Quantum machine learning of large datasets using randomized measurements

IOP Publishing Mach. Learn.: Sci. Technol. 4 (2023) 015005 https://doi.org/10.1088/2632-2153/acb0b4

MACHINE LEARNING
Science and Technology

OPEN ACCESS

RECEIVED
12 December 2022

ACCEPTED FOR PUBLICATION
5 January 2023

PUBLISHED
20 January 2023

BENCHMARK

Quantum machine learning of large datasets using randomized measurements

Tobias Haug*, Chris N Self and M S Kim

QOLS, Blackett Laboratory, Imperial College, London SW7 2AZ, United Kingdom
* Author to whom any correspondence should be addressed.

E-mail: thaug@ic.ac.uk

Keywords: quantum machine learning, quantum algorithm, quantum kernel, quantum computing, supervised learning

Abstract

Quantum computers promise to enhance machine learning for practical applications. Quantum machine learning for real-world data has to handle extensive amounts of high-dimensional data. However, conventional methods for measuring quantum kernels are impractical for large datasets as they scale with the square of the dataset size. Here, we measure quantum kernels using randomized measurements. The quantum computation time scales linearly with dataset size and quadratic for classical post-processing. While our method scales in general exponentially in qubit number, we gain a substantial speed-up when running on intermediate-sized quantum computers. Further, we efficiently encode high-dimensional data into quantum computers with the number of features scaling linearly with the circuit depth. The encoding is characterized by the quantum Fisher information metric and is related to the radial basis function kernel. Our approach is robust to noise via a cost-free error mitigation scheme. We demonstrate the advantages of our methods for noisy quantum computers by classifying images with the IBM quantum computer. To achieve further speedups we distribute the quantum computational tasks between different quantum computers. Our method enables benchmarking of quantum machine learning algorithms with large datasets on currently available quantum computers.

1. Introduction

Quantum machine learning aims to use quantum computers to enhance the power of machine learning [1, 2]. One possible route to quantum advantage in machine learning is the use of quantum embedding kernels [3–6], where quantum computers are used to encode data in ways that are difficult for classical machine learning methods [7–9]. Noisy intermediate scale quantum computers [10, 11] may be capable of solving tasks difficult for classical computers [12, 13] and have shown promise in running proof-of-principle quantum machine learning applications [14–25]. However, currently available quantum computers are at least 6 orders of magnitude orders slower than classical computers. Furthermore, running quantum computers is comparatively expensive, necessitating methods to reduce quantum resources above all else. Thus, it is important to develop better methods to run and benchmark noisy quantum computers. Here, several bottlenecks limit quantum hardware for machine learning in practice. First, the quantum cost of measuring quantum kernels with conventional methods scales quadratically with the size of the training dataset [5]. This quadratic scaling is a severe restriction, as commonly machine learning relies on large amounts of data. Second, the data has to be encoded into the quantum computer in an efficient manner and generate a useful quantum kernel. Various encodings have been proposed [26, 27], however the number of features is often limited by the number of qubits [19, 20] or the quantum kernel is characterized only in a heuristic manner. Finally, the inherent noise of quantum computers limits the quality of the experimental results. Error mitigation has been proposed to reduce the effect noise [28], however in general this requires a large amount of additional quantum computing resources [29].

Here, we use randomized measurements to calculate quantum kernels. The quantum computing time scales linearly and the classical post-processing time quadratically with the size of the dataset. While our

method scales in general exponentially in the number of qubits, compared to other methods a substantially lower number of measurements is needed for intermediate-sized quantum computers of about ten qubits. Additionally, we can reuse the collected measurement data to effectively mitigate the noise of quantum computers. To efficiently load high-dimensional data into the quantum computer, we apply an encoding that scales linearly with the depth of parameterized quantum circuits (PQCs). The resulting quantum kernel is characterized with the quantum Fisher information metric (QFIM) and can be approximately described by the radial basis function (RBF) kernel. We introduce the natural PQC (NPQC) with an exactly known QFIM and demonstrate its usefulness for quantum machine learning. We implement our approach on the IBM quantum computer to classify handwritten images of digits with high accuracy. We experimentally demonstrate further speedups by parallelizing quantum computational tasks between different quantum computers. With our approach, currently available quantum computers can process larger datasets containing ten thousands of entries within a feasible time, extending the range of quantum machine learning algorithms that can be run in practice.

2. Support vector machine

Our goal is to classify unlabeled test data by learning from labeled training data as shown in figure 1(a). The dataset for the supervised learning task $\{\{\boldsymbol{x}_i, y_i\}\}_{i=1}^{L}$ contains in total L items. The ith data item is described by a M-dimensional feature vector $\boldsymbol{x}_i$ and corresponding label y_i. Label y_i belongs to C possible classes, while the feature vector $\boldsymbol{x}_i = \{\boldsymbol{x}_i^{(n)}\}_{n=1}^{M}$ consists of M real-valued entries. To learn and classify data, we use a kernel $K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ that is a measure of distance between feature vectors $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ [2]. The kernel corresponds to an embedding of the M-dimensional data into a higher-dimensional space, where analysis of the data becomes easier [30]. In quantum kernel learning, we embed the data into the high-dimensional Hilbert space of the quantum computer and use it to calculate the kernel (see figure 1(b)). With the kernels, we train a support vector machine (SVM) to find hyperplanes that separate two classes of data (see figure 1(c)). The SVM is optimized using the kernels of the training dataset with a semidefinite program that can be efficiently solved with classical [31] or quantum computers [32, 33].

$$\max_{\boldsymbol{\alpha}} \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} y_i y_j \alpha_i \alpha_j K(\boldsymbol{x}_i, \boldsymbol{x}_j) \tag{1}$$

subject to the conditions $\sum_i \alpha_i y_i = 0$ and $\alpha_i \geqslant 0$. After finding the optimal weights $\boldsymbol{\alpha}^*$, the SVM predicts the class of a feature vector $\boldsymbol{\eta}$ as $y_i^{\text{pred}} = \text{sign}(\sum_i \alpha_i^* y_i K(\boldsymbol{x}_i, \boldsymbol{\eta}) + b)$, where b is calculated from the weights. One can extend this approach to distinguish C classes by solving C SVMs that separate each class from all other classes.

The power of the SVM highly depends on a good choice of kernel $K(\boldsymbol{x}_i, \boldsymbol{x}_j)$, such that it captures the essential features of the dataset. In the following, we propose a powerful class of quantum kernels that can be implemented with currently available quantum computers. Then, we show how to compute kernels for large datasets and mitigate the noise inherent in real quantum devices.

3. Encoding

A crucial question is how to efficiently encode a high-dimensional feature vector into a quantum computer while providing a useful kernel for machine learning. We encode the M-dimensional feature vector $\boldsymbol{x}_i$ as M-dimensional parameter $\boldsymbol{\theta}_i$ of a PQC via

$$\boldsymbol{\theta}_i = \boldsymbol{\theta}_\text{r} + c\boldsymbol{x}_i\,, \tag{2}$$

where c is a scaling constant and $\boldsymbol{\theta}_\text{r}$ the reference parameter. As shown in figure 1(b), we use hardware efficient PQCs with N qubits and d layers of unitaries for the encoding [34]. The lth layer is composed of a product of parameterized single qubit rotations $R_{l,k}(\boldsymbol{\theta}_i^{(n_{l,k})})$ acting on qubit k and non-parameterized entangling gates W_l that generate the quantum state $|\psi(\boldsymbol{\theta}_i)\rangle = \prod_{l=1}^{d} W_l(\prod_k R_{l,k}(\boldsymbol{\theta}_i^{(n_{l,k})}))|0\rangle^{\otimes N}$.

Our choice of quantum kernel measures the distance between two encoding states as given by the fidelity between $\rho(\boldsymbol{\theta}_i)$ and $\rho(\boldsymbol{\theta}_j)$ [8, 27]

$$K(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j) = \text{Tr}(\rho(\boldsymbol{\theta}_i)\rho(\boldsymbol{\theta}_j))\,, \tag{3}$$

which for pure states $\rho(\boldsymbol{\theta}_i) = |\psi(\boldsymbol{\theta}_i)\rangle\langle\psi(\boldsymbol{\theta}_i)|$ reduces to $K(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j) = \left|\langle\psi(\boldsymbol{\theta}_i)|\psi(\boldsymbol{\theta}_j)\rangle\right|^2$.

We can formalize the expressive power of our encoding with the QFIM $\mathcal{F}(\boldsymbol{\theta})$, which is a $M \times M$ dimensional positive-semidefinite matrix that provides information about the kernel in the proximity of

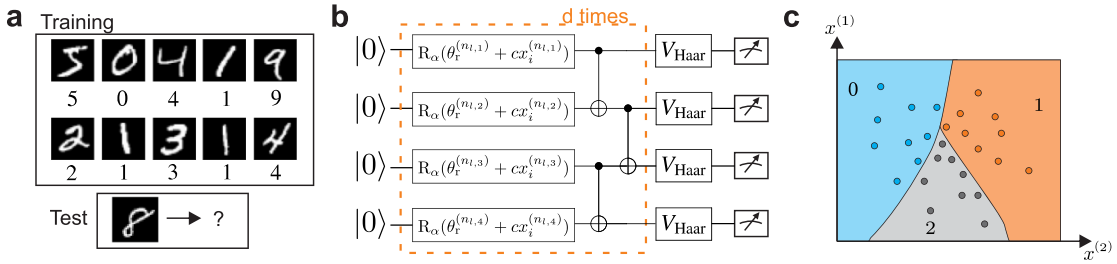

Figure 1. (a) Supervised learning to classify images of handwritten digits. By learning from a training set of labeled images, our goal is to identify previously unseen test data correctly. The support vector machine (SVM) learns using a kernel (equation (3)) which is a measure of distance between the data. (b) We learn a dataset of L images with $i = \{1, \ldots, L\}$, each with M pixels $n = \{1, \ldots, M\}$, where we denote each pixel with $\boldsymbol{x}_i^{(n)} \in \mathbb{R}$. For the ith image, we encode the M-dimensional feature vector $\boldsymbol{x}_i$ into a parameterized quantum circuit (PQC) with an M-dimensional parameter vector $\boldsymbol{\theta}_i$. The PQC has N qubits and d layers of parameterized single qubit rotations and two-qubit entangling gates. We encode the $n_{l,k} \equiv n$ entry of the feature vector into a single qubit rotation acting on qubit k and layer l via $\theta_i^{(n)} = \theta_r^{(n)} + cx_i^{(n)}$ (equation (2)), where $\boldsymbol{\theta}_r$ is a fixed reference parameter and c a scaling factor. The number of encoded features scales linearly with N and d. The kernel (equation (3)) is characterized by the quantum Fisher information metric (QFIM) $\mathcal{F}(\boldsymbol{\theta}_r)$ and can be approximately described by the radial basis function kernel (equation (5)). We calculate the quantum kernel by measuring the PQC in randomized local bases of Haar random unitaries V_{Haar}. (c) The SVM trained with the quantum kernel draws the decision boundaries (here shown for a two-dimensional feature vector space and three possible digits) that classify each feature vector to its corresponding label.

$\boldsymbol{\theta}$ [35]. For a pure state $|\psi\rangle = |\psi(\boldsymbol{\theta})\rangle$ it is given by $\mathcal{F}_{ij}(\boldsymbol{\theta}) = 4[\langle \partial_i\psi|\partial_j\psi\rangle - \langle \partial_i\psi|\psi\rangle\langle\psi|\partial_j\psi\rangle]$, where $\partial_j|\psi\rangle$ is the gradient in respect to the j-th element of $\boldsymbol{\theta}$ [36]. In the limit $c \to 0$ of encoding equation (2), the kernel of a pure quantum state can be written as

$$K(\boldsymbol{\theta}_r, \boldsymbol{\theta}_r + c\boldsymbol{x}_i) = 1 - \frac{c^2}{4}\boldsymbol{x}_i^T \mathcal{F}(\boldsymbol{\theta}_r)\boldsymbol{x}_i = 1 - \frac{c^2}{4}\sum_{k=1}^{M} \lambda_k g_k\,, \tag{4}$$

where λ_k is the kth eigenvalue of the QFIM $\mathcal{F}(\boldsymbol{\theta}_r)$ and $g_k = |\langle \boldsymbol{x}_i, \boldsymbol{\mu}_k\rangle|^2$ is the inner product of the feature vector $\boldsymbol{x}_i$ and the kth eigenvector $\boldsymbol{\mu}_k$ of $\mathcal{F}(\boldsymbol{\theta}_r)$. $R = \text{rank}(\mathcal{F})$ (the number of non-zero eigenvalues) of $\mathcal{F}(\boldsymbol{\theta}_r)$ is an important measure of the properties of the PQC and the encoding [35]. The $M - R$ eigenvectors $\boldsymbol{\mu}_k$ with $\lambda_k = 0$ have no effect on the kernel with $K(\boldsymbol{\theta}, \boldsymbol{\theta} + c\boldsymbol{\mu}_k) = 1$. Thus, feature vectors $\boldsymbol{x}_q \in \text{span}\{\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_{M-R}\}$ that lie in the space of eigenvectors with eigenvalue zero cannot be distinguished using the kernel as they have the same value $K(\boldsymbol{\theta}, \boldsymbol{\theta} + c\boldsymbol{x}_q) = 1$. Further, the size of the eigenvalues λ_k determines how strongly the kernel changes in direction $\boldsymbol{\mu}_k$ of the feature space. By appropriately designing the QFIM as the weight matrix of the kernel, generalizing from data could be greatly enhanced [27, 35, 37]. For example, the feature subspace with eigenvalue 0 could be engineered such that it coincides with data that belongs to a particular class. Conversely, features that strongly differ between different classes could be tailored to have large eigenvalues such that they can be easily distinguished [37]. For a PQC with N qubits the rank is upper bounded by $R \leqslant 2^{N+1} - 2$, which is the maximal number of features that can be reliably distinguished by the kernel [35].

It has been recently shown that the kernel of pure quantum states of hardware efficient PQCs can be approximated as Gaussian or RBF kernels [38], which are one of the most popular non-linear kernels with wide application in various machine learning methods [39]. Specifically, for small enough c with the encoding equation (2), we can approximately describe the quantum kernel as

$$K(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j) \approx \exp\left[-\frac{c^2}{4}(\boldsymbol{x}_i - \boldsymbol{x}_j)^T \mathcal{F}(\boldsymbol{\theta}_r)(\boldsymbol{x}_i - \boldsymbol{x}_j)\right], \tag{5}$$

which is the RBF kernel with the QFIM as weight matrