time-series Classification Challenge (PLAsTiCC) [14] that simulates observations of the Vera C. Rubin Observatory [15]. The PLAsTiCC data consists of simulated astronomical time series for several different classes of astronomical objects. The time series consist of measurements of flux at six wavelength bands. Here we work on data from the training set of the challenge. To transform the problem into a binary classification problem, we focus on the two most represented classes, 42 and 90, which correspond to types II and Ia supernovae, respectively.

Each time series can have a different number of flux measurements in each of the six wavelength bands. In order to classify different time series using an algorithm with a fixed number of inputs, we transform each time series into the same set of derived quantities. These include: the number of measurements; the minimum, maximum, mean, median, standard deviation, and skew of both flux and flux error; the sum and skew of the ratio between flux and flux error, and of the flux times squared flux ratio; the mean and maximum time between measurements; spectroscopic and photometric redshifts for the host galaxy; the position of each object in the sky; and the first two Fourier coefficients for each band, as well as kurtosis and skewness. In total, this transformation yields a 67-dimensional vector for each object.

To prepare data for the quantum circuit, we convert lognormal-distributed spectral inputs to log scale, and normalize all inputs to $\left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$. We perform no dimensionality reduction. Our data processing pipeline is consistent with the treatment applied to state-of-the-art classical methods. Our classical benchmark is a competitive solution to this problem, although significant additional feature engineering leveraging astrophysics domain knowledge could possibly raise the benchmark score by a few percent.

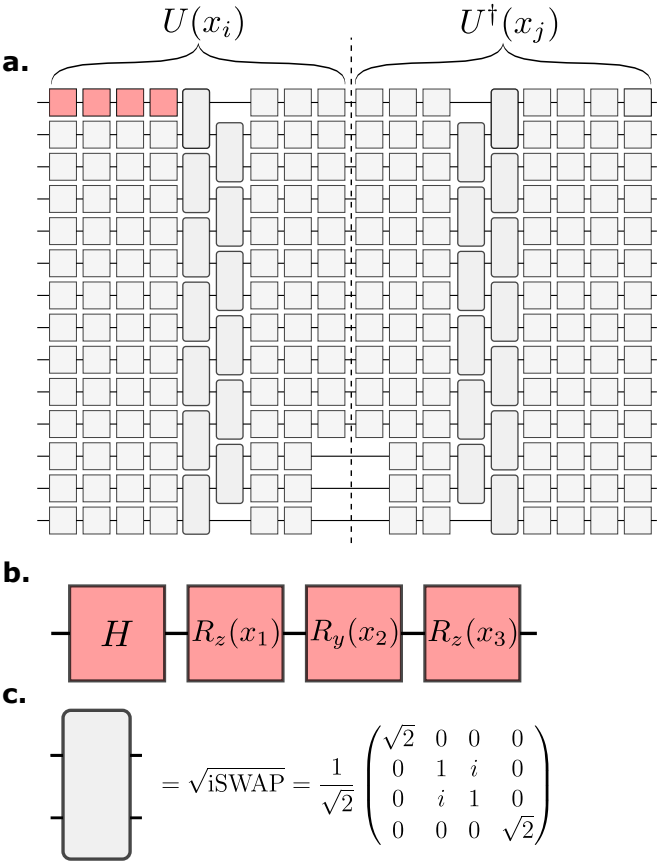


FIG. 2: **a.** 14-qubit example of the type 2 circuit used for experiments in this work. The dashed box indicates $U(x_i)$, while the remainder of the circuit computes $U^\dagger(x_j)$ to ouput $|\langle\phi(x_j)|\phi(x_i)\rangle|^2$. Non-virtual gates occurring at the boundary (dashed line) are contracted for hardware runs. **b.** The basic encoding block consists of a Hadamard followed by three single-qubit rotations, each parameterized by a different element of the input data x (normalization and encoding constants omitted here). **c.** We used the $\sqrt{\text{iSWAP}}$ entangling gate, a hardware-native two-qubit gate on the Sycamore processor.

B. Circuit design

To compute the kernel matrix $K_{ij} \equiv k(x_i, x_j)$ over the fixed dataset we must run R repetitions of each circuit $U^\dagger(x_j)U(x_i)$ to determine the total counts ν_0 of the all zeros bitstring, resulting in an estimator $\hat{K}_{ij} = \frac{\nu_0}{R}$. This introduces a challenge since quantum kernels must also be sampled from hardware with low enough statistical uncertainty to recover a classifier with similar performance to noiseless conditions. Since the likelihood of large relative statistical error between K and $\hat{K}$ grows with decreasing magnitude of $\hat{K}$ and decreasing R, the performance of the hardware-based classifier will degrade when the kernel matrix to be sampled is populated by small entries. Conversely, large kernel magnitudes are a desirable feature for a successful quantum kernel classifier, and a key goal in circuit design is to balance the requirement of large kernel matrix elements with a choice of mapping that is difficult to compute classically. Another significant design challenge is to construct a circuit that separates data according to class without mapping data so far apart as to lose information about class relationships - an effect sometimes referred to as the "curse of dimensionality" in classical machine learning.

For this experiment, we accounted for these design challenges and the need to accommodate high-dimensional data by mapping data into quantum state space using the quantum circuit shown in Figure 2. Each local rotation in the circuit is parameterized by a single element of preprocessed input data so that inner products in the quantum state space correspond to a similarity measure for features in the input space. Importantly,

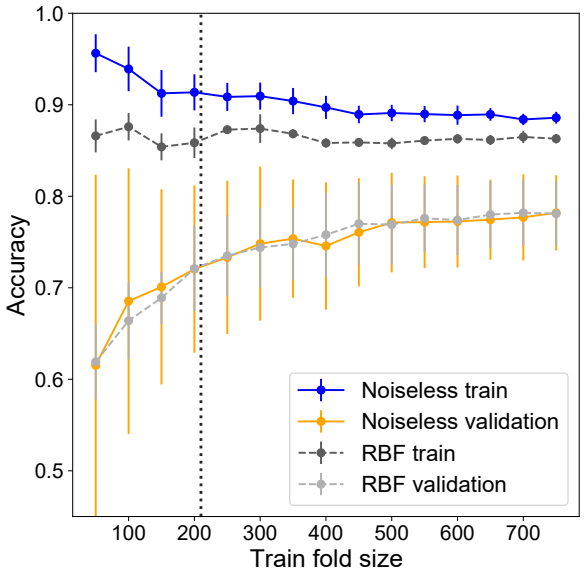


FIG. 3: Learning curve for an SVM trained using noiseless circuit encoding on 17 qubits vs. RBF kernel $k(x_i, x_j) = \exp(-\gamma||x_i - x_j||^2)$. Points reflect train/test accuracy for a classifier trained on a stratified 10-fold split resulting in a size-x balanced subset of preprocessed supernova datapoints. Error bars indicate standard deviation over 10 trials of downsampling, and the dashed line indicates the size $m = 210$ of the training set chosen for this experiment.

the circuit structure is constrained by matching the input data dimensionality to the number of local rotations so that the circuit depth and qubit count individually do not significantly impact the performance of the SVM classifier in a noiseless setting. This circuit structure consistently results in large magnitude inner products (median $K \geq 10^{-1}$) resulting in estimates for $\hat{K}$ with very little statistical error. We provide further empirical evidence justifying our choice of circuit in Appendix IV.

III. HARDWARE CLASSIFICATION RESULTS

A. Dataset selection

We are motivated to minimize the size $\mathcal{T} \subset \mathcal{X}$ since the complexity cost of training an SVM on m datapoints scales as $\mathcal{O}(m^2)$. However too small a training sample will result in poor generalization of the trained model, resulting in low quality class predictions for data in the reserved size-v test set $\mathcal{V}$. We explored this tradeoff by simulating the classifiers for varying train set sizes in Cirq [16] to construct learning curves (Figure 3) standard in machine learning. We found that our simulated 17-qubit classifier applied to 67-dimensional supernova data was competitive compared to a classical SVM trained using the Radial Basis Function (RBF) kernel on identical data subsets. For hardware runs, we constructed train/test datasets for which the mean train and k-fold validation scores achieved approximately the mean performance over randomly downsampled data subsets, accounting for the SVM hyperparameter optimization. The final dataset for each choice of qubits was constructed by producing a 1000×1000 simulated kernel matrix , repeatedly performing 4-fold cross validation on a size-280 subset, and then selecting as the train/test set the exact elements from the fold that resulted in an accuracy closest to the mean validation score over all trials and folds.

B. Hardware classification and Postprocessing

We computed the quantum kernels experimentally using the Google Sycamore processor [9] accessed through Google's Quantum Computing Service. At the time of experiments, the device consisted of 23 superconducting qubits with nearest neighbor (grid) connectivity. The processor supports single-qubit Pauli gates with $> 99\%$ randomized benchmarking fidelity and $\sqrt{i\text{SWAP}}$ native entangling gates with XEB fidelities [9, 17] typically greater than 97%.

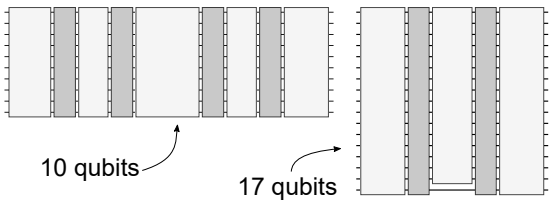
To test our classifier performance on hardware, we trained a quantum kernel SVM using n qubit circuits for $n \in \{10, 14, 17\}$ on $d = 67$ supernova data with balanced class priors using a $m = 210, v = 70$ train/test split. We ran 5000 repetitions per circuit for a total of $m(m-1)/2 + mv \approx 1.83 \times 10^8$ experiments per number of qubits. As described in Section III A, the train and test sets were constructed to provide a faithful representation of classifier accuracy applied to datasets of restricted size. Typically the time cost of computing the decision function (Equation 1) is reduced to some fraction of mv since only a small subset of training inputs are selected as support vectors. However in hardware experiments we observed that a large fraction ($> 90\%$) of data in $\mathcal{T}$ were selected as support vectors, likely due to a combination of a complex decision boundary and noise in the calculation of $\hat{K}$.

Training the SVM classifier in postprocessing required choosing a single hyperparameter C that applies a penalty for misclassification, which can significantly affect the noise robustness of the final classifier. To determine C without overfitting the model, we performed leave-one-out cross validation (LOOCV) on $\mathcal{T}$ to determine C_{opt} corresponding to the maximum mean LOOCV score. We then fixed $C = C_{opt}$ to evaluate the test accuracy $\frac{1}{v}\sum_{j=1}^{v} \Pr(f(x_j) \neq y_j)$ on reserved datapoints taken from $\mathcal{V}$. Figure 4 shows the classifier accuracies for each number of qubits, and demonstrates that the performance of the QKM is not restricted by the number of qubits used. Significantly, the QKM classifier performs reasonably well even when observed bitstring probabil-

a.

Qubits	10	14	17
Depth	35(27)	23(17)	21(15)
$\sqrt{\text{iSWAP}}$ layers	8	4	4
$\sqrt{\text{iSWAP}}$ ct.	40	28	32
1q rot. ct.	254(174)	246(162)	270(170)

b.



c.

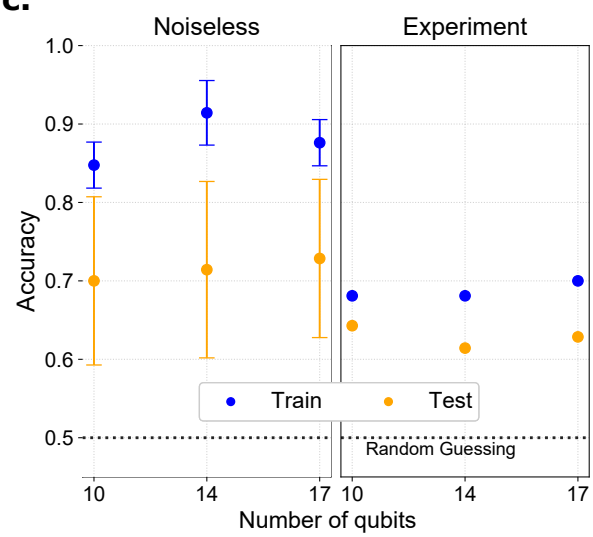


FIG. 4: **a.** Parameters for the three circuits implemented in this experiment. Values in parentheses are calculated ignoring contributions due to virtual Z gates. **b.** The depth of the each circuit and number of entangling layers (dark grey) scales to accommodate all 67 features of the input data, so that the expressive power of the circuit doesn't change significantly across different numbers of qubits. **c.** The test accuracy for hardware QKM is competitive with the noiseless simulations even in the case of relatively low circuit fidelity, across multiple choices of qubit counts. The presence of hardware noise significantly reduces the ability of the model to overfit the data. Error bars on simulated data represent standard deviation of accuracy for an ensemble of SVM classifiers trained on 10 size-m downsampled kernel matrices and tested on size-v downsampled test sets (no replacement). Dataset sampling errors are propagated to the hardware outcomes but lack of larger hardware training/test sets prevents appropriate characterization of of a similar margin of error.

ities (and therefore $\hat{K}_{ij}$) are suppressed by a factor of 50%-70% due to limited circuit fidelity. This is due in part to the fact that the SVM decision function is invariant under scaling transformations $K \to rK$ and highlights the noise robustness of quantum kernel methods.

IV. CONCLUSION AND OUTLOOK

Whether and how quantum computing will contribute to machine learning for real world classical datasets remains to be seen. In this work, we have demonstrated that quantum machine learning at an intermediate scale (10 to 17 qubits) can work on "natural" datasets using Google's superconducting quantum computer. In particular, we presented a novel circuit ansatz capable of processing high-dimensional data from a real-world scientific experiment without dimensionality reduction or significant pre-processing on input data, and without the requirement that the number of qubits matches the data dimensionality. We demonstrated classification results that were competitive with noiseless simulation despite hardware noise and lack of quantum error correction. While the circuits we implemented are not candidates for demonstrating quantum advantage, these findings suggest quantum kernel methods may be capable of achieving high classification accuracy on near-term devices.

Careful attention must be paid to the impact of shot statistics and kernel element magnitudes when evaluating the performance of quantum kernel methods. This work highlights the need for further theoretical investigation under these constraints, as well as motivates further studies in the properties of noisy kernels.

The main open problem is to identify a "natural" data set that could lead to beyond-classical performance for quantum machine learning. We believe that this can be achieved on datasets that demonstrate correlations that are inherently difficult to represent or store on a classical computer, hence inherently difficult or inefficient to learn/infer on a classical computer. This could include quantum data from simulations of quantum many-body systems near a critical point or solving linear and nonlinear systems of equations on a quantum computer [18, 19]. The quantum data could be also generated from quantum sensing and quantum communication applications. The software library TensorFlow Quantum (TFQ) [20] was recently developed to facilitate the exploration of various combinations of data, models, and algorithms for quantum machine learning. Very recently, a quantum advantage has been proposed for some engineered dataset and numerically validated on up to 30 qubits in TFQ using similar quantum kernel methods as described in this experimental demonstration [4]. These developments in quantum machine learning alongside the experimental results of this work suggest the exciting possibility for realizing quantum advantage with quantum machine learning on near term processors.

ACKNOWLEDGMENTS

We would like to thank Google Quantum AI team for time on their Sycamore-chip quantum computer. In particular, the presentation and discussion with Kostyantyn Kechedzhi on error mitigation techniques that was incorporated into this experiment and Ping Yeh's participation in some of the group discussions. Pedram Roushan provided a great deal of useful feedback on early versions of the draft and joined in several useful discussions. We would also like to thank Stavros Efthymiou for some early work on the quantum circuit simulations, and Brian Nord for consultation on interesting datasets in the domain of astrophysics and cosmology.

EP is partially supported through A Kempf's Google Faculty Award. JC, GP, and EP are partially supported by the DOE/HEP QuantISED program grant HEP Machine Learning and Optimization Go Quantum, identification number 0000240323. This manuscript has been authored by Fermi Research Alliance, LLC under Contract No. DE-AC02-07CH11359 with the U.S. Department of Energy, Office of Science, Office of High Energy Physics.

[1] V. Havlícek, 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).

[2] M. Schuld and N. Killoran, Quantum machine learning in feature hilbert spaces, Phys. Rev. Lett. **122**, 040504 (2019).

[3] Y. Liu, S. Arunachalam, and K. Temme, A rigorous and robust quantum speed-up in supervised machine learning (2020), arXiv:2010.02174 [quant-ph].

[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 (2020), arXiv:2011.01938 [quant-ph].

[5] T. Kusumoto, K. Mitarai, K. Fujii, M. Kitagawa, and M. Negoro, Experimental quantum kernel machine learning with nuclear spins in a solid (2019), arXiv:1911.12021 [quant-ph].

[6] 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**, 1 (2020).

[7] J. Preskill, Quantum Computing in the NISQ era and beyond, Quantum **2**, 79 (2018).

[8] S. L. Wu, J. Chan, W. Guan, S. Sun, A. Wang, C. Zhou, M. Livny, F. Carminati, A. D. Meglio, A. C. Y. Li, J. Lykken, P. Spentzouris, S. Y.-C. Chen, S. Yoo, and T.-C. Wei, Application of quantum machine learning using the quantum variational classifier method to high energy physics analysis at the lhc on ibm quantum computer simulator and hardware with 10 qubits (2020), arXiv:2012.11560 [quant-ph].

[9] F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin, R. Barends, R. Biswas, S. Boixo, F. G. S. L. Brandao, D. A. Buell, B. Burkett, Y. Chen, Z. Chen, B. Chiaro, R. Collins, W. Courtney, A. Dunsworth, E. Farhi, B. Foxen, A. Fowler, C. Gidney, M. Giustina, R. Graff, K. Guerin, S. Habegger, M. P. Harrigan, M. J. Hartmann, A. Ho, M. Hoffmann, T. Huang, T. S. Humble, S. V. Isakov, E. Jeffrey, Z. Jiang, D. Kafri, K. Kechedzhi, J. Kelly, P. V. Klimov, S. Knysh, A. Korotkov, F. Kostritsa, D. Landhuis, M. Lindmark, E. Lucero, D. Lyakh, S. Mandrà, J. R. McClean, M. McEwen, A. Megrant, X. Mi, K. Michielsen, M. Mohseni, J. Mutus, O. Naaman, M. Neeley, C. Neill, M. Y. Niu, E. Ostby, A. Petukhov, J. C. Platt, C. Quintana, E. G. Rieffel, P. Roushan, N. C. Rubin, D. Sank, K. J. Satzinger, V. Smelyanskiy, K. J. Sung, M. D. Trevithick, A. Vainsencher, B. Villalonga, T. White, Z. J. Yao, P. Yeh, A. Zalcman, H. Neven, and J. M. Martinis, Quantum supremacy using a programmable superconducting processor, Nature **574**, 505 (2019).

[10] C. Cortes and V. Vapnik, Support-vector networks, Machine learning **20**, 273 (1995).

[11] B. E. Boser, I. M. Guyon, and V. N. Vapnik, A training algorithm for optimal margin classifiers, in *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, COLT '92 (ACM, New York, NY, USA, 1992) pp. 144–152.

[12] M. Aizerman, E. Braverman, and R. Rozoner, Theoretical foundations of potential function method in pattern recognition learning, Automation and Remote Control **6**, 821 (1964).

[13] N. Aronszajn, Theory of reproducing kernels, Transactions of the American mathematical society **68**, 337 (1950).

[14] The PLAsTiCC team, T. A. Jr., A. Bahmanyar, R. Biswas, M. Dai, L. Galbany, R. Hložek, E. E. O. Ishida, S. W. Jha, D. O. Jones, R. Kessler, M. Lochner, A. A. Mahabal, A. I. Malz, K. S. Mandel, J. R. Martínez-Galarza, J. D. McEwen, D. Muthukrishna, G. Narayan, H. Peiris, C. M. Peters, K. Ponder, C. N. Setzer, The LSST Dark Energy Science Collaboration, and The LSST Transients and Variable Stars Science Collaboration, The photometric lsst astronomical time-series classification challenge (plasticc): Data set (2018), arXiv:1810.00001 [astro-ph.IM].

[15] Vera C. Rubin Observatory, `https://www.lsst.org/about` (2020).

[16] Q. A. team and collaborators, Cirq (2020).

[17] C. Neill, P. Roushan, K. Kechedzhi, S. Boixo, S. V. Isakov, V. Smelyanskiy, A. Megrant, B. Chiaro, A. Dunsworth, K. Arya, R. Barends, B. Burkett, Y. Chen, Z. Chen, A. Fowler, B. Foxen, M. Giustina, R. Graff, E. Jeffrey, T. Huang, J. Kelly, P. Klimov, E. Lucero, J. Mutus, M. Neeley, C. Quintana, D. Sank, A. Vainsencher, J. Wenner, T. C. White, H. Neven, and J. M. Martinis, A blueprint for demonstrating quantum supremacy with superconducting qubits, Science **360**, 195 (2018), https://science.sciencemag.org/content/360/6385/195.full.pdf.

[18] B. T. Kiani, G. D. Palma, D. Englund, W. Kaminsky,

M. Marvian, and S. Lloyd, Quantum advantage for differential equation analysis (2020), arXiv:2010.15776 [quant-ph].
[19] S. Lloyd, G. D. Palma, C. Gokler, B. Kiani, Z.-W. Liu, M. Marvian, F. Tennie, and T. Palmer, Quantum algorithm for nonlinear differential equations (2020), arXiv:2011.06571 [quant-ph].
[20] M. Broughton, G. Verdon, T. McCourt, A. J. Martinez, J. H. Yoo, S. V. Isakov, P. Massey, M. Y. Niu, R. Halavati, E. Peters, M. Leib, A. Skolik, M. Streif, D. V. Dollen, J. R. McClean, S. Boixo, D. Bacon, A. K. Ho, H. Neven, and M. Mohseni, Tensorflow quantum: A software framework for quantum machine learning (2020), arXiv:2003.02989 [quant-ph].
[21] C. J. Burges, A tutorial on support vector machines for pattern recognition, Data mining and knowledge discovery **2**, 121 (1998).
[22] J. Shawe-Taylor, N. Cristianini, *et al.*, *Kernel methods for pattern analysis* (Cambridge university press, 2004).
[23] R. Fletcher, *Practical Methods of Optimization.*, Vol. 2 (John Wiley and Sons, Inc., 1987).
[24] H. W. Kuhn and A. W. Tucker, Nonlinear programming, in *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability* (University of California Press, Berkeley, Calif., 1951) pp. 481–492.
[25] M. Goemans, Lecture notes in 18.310a principles of discrete applied mathematics (2015).
[26] E. Farhi and H. Neven, Classification with quantum neural networks on near term processors (2018), arXiv preprint arXiv:1802.06002 (2018).
[27] K. P. F.R.S., Liii. on lines and planes of closest fit to systems of points in space, The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science **2**, 559 (1901), https://doi.org/10.1080/14786440109462720.
[28] T. Tilma and E. Sudarshan, Generalized euler angle parameterization for u(n) with applications to su(n) coset volume measures, Journal of Geometry and Physics **52**, 263 (2004).
[29] A. Kandala, A. Mezzacapo, K. Temme, M. Takita, M. Brink, J. M. Chow, and J. M. Gambetta, Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets, Nature **549**, 242 (2017).
[30] Doug Strain, Private communication (2019).
[31] J. H. Friedman, *Flexible Metric Nearest Neighbor Classification*, Tech. Rep. (1994).
[32] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, Scikit-learn: Machine learning in Python, Journal of Machine Learning Research **12**, 2825 (2011).
[33] R. Sedgewick, *Algorithms in c, Part 5: Graph Algorithms, Third Edition*, 3rd ed. (Addison-Wesley Professional, 2001).
[34] A. A. Hagberg, D. A. Schult, and P. J. Swart, Exploring network structure, dynamics, and function using NetworkX, 7th Python in Science Conference (SciPy 2008) , 11 (2008).
[35] J. Bi and T. Zhang, Support vector classification with input data uncertainty, in *Advances in neural information processing systems* (2005) pp. 161–168.

Appendix A: Binary classification with Support Vector Machines

Supervised learning algorithms are tasked with the following problem: Given input data $\mathcal{X} \subset \mathbb{R}^d$ composed of d-dimensional datapoints and the corresponding class labels taken from $\mathcal{Y} = \{-1, 1\}$ attached to each datapoint, construct a function f that can successfully predict $f(x_i) = y_i$ given a datapoint-label pair taken from the dataset $(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}$.

We now introduce the theoretical foundations for the Support Vector Machine, or SVM (see [21, 22] for a thorough review). A linear SVM performs binary classification by constructing a $(d-1)$ dimensional hyperplane $\langle x, w\rangle + b$ that divides elements of $\mathcal{X}$ according to their class, by finding the hyperplane with the largest perpendicular distance ("margin") to elements of either class. The hyperplane parameters capable of classifying linearly separable data must satisfy the inequality

$$y_i \left(\langle x_i, w\rangle + b\right) \geq 1 \tag{A1}$$

which corresponds to a symmetric margin of $2/||w||$ dividing classes of linearly separable data. Maximizing the margin therefore corresponds to minimizing the hyperplane normal vector, so the task of the SVM is to find the solution to the convex problem

$$\min_{w,b} \frac{1}{2}||w||^2 \tag{A2}$$

Equations A1-A2 frame a constrained optimization problem that can be solved by method of Lagrange multipliers. The Lagrangian to minimize is then

$$L_P = \frac{1}{2}||w||^2 - \sum_{i=1} \alpha_i y_i (\langle x_i, w\rangle + b) + \sum_{i=1} \alpha_i \tag{A3}$$

Recognizing that the inequality of Equation A1 can only be satisfied by fully separable data, SVM classifiers trained on real data typically employ so-called "slack" variables that loosen the classification constraints and introduce a misclassification penalty to the Lagrangian formulation [10]. The constraints for training a linear SVM on non-separable data using slack variables ξ_i then takes on the form:

$$y_i \left(\sum_{s \in SV} \alpha_s y_s k(x_i, x_s) + b \right) \geq 1 - \xi_i \tag{A4}$$

$$\xi_i \geq 0 \quad \forall i \tag{A5}$$

where we choose to assign an L2 penalty for misclassification by adding an additional cost term to the objective function A2, resulting in the modified objective function

$$\min_{w,b} \frac{1}{2}||w||^2 + \frac{C}{2} \sum_i \xi_i^2 \tag{A6}$$

Applying the method of Lagrange multipliers, the primal Lagrangian for Equation A6 is

$$L_P = \frac{1}{2}||w||^2 - \sum_{i=1} \alpha_i (y_i(\langle x_i, w \rangle + b) - 1 + \xi_i) - \sum_i \mu_i \xi_i + \frac{C}{2} \sum_i \xi_i^2 \tag{A7}$$

Alternatively this can be reformulated to the Wolfe dual problem [23], with the goal of *maximizing* the dual Lagrangian

$$L_D = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle - \frac{C}{2} \sum_i \xi_i^2 \tag{A8}$$

subject to the constraints:

$$0 \leq \alpha_i \leq C \tag{A9}$$

$$\sum_i \alpha_i y_i = 0 \tag{A10}$$

As Equation A6 is a convex programming problem, the solutions to the primal Lagrangian of Equation A7 and the dual Lagrangian of Equation A8 are subject to the Karush-Kuhn-Tucker (KKT) conditions [24]:

$$w - \sum_i \alpha_i y_i x_i = 0 \tag{A11}$$

$$\sum_i \alpha_i y_i = 0 \tag{A12}$$

$$C - \alpha_i - \mu_i = 0 \tag{A13}$$

$$\alpha_i (y_i(\langle x_i, w \rangle + b) - 1 + \xi_i) = 0 \tag{A14}$$

$$y_i(\langle x_i, w \rangle + b) - 1 + \xi_i \geq 0 \tag{A15}$$

$$\mu_i \xi_i = 0 \tag{A16}$$

$$C \geq \alpha_i \geq 0 \tag{A17}$$

$$\mu_i \geq 0 \tag{A18}$$

$$\xi_i \geq 0 \tag{A19}$$

These conditions determine the hyperplane intercept b and also describe the geometry of the maximal margin hyperplane for the trained SVM. Once the optimal set of parameters $\vec{\alpha}$ is determined with respect to the training inputs $\mathcal{X}$, the linear SVM predicts the class of a data point using the decision function

$$f(x_p) = \sum_{s \in SV} \alpha_s y_s \langle x_p, x_s \rangle + b \tag{A20}$$

where the sum runs over the indices of the support vectors, or equivalently all nonzero α_i.

Equation A20 and the dual Lagrangian A8 no longer explicitly reference elements of the input space $w, x_i \in \mathbb{R}$, so the optimization problem is still valid under the substitution $x \to \phi(x)$ for some mapping $\phi : \mathcal{X} \to \mathcal{H}$ where $\mathcal{H}$ is a Hilbert space (this is the so-called "kernel trick" [12]). This permits us to embed input data into higher dimensional Hilbert space for some choice of ϕ and then train the SVM on inner products in the mapped space, $\langle \phi(x_i), \phi(x_j) \rangle_{\mathcal{H}} \equiv k(x_i, x_j)$, where the status of k as an inner product guarantees that it is symmetric, positive-definite. The resulting SVM will then be capable of constructing decision boundaries that are nonlinear in the input space $\mathbb{R}^d$ resulting in a decision function

$$f(x_p) = \sum_{s \in SV} \alpha_s y_s k(x_p, x_s) + b \tag{A21}$$

Evaluating $k(x_p, x_s) = K_{ps}$ for a fixed set $x_p, x_s \in \mathcal{T} \cup \mathcal{V}$ recovers Equation 1 from the main body (since $\alpha_i = 0$ for i outside the support vector set by Equations A13 and A16).

Appendix B: Circuit structure and Hilbert space embedding

1. Statistical uncertainty and vanishing kernels

We now discuss limitations to hardware-based quantum kernel methods due to statistical uncertainty. Recall that each kernel matrix element $K_{ij} = k(x_i, x_j)$ is computed by sampling the output of a circuit $U^\dagger(x_j)U(x_i)$ for a total of R repetitions and counting the number ν_0 of all-zeros bitstrings that appear. This experiment constitutes R trials of a Bernoulli process parameterized by K_{ij}; the unbiased estimators for K_{ij} and associated variance are therefore given by:

$$\hat{K}_{ij} = \frac{\nu_0}{R} \tag{B1}$$

$$\mathrm{Var}(\hat{K}_{ij}) = \frac{\hat{K}_{ij}(1 - \hat{K}_{ij})}{R - 1} \tag{B2}$$

Note that positive definiteness of $\hat{K}$ is not necessarily preserved in the presence of statistical error and hardware noise, but in practice we found this had little effect on the ability of the SVM to classify data. The $O(R^{-1/2})$ sampling error of Equation B2 combined with a requirement that $||\hat{K} - K||_F = \left(\sum_{ij} |K_{ij} - \hat{K}_{ij}|\right)^{1/2} \leq \epsilon m$ (where $m = |\mathcal{X}|$) would suggest that an ϵ-close estimation of K could be achieved using $R = O(\epsilon^{-2}N^2)$ shots per kernel element. In the main body we argue that this fails to bound *relative* error between kernel matrices. This is evident in the symmetric Chernoff bound for the relative error of a sampled $\hat{K}_{ij}$ [25]:

$$P\left(\frac{|\hat{K} - K|}{K} \geq \varepsilon\right) \leq 2e^{-RK\varepsilon^2/3} \tag{B3}$$

for which the probablity of large relative error quickly becomes unbounded for $R << \mathcal{O}(K_{ij}^{-1})$. Relative error is relevant by the following reasoning: Let L'_D be the L1-penalized dual Lagrangian corresponding to a kernel constructed from the transformation $K' = rK$, given by

$$L'_D = \sum_i \alpha'_i - \frac{1}{2} \sum_{i,j} \alpha'_i \alpha'_j y_i y_j K'_{ij} \tag{B4}$$

If α_{opt} contains the parameters maximizing the Lagrangian

$$L_D = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j K_{ij} \tag{B5}$$

then α_{opt} also maximizes the Lagrangian $\frac{1}{r} L_D$ which may be rewritten as follows:

$$\frac{1}{r}L_D = \sum_i \frac{\alpha_i}{r} - \frac{1}{2}\sum_{i,j}\frac{\alpha_i\alpha_j}{r^2}y_i y_j (rK_{ij}) \tag{B6}$$

By comparison with Equation B4 L'_D has the immediate choice of unique solution $\alpha'_{opt} = \alpha_{opt}/r$, which may be achieved by appropriate choice of penalty parameter C appearing in the KKT conditions A11-A19 for L'_D (or equivalently in the primal problem L'_P before kernelizing the problem). A similar result can be attained if an L2 misclassification penalty is used by restricting the Lagrange multipliers α. Consequently the decision function for an SVM classifier trained on the kernel matrix K' is identical to the decision function for the SVM classifier trained on the original K. This result makes intuitive sense for the choice of a linear kernel $K_{ij} = \langle x_i, x_j \rangle$, for which stretching/shrinking each datapoint $x_i \to x_i/\sqrt{r}$ has no effect on the geometry of the maximal margin hyperplane. By choosing some r for the transformation $K \to rK$, $\hat{K} \to r\hat{K}$, the absolute error $||K - \hat{K}||_F$ may be made arbitrarily large or small without affecting the performance of the associated SVM classifier, which suggests that $||K - \hat{K}||_F$ does not completely characterize the resulting SVM accuracy.

The high dimensionality of quantum state space poses a threat to the experimental feasibility of quantum kernel methods since the relative statistical error incurred by finite shot statistics grows as the magnitude of the sampled kernel element shrinks. Using a naive example, a randomly selected encoding unitary will almost certainly result in vanishing kernel elements by strong measure concentration: If we treat $S = \{U^\dagger(x_j)U(x_i)\}$ as a distribution of unitaries that are random with respect to the Haar measure it is well known that the expected probability for any specific bitstring (and therefore $\frac{\nu_0}{R}$) sampled from a unitary in S scales as $\mathcal{O}(2^{-n})$. In practice, the preprocessing of input data and circuit structure must be chosen with careful attention given to the corresponding distribution on K.

To explore this effect we constructed a classifier similar to quantum circuits described in [1, 26], depicted in Figure 5b and given by

$$\begin{aligned} U(x) &= H^{\otimes n}V(x)H^{\otimes n}V(x) \\ V(x) &= \exp\left(\sum_{i=1}^{n} c_1 x_i Z_i + \sum_{(i,j)\in NN} c_2(x_i - x_j)Z_i Z_j\right) \end{aligned} \tag{B7}$$

where the entangling gates are selected among nearest neighbors on the (simulated) circuit grid. We chose to parameterize entanglers by $c_2(x_i - x_j)$ instead of quadratic terms proportional to $x_i x_j$ to eliminate concentration of $E[x_i x_j] \to 0$ that occurs for our choice of normalization. For clarity we refer to the circuit described by Equation B7 as "Type 1". An increase in the connectivity results in a higher gate/parameter count, resulting in generally smaller sampled kernel elements. This circuit structure requires that input data have dimension equal to the number of qubits. We applied Principal Component Analysis (PCA) [27] to reduce the 67-dimensional data to n dimensions and then standardized the data to the interval $[-\pi/2, \pi/2]$. We defined additional hyperparameters (c_1, c_2) that can be tuned to optimize the cross-validated performance of the corresponding SVM and control the resulting distribution of kernel matrix elements. Figure 5a shows that the magnitude of K vanishes with respect to increasing c_1, c_2 or number of qubits n. This does not necessarily result low accuracies for the associated SVM classifiers, but describes a family of kernels that are infeasible to sample on hardware. It is possible to preserve the magnitude of K if c_1 and c_2 are scaled down with increasing n but for small enough angles over/under-rotation errors and noise will become dominating factors in hardware outcomes, while the limit $c_2 \to 0$ results in a circuit that can be simulated trivially.

These results motivate a new approach for encoding data on large numbers of qubits, especially if the input data dimensionality is large. To compute large-magnitude kernels on high dimensional data without dimensionality reduction, we designed a circuit encoding to map input data $x_i, z_i \in X \subset \mathbb{R}^d$ into a subspace of $\mathbb{C}^{2^n}$ using an approximately orthogonal parameterization of $U(2^n)$, the group describing n-qubit unitaries. While examples of exactly orthogonal parameterizations of $U(2^n)$ exist, such as Euler angle parameterization of $U(2^n)$ [28], such schemes are generally inefficient to implement on hardware. We approximate such an encoding using circuits structured similarly to the Hardware Efficient Ansatz [29] consisting of an initial layer parameterizing $\bigotimes^n U(2)$ interspersed with local entanglers. This circuit structure (referred to here as "Type 2") is shown in Figure 2 of the main body and can be expressed in terms of individual gates as

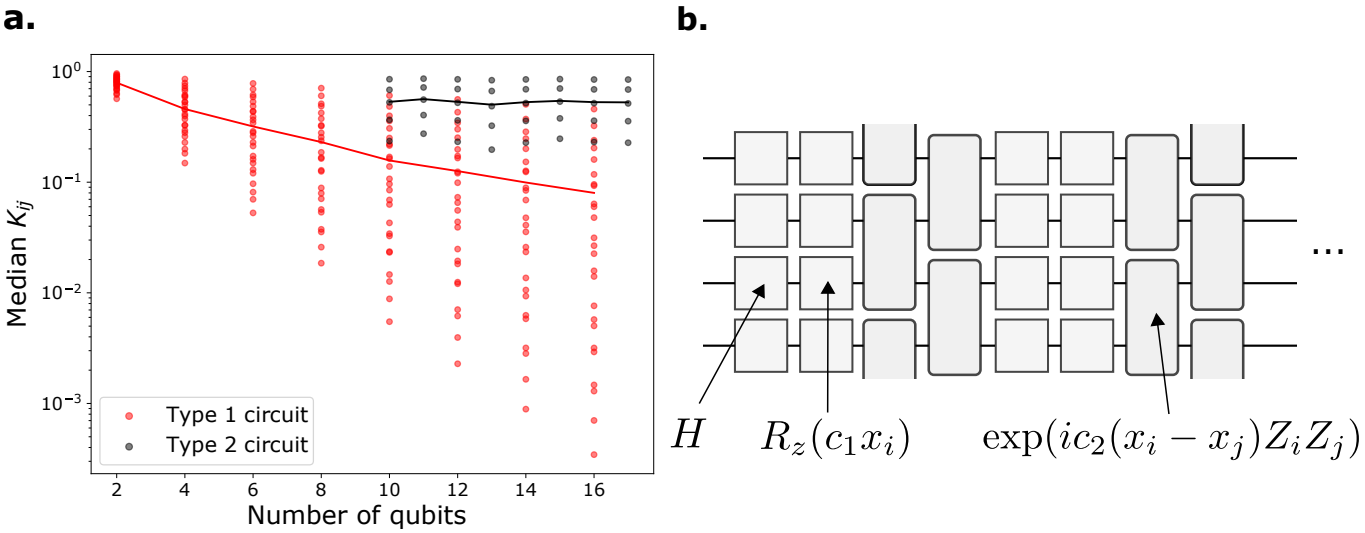


FIG. 5: Circuit structure and data preprocessing has a large impact on the resulting distributions of kernel matrix elements. (a) Distributions of median K with respect to a coarse grid search over $c_1, c_2 \in \{0.1, 0.15, 0.2, 0.25, 0.3\}$ for type 1 circuits with n-dimensional PCA compressions as input suggest that vanishing kernel magnitudes (red) make much of the gridsearch space inaccesible to realistic hardware experiments for even modest numbers of qubits. We found no such trend in K for type 2 circuits (Equation 2) implemented in our experiments with 67-dimensional input data.

$$U(x) = \prod_{\ell=1}^{L} U_B U_A(S_\ell(x)) \tag{B8}$$

$$U_A(z) = \bigotimes_{i=1}^{n} H^{(i)} R_z^{(i)}(c_1 z_{i2}) R_y^{(i)}(c_1 z_{i1}) R_z^{(i)}(c_1 z_{i0})$$

$$U_B = \prod_{(i,j)\in E(G)} \sqrt{\text{iSWAP}}^{(i,j)}$$

where $E(G)$ denotes the set of edges composing a length-n simple path permitted by the Sycamore connectivity, superscript (i) indicates action on qubit i, and $S : \mathbb{R}^d \to \mathbb{R}^{3n}$ denotes selection of a subset of $3n$ elements from the input data to be encoded into a given rotation layer. The specific choice of rotation and entangling gates was influenced by the gate set available on the processor at the time the experiments were conducted, namely $\sqrt{\text{iSWAP}}$ and the Sycamore gate [9]. Note that the use of Z rotations in U_A reduces the hardware depth of the corresponding circuit by 50% [30]. This architecture therefore encodes d-dimensional input data in $\mathcal{O}(d/n)$ depth on hardware. The fact that this circuit choice may be made arbitrarily shallow in number of qubits n rules out the possibility of demonstrating quantum advantage, but the experimental and design challenges of implementing this architecture are relevant to QKM in general.

As depicted in Figure 1, single qubit rotations in each circuit were "filled" sequentially from left to right, top to bottom beginning with the first element x_1 of the data and ending with x_{67}. Exceptions were made to this pattern to more evenly distribute single qubit rotations between entangling layers. We explored randomized filling schemes and found no significant difference to circuit performance nor inner product magnitudes. When the number of qubits is not an integer factor of 67, gaps appeared in at most two layers of the circuit.

2. Hyperparameter tuning in the quantum circuit

Our circuit design introduces a single parameter c_1 for multiplicative scaling of on elements of preprocessed x. We observed that train/test performance varied with respect to c_1 and therefore treated it as a hyperparameter of the kernel function to be optimized in simulation. In practice, the same optimization procedure can be carried out using sweeps over c_1 for hardware submissions. Figure 6a shows a typical outcome of the hyperparameter tuning process

and demonstrates both a local optimum for validation scores with respect to c_1 as well as a capacity for overfitting in the large-c_1 limit.

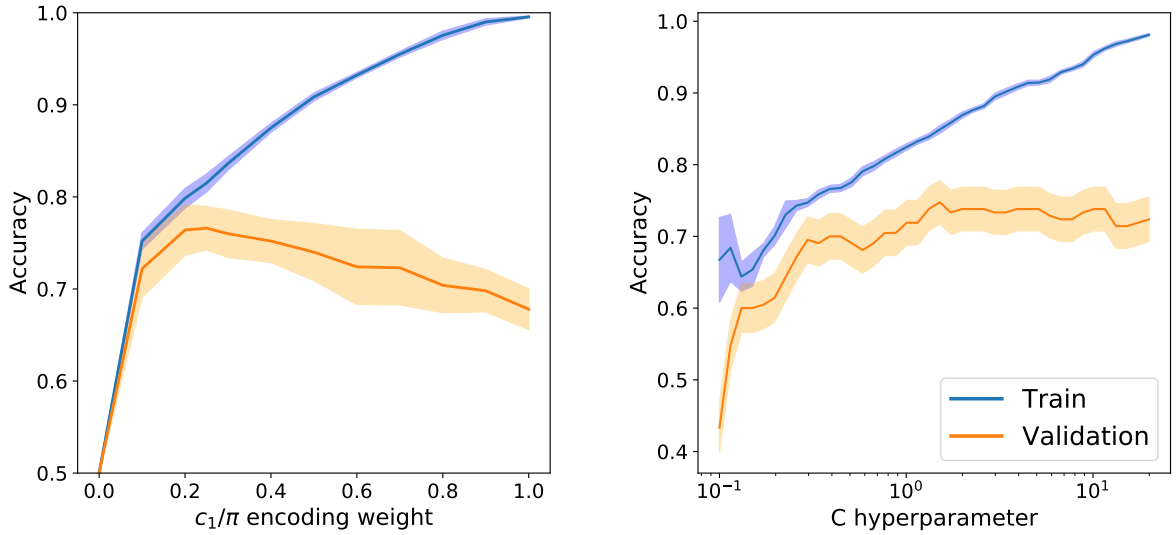


FIG. 6: Hyperparameter tuning for simulated type 2 circuits consisted of grid search optimization over two parameters. (left) The encoding parameter c_1 multiplies each encoded data element and impacts the typical separation (magnitude of K) for mapped feature space vectors. (right) The L2 penalty hyperparameter C used to train the SVM allows for robust performance in the presence of noise. Both plots correspond to 10 qubit circuits used in the experiment (c_1 was optimized over an $m = 1000$ subset of SN-67 dataset while C was optimized over experimental run dataset with $m = 210$).

The choice of optimal c_1 has a direct impact on the proximity of states $|\psi(x)\rangle$ mapped into the quantum state space; in the trivial limit $c_1 \to 0$ the unitary $U(x)$ becomes the identity map and $K_{ij} \to 1$; similarly in the large c_1 limit the angles between mapped input data grow linearly and K quickly vanishes. Therefore there is a tension between producing mapped states $\{|\psi(x_i)\rangle\}$ with good separability but without the isolation of mapped points in high dimensional space that would degrade performance, a phenomenon in classical machine learning known as the "curse of dimensionality" (e.g. [31]).

Appendix C: Dataset selection and preprocessing

We used the dataset provided in the Photometric LSST Astronomical Time-series Classification Challenge (PLAsTiCC) [14]. After engineering the 67-float data with binary labels (Section II A), preprocessing consisted of the following steps:

1. *Logscale transformation*: Many of the features were distributed in a lognormal distribution which motivates use of the transformation $x_i \to \log_{10}(x_i)$. Some information loss resulted from taking the absolute value of median flux, for which approximately 4% of entries in the original dataset were negative. All other absolute value operations resulted in negligible loss of sign information.

2. *Normalization/scaling and outliers*: Since the local rotations of $U(x)$ are 2π-periodic, an effective normalization scheme is to map the boundaries of the input range to $\left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$. Realistically, outliers will result in the effective range for most data being compressed into a much smaller space (thereby amplifying the effect of over/under-rotation errors) and so we scaled data in such a way to ignore the effects of large outliers by applying the transformation:

$$x' = \pi \left(\frac{x - P_1}{P_{99} - P_1} \right) - \frac{\pi}{2}$$

to every element of input data, where P_k denotes the value of the k-the percentile of the input domain. This is equivalent to typical implementations of a robust scaler with the quantile range set to (0.01, 0.99) (e.g. [32]),

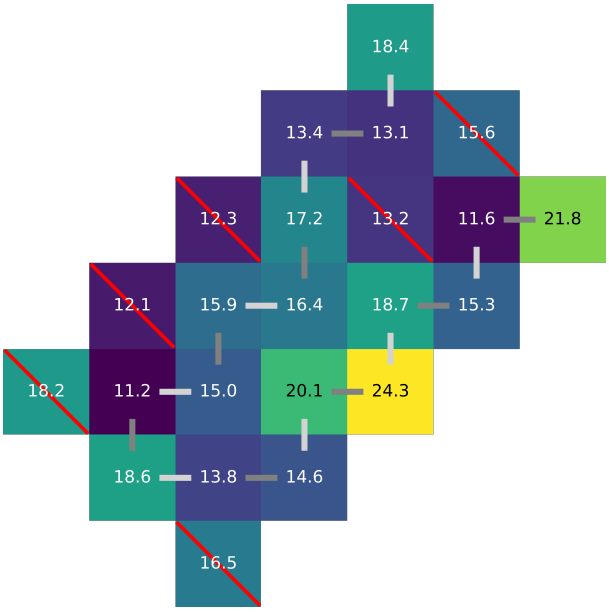


FIG. 7: Sample results of automated qubit selection with rejected qubits denoted by a red slash. Entangler patterns (light/dark gray) are overlaid on the Sycamore 23-qubit grid annotated with T_1 in μs. No more than 19 qubits may be assigned to the grid using our connectivity scheme, so that qubit selection has diminishing effects on performance as $n \to 19$.

which removes the effects of outliers on the rescaling. Then hyperparameter tuning was used to adjust the rotation parameters by a further multiplicative factor c_1 (see Appendix B 2).

Appendix D: Error mitigation

1. Device parameters

Periodic calibrations of the Sycamore superconducting qubit device produce diagnostic data describing qubit and gate performances. Calibration metrics relevant to this experiment included readout errors p_{00} (probability of a computational basis measurement reporting a "1" when the result should have been "0") and p_{11} (probability of a measurement reporting a "0" when the result should have been "1"), single qubit T_1, single qubit gate RB error, and $\sqrt{\text{iSWAP}}$ gate cross-entropy benchmarking (XEB) error. The readout error probabilities were used primarily for constructing and analyzing post-processing techniques for error correction while the remainder of the calibration data were fed in to an automated qubit selection algorithm.

2. Automated qubit selection

To improve the performance of the algorithm for a given number of qubits, we designed a graph traversal algorithm to select qubits based on diagnostic data taken during device calibration. Less weight was applied to p_0 and p_1 due to the availability of readout error mitigation techniques. We constructed a qubit graph $G_q = (E, V)$ where edges represent entangling gate connectivity and nodes represent qubits according to the Sycamore 23-qubit grid layout. The optimization was done by traversing all simple paths of fixed length k (implemented according to [33, 34]), and then scoring each path according to some function of the metrics for the subset of nodes and edges visited. Stated as an optimization problem, given an objective function $f : V \times E \to \mathbb{R}$ scoring subsets of vertices and edges composing an Eulerian graph, this algorithm finds the maximum evaluation of f over all possible graphs G_q:

$$\max_{f(V_k(G),E_k(G))} G_q \tag{D1}$$

subject to the constraints $|V_k(G)| = k$, $|E_k(G)| = k - 1$. Before applying f, the heterogeneous calibration data were normalized to the range $[0, 1]$ and inverted if they represented an error (as opposed to a fidelity). Letting the value of

the p-th category of calibration data for the i-th qubit v_i be $c_p(v_i)$, and similarly the p-th category of calibration for the i-th edge e_i be $d_p(e_i)$, and defining g_p as a scoring function applied to the p-th processed calibration metric, our implementation of f takes on the form

$$f(V) = \sum_{p \in C_1} \sum_{v_i \in G} g_p(c_p(v_i)) + \sum_{p \in C_2} \sum_{e_i \in G} g_p(c_p(e_i)) \tag{D2}$$

where C_1 and C_2 represent the calibration metrics corresponding to single and pairs of qubits respectively. We implemented g_p as a logarithmic function for T_1, T_2, and $f_{XEB,2q}$ metrics and a linear function for p_{00} and p_{11} metrics. Figure 7 shows the results of an example optimization overlaid on T_1 calibration results.

3. Readout error correction

Readout error resulting from relaxation and thermal excitation can be modelled by a stochastic bitflip process applied to the observed bitstrings. Here we describe an efficient and accurate technique for correcting readout error for quantum kernel methods.

Let $p(y^n|x^n)$ describe the conditional probability for observing bitstring y^n after exposing the bitstring x^n to n distinct bitflip channels, and let $q^k(y|x)$ for $x, y \in \{0, 1\}$ describe the corresponding probability for observing bit "y" after exposing the k-th bit "x" to a single bitflip channel. Then for the k-th qubit, the metrics introduced in Appendix D 1 as $p_{00} = q^k(1|0)$ and $p_{11} = q^k(0|1)$ may be used to partially undo readout error by means of postprocessing. We define a response matrix $R \in \mathbb{R}^{2^n \times 2^n}$ elementwise as $(R)_{xy} = p(y^n|x^n)$ that contains as its elements the total probability for transition from bitstring $x^n = x_1 \dots x_n$ to bitstring $y^n = y_1 \dots y_n$ computed as the product of $q^k(1|0)$ and $q^k(0|1)$ corresponding to each individual bit:

$$R_{xy} \equiv p(y_1 \dots y_n | x_1 \dots x_n) = \prod_{k=1}^{n} q^k(y_k|x_k) \tag{D3}$$

For simplicity, we assume that each individual bitflip may be modelled as an independent process, although the techniques discussed here are readily applicable to a system of dependent bitflips if the bitflip likelihoods are experimentally measured in parallel. Then evidently,

$$R = \bigotimes_{k=1}^{n} \begin{pmatrix} q^k(0|0) & q^k(0|1) \\ q^k(1|0) & q^k(1|1) \end{pmatrix} \tag{D4}$$

Note that R is generally asymmetric since typically $q^k(1|0) < q^k(0|1)$. While multiplying R^{-1} by the set of observed bitstring frequencies would recover the prior distribution of bitstring frequencies with good fidelity, standard matrix inversion is subject to instabilities and is not tractable for even modest numbers of qubits.

Since only the frequency of the all-zero's bitstring is necessary to compute $\hat{K}_{ij}$, we implemented correction using a small subset of bitstring transition probabilities to perform quick and relatively high-fidelity readout error correction in post-processing. We generated R and then truncated the full 2^n-dimensional basis to the bitstring space containing strings with Hamming weight $\leq k_{max}$ for some n-dependent k_{max} resulting in a truncated response matrix R_t. We then computed the pseudo-inverse R_t^{-1} and performed error correction by simple matrix multiplication on the array of experimental readout frequencies (similarly truncated). This simplification comes at the expense of knowledge about any other post-correction frequencies since the other bitstrings in the truncated space are off-center within the Hamming sphere of kept bitstrings, resulting in a bias in the inverted linear map.

We now analyze the effect of truncation on the readout error correction. The number of simultaneous readout error events (either relaxation or excitation) may be modelled as an induced random variable $Z = \sum_k X_k$ for $\Pr(X_k = 1) = q^k(\neg x|x)$. This distribution has expected value $\mu = \sum_k q^k(\neg x|x)$ and an exponentially suppressed likelihood for simultaneous readout errors via the Chernoff bound $\Pr(Z \geq k) \leq \exp(k - \mu - k \log(k/\mu))$. Thus a natural measure for the effect of Hamming weight truncation is the empirical probability allocated to the complement of the truncated subspace:

$$\Pr(Z > k_{max}) = 1 - \sum_{i=0}^{k_{max}} \Pr(Z = i) \tag{D5}$$

While Equation D5 describes the probability of events outside the truncated subspace, it does not directly translate to failure probability for truncated readout correction. To explore this effect numerically, we computed the output

probability distributions for a 10-qubit quantum kernel circuit sampled for 5000 repetitions, and then introduced artificial readout error (using bitflip probabilities taken from the Sycamore processor) followed by roughly 5% Gaussian noise on the sampled distribution. Figure 8 shows the error distribution as well as the effect of correcting using an inverted truncated response matrix for a variety of kernel magnitudes and truncation weights. We observed similar behavior for 14 and 17-qubit simulated experiments; readout error for quantum kernel experiments may be corrected reasonably well using a small fraction of the full bitflip response matrix.

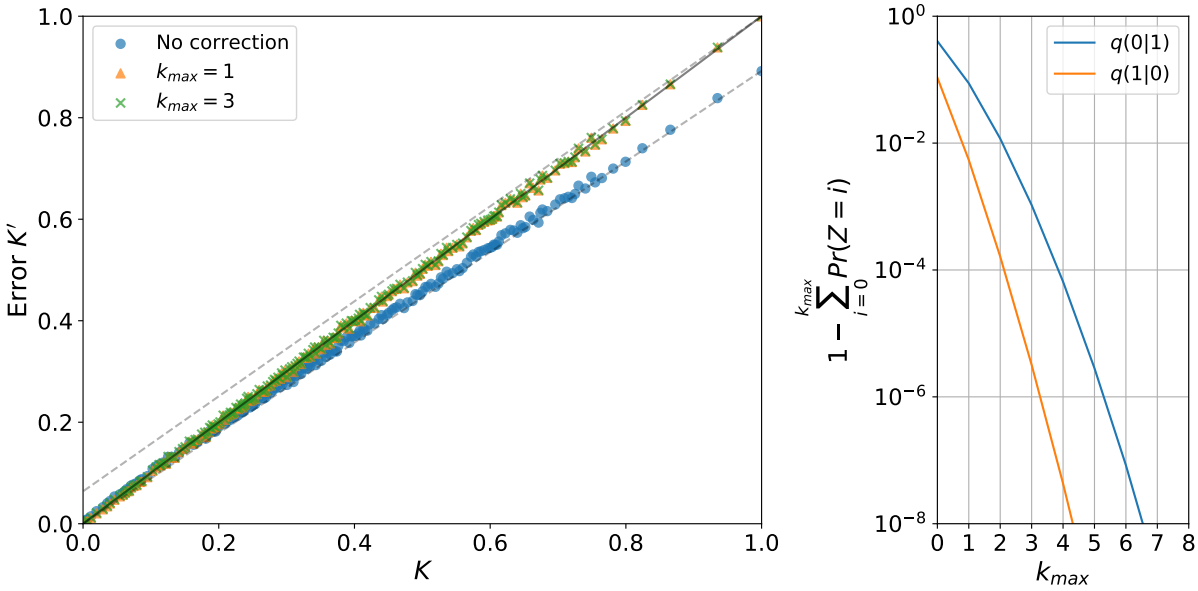


FIG. 8: Truncated readout error correction for 10 qubit circuit. (left) Correcting on the likely subspace described by $k_{max} = 1$ provides significant error correction, and further increasing k_{max} provides diminishing returns. Dashed lines indicate the infinite-shot upper/lower bounds given by Equation E6 (bounds are violated when empirical bitflip probabilities do not match imposed readout error probabilities); black line indicates perfect error correction. (right) Empirical probability for transition out of the weight $\leq k_{max}$ vanishes exponentially in the highest weight considered.

Figure 8 suggests a linear relationship between the kernel K' computed in the presence of readout error and the noiseless kernel K. While readout error does not constitute a linear process in general (the underlying bitflip probabilities and bitstring distributions may be modified to give rise to arbitrary effects on K within the bounds of Equation E6), by the arguments in Section IV demonstrating such an effect in general would imply minimal effect of readout error on the corresponding classifier. The degree to which readout error plays a role in quantum kernel methods is therefore an important research area for implementation on near-term processors.

4. Crosstalk optimization

Cross-talk between two-qubit gates on implemented on superconducting processors can contribute to decoherence and decrease circuit fidelity. Since our choice of circuit ansatz requires entire layers of entangling gates, we conducted diagnostic runs to determine whether executing these gates sequentially (in different staggering patterns) could improve performance compared to executing the gates simultaneously. We found that the completely parallel execution of entangling gates achieved the lowest cross entropy with respect to noiseless simulation compared to partially sequential arrangements. Therefore all entangling gates in this experiment were run in parallel.

Appendix E: Hardware error and performance

Figure 9 shows a typical outcome for sampled kernel elements $\hat{K}_{ij}$ compared to their true values as determined from noiseless simulation. Notably, all of the hardware outcomes are strongly biased towards zero as a result of decoherence. The observation that the SVM decision function is scale-invariant (Appendix IV) suggests that the performance of an SVM trained using data subject to hardware error will only be affected to the degree that the sampled kernel elements

$\hat{K}_{ij}$ differ from *some linear transformation of the corresponding exact elements* K_{ij}, so that the circuit fidelity and other typical metrics for hardware performance are not predictive of classifier performance in any obvious way. For instance, we achieved comparable test accuracy to noiseless simulation for $n = 14$ qubits despite circuit fidelity in the neighborhood of 30%.

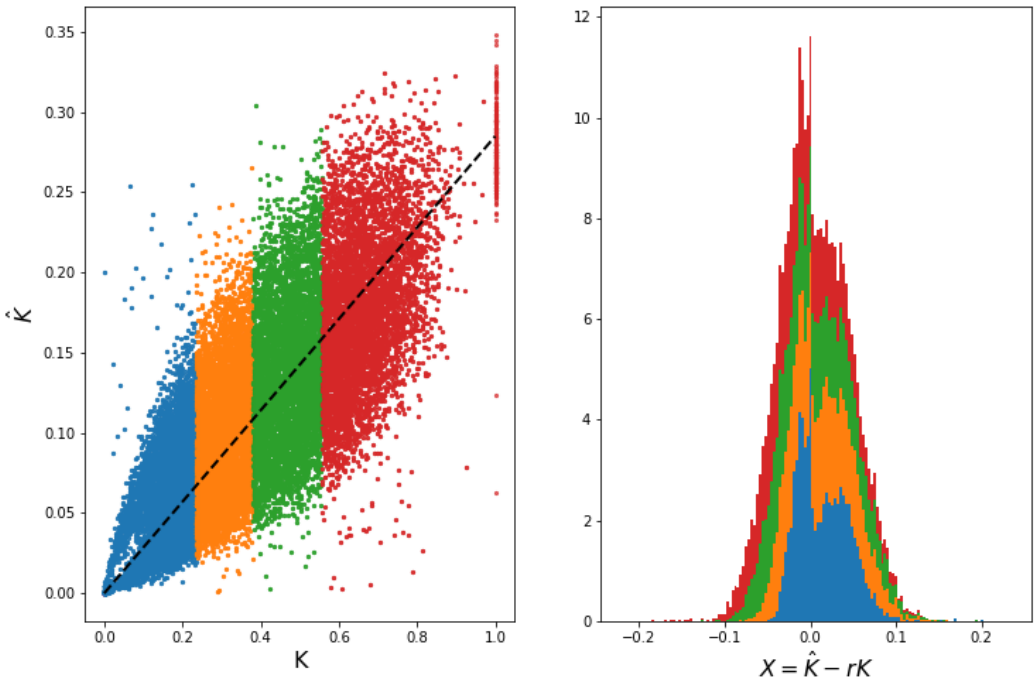


FIG. 9: (left) Distribution of sampled $\hat{K}$ compared to simulated K for 17 qubit train set ($m = 210$) colored according to quartiles of K. The dashed line indicates the value of rK for $r = 0.29$, demonstrating that the hardware trends towards some scaled value of K. The mean value of K_{ii} corresponding to $\mathrm{Tr}(|0\rangle\langle 0|) \equiv 1$ (elements in the top right) is a useful proxy for circuit fidelity and tends towards 30% for the qubit counts investigated. (right) The distribution of $\hat{K}$ around rK is irregular but exhibits consistent patterns with respect to kernel magnitude. The hardware error is therefore not normal with respect to (a scaled version of) K, which complicates bounds for guaranteed performance.

1. Effects of readout error in quantum kernel methods

Since $\hat{K}$ is estimated by computing the empirical probability of the all zeros bitstring $p(0)$, the effects of readout error may be bounded in a straightforward manner. The following analysis will consider readout error as the sole source of noise and ignore statistical effects and other sources of decoherence. The resulting bounds apply to only to the infinite-shot limit but will be shown to be approximately correct in the low-shot limit. As in Section D 3, we let $p(y^n|x^n)$ describe the conditional probability for observing bitstring y^n after exposing the bitstring x^n to n distinct bitflip channels, and let $q^k(y|x)$ for $x, y \in \{0, 1\}$ describe the corresponding probability for observing "y" after exposing the k-th bit "x" to a single bitflip channel. After exposing the all-zeros bitstring to a stochastic bitflip process repeatedly, the lowest possible value for $\hat{K}$ occurs when no bitstrings transform into the all-zeros bitstring. The expected fraction of events remaining is

$$p(0|0) = \prod_{k}^{n} (1 - q^k(1|0)) \tag{E1}$$

The maximum increase in the observed $p(0)$ will result from transitions from other bitstrings into the all-zeros bistring. Intuitively this will occur almost entirely due to low-weight bitstrings with just a few bitflips. The probability of transition into the all-zeros bitstring from an arbitrary starting bitstring $y^n = y_1 y_2 \dots y_n$ is $p(0|y^n) = \prod_{k=1}^{n} q^k(0|y_k)$, while the log-odds of each term in this product may be rewritten as

$$\log q(0|y_k) = \log q(0|0)(1 - y_k) + \log q^k(0|1) y_k \tag{E2}$$

The total log-odds for transition into 0 is then

$$\log p(0|y^n) = \sum_{k=1}^{n} \log q^k(0|0) + \sum_{k=1}^{n} \log \frac{q^k(0|1)}{q^k(0|0)} y_k \tag{E3}$$

which is in the form $w \cdot y^n + b$ with $w \in \mathbb{R}^n$, $b \in \mathbb{R}$, indicating a linear dependence of $\log p(0|y^n)$ on y^n. Since the logarithm is strictly increasing, the corresponding integer programming problem for maximizing $p(0|y^n)$ is:

$$\max_{w} \sum_{k=1}^{n} w_k x_k \tag{E4}$$

$$\text{subject to } x_k \in \{0, 1\}, \qquad \sum_{k=1}^{n} x_k > 0 \tag{E5}$$

where the second constraint avoids the trivial solution of the all-zeros bitstring. By inspection of Equation E3, $q^k(0|0) = 1 - q^k(1|0)$ implies that w is strictly negative with the realistic assumption that $q^k(1|0) < 0.5$, $q^k(0|1) < 0.5$. In this case, the solution maximizing Equation E4 must be a bitstring with weight 1 with a "1" at the position $\text{argmax}_k q^k(0|1)$. Combined with Equation E1 this results in the bound:

$$\hat{K} \prod_{k}^{n} \left(1 - q^k(1|0)\right) \leq \hat{K}' \leq \left(1 - \hat{K}\right) \max_{k} q^k(0|1) + \hat{K} \tag{E6}$$

Where $\hat{K}'$ describes the estimated kernel element after readout error has occurred. This bound is plotted in Figure 8 for one instance of a 10-qubit readout error calibration. The form of these bounds further justify the need for large kernels, since the bound becomes increasingly loose as $\hat{K}$ approaches the magnitude of typical readout error probabilities on the quantum processor.

We now discuss the implementation of the readout error correction technique described in Appendix D 3. While routine calibration of the device returns diagnostic information on readout error probabilities $q^k(0|1)$ and $q^k(1|0)$, these quantities may drift in the time between calibration and experiment resulting in lower quality readout correction. To account for drift, we periodically estimated $q^k(0|1)$ and $q^k(1|0)$ by preparing a sequence of random bitstrings $|s\rangle$ and the complement $|s \oplus 1\rangle$ and then measured in the computational basis. We then computed empirical bitflip likelihoods for measurements in parallel over the specific qubits used in each experiment and computed the time-averaged likelihoods to use for readout correction. We averaged over the different prepared states to reduce the impact of imperfect state preparation on measured outcomes.

Figure 10 shows learning curves computed for the 14-qubit experiment with and without readout correction for post-processing. While classifiers trained using error-corrected results are capable of achieving higher accuracies on the reserved test set, the choice of C hyperparameter for improved test accuracy did not generally correspond to improved accuracies for the validation sets. Therefore, we could not consistently improve the classifier performance by applying readout correction and opted to present the results achieved without readout correction in Figure 4 in the main body.

2. Effects of statistical error on SVM accuracy

While recent work [3] has established performance bounds relating SVM accuracy to statistical sampling error for quantum kernel methods, we conducted experiments using orders of magnitude fewer repetitions than necessary to achieve robust classification. In addition, modified SVM algorithms exist for processing noisy inputs in the data space [35] but these modifications do not generalize to processing noise in the feature space (i.e. $\Delta k(x_i, x_j)$). Given these limitations, we chose to explore the effects of statistical noise numerically to determine the relative impact of statistical noise on final classifier accuracy compared to other sources of error.

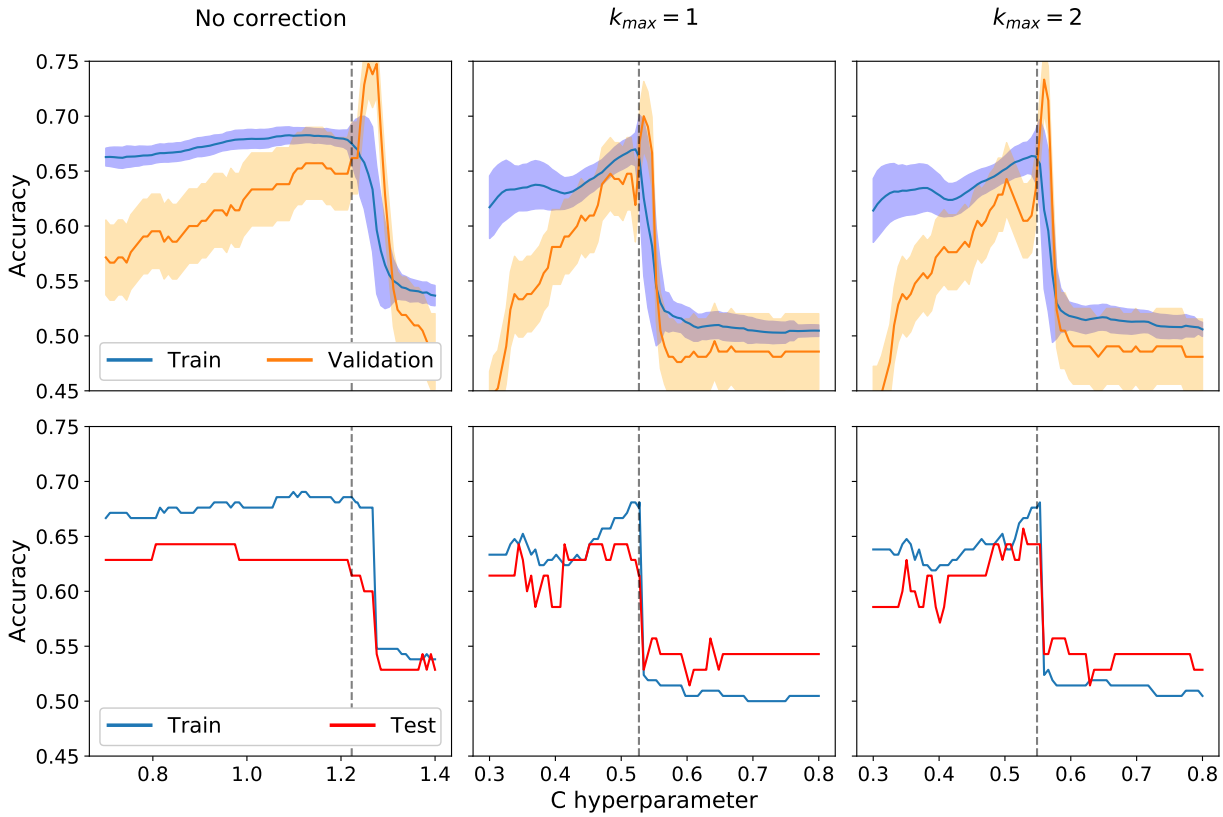


FIG. 10: We analyzed the LOO cross-validation accuracy versus accuracy on the reserved test set to determine the effect of (truncated) readout correction using readout error likelihoods determined experimentally at periodic intervals during the 14-qubit experiment. While the trained classifiers are often able to achieve significantly higher accuracy on the reserved test set, the error-corrected validation accuracy is not predictive of improved test accuracy. For instance, the $k_{max} = 2$ classifier achieves a 65.7% validation accuracy and a 64.3% test accuracy while the classifier with no readout correction achieves 66.1% validation accuracy and 61.4% test accuracy, indicating that the improved test score cannot be consistently predicted by LOOCV.

Figure 11 shows simulated trials of the type 2 circuits used for experimental runs. Each circuit was initially simulated with an full wavefunction simulator, and then the amplitude $K = |\langle 0|\psi\rangle|^2$ was used to sample R repetitions from the implied binomial distribution $\mathrm{Bin}(R, K)$. The sampled kernel elements were used to train and validate SVM classifiers following the procedure outlined in Section III B. The results indicate there are diminishing returns in classifier accuracy beyond the $R = 5000$ repetitions we used for the experiments, but that this choice incurs $\sim$ 1-2% classifier error compared to the infinite-shot limit. As expected, greater statistical noise results in less overfitting for the model.

3. Classifier results

Figure 12 shows the result of tuning the hyperparameter C controlling the $L2$ penalty for violating the SVM margin. The C values resulting in optimal validation scores were then used to determine the final train/test scores reported in Figure 4 of the main body.

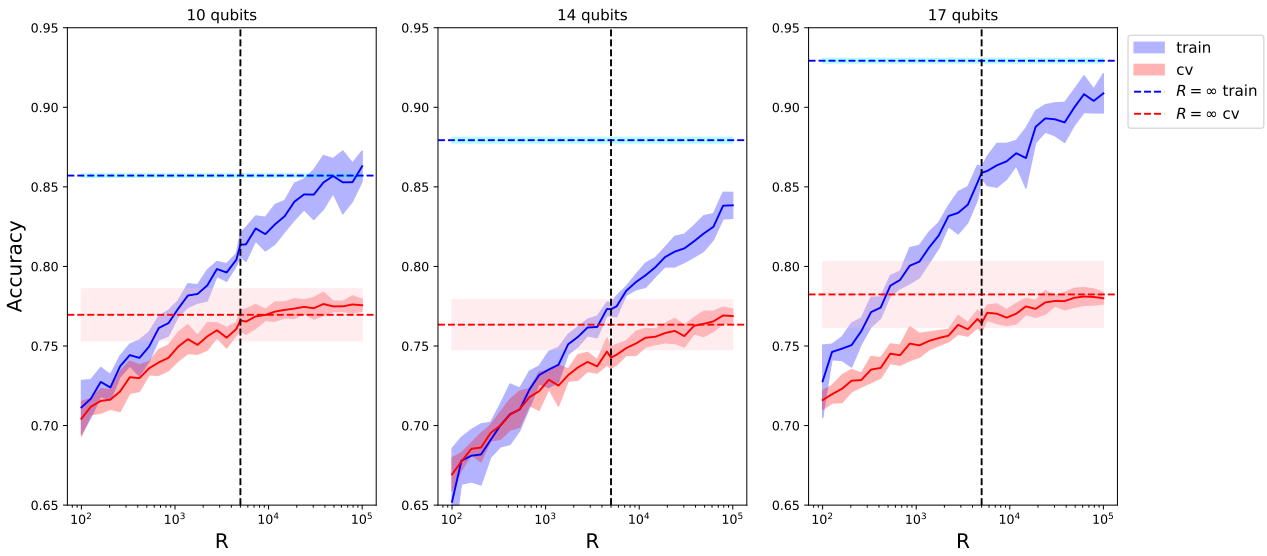


FIG. 11: We empirically investigated the effect of statistical noise by generating a sampling a binomial distribution with success probability equal to K and enfocing symmetry on the resulting kernel matrix. The results show that additional circuit repetitions beyond $R \approx 5000$ provide diminishing returns to the validation accuracy of the classifier. We note that typically 50,000 circuit repetitions per kernel element are required to achieve cross-validated accuracy comparable to noiseless simulation. Fill for finite-R represents 1σ interval for stratified 10-fold cross-validated train/test scores over 10 trials of downsampling to R shots.

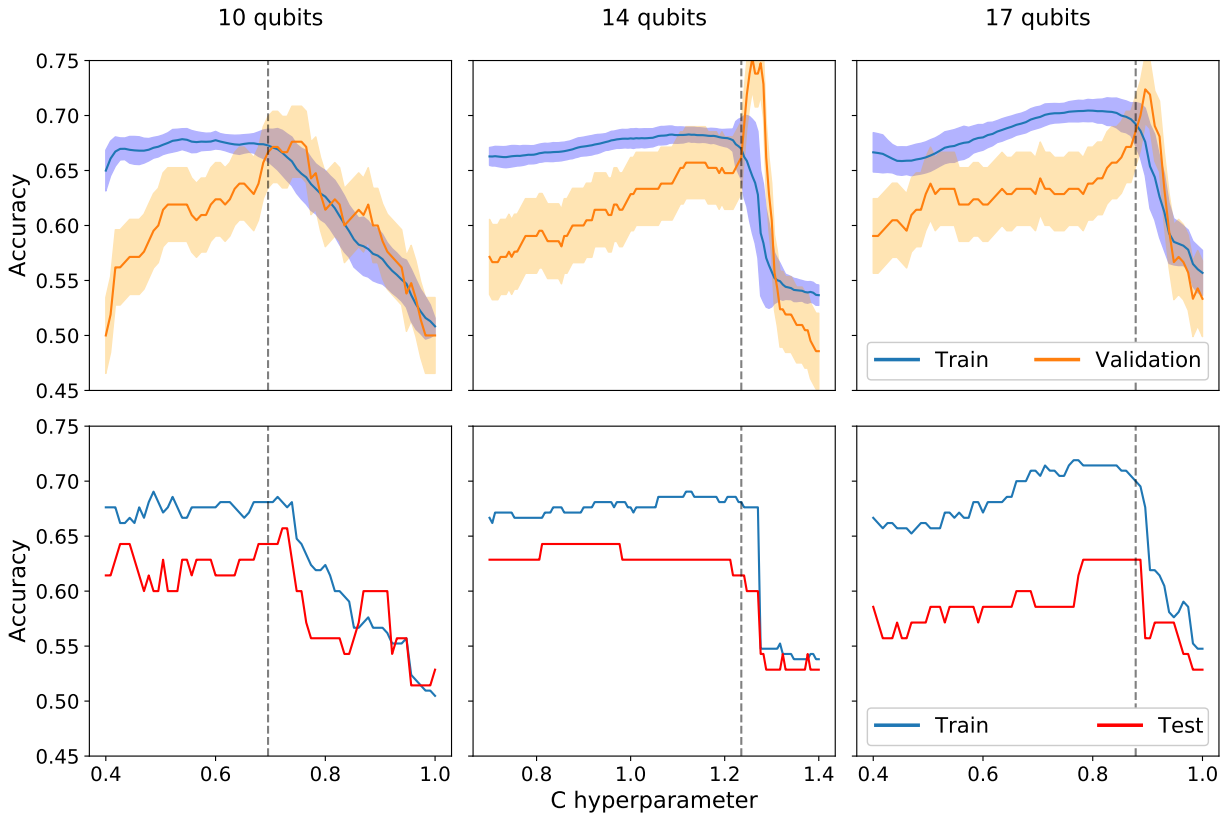


FIG. 12: Hyperparameter optimization for hardware kernels is performed by tuning the L1 penalty parameter C via LOOCV on the training data (top row) during model validation, which then becomes fixed for evaluation of the model on the test set (bottom row). The capacity of the hardware-based models to overfit the data is drastically reduced, and oftentimes the SVM behavior becomes pathological. To avoid undesirable generalization behavior, the validation score corresponding to the optimal C was required to be no greater than the corresponding training score. The vertical dashed line indicates the optimal C decided in the validation stage.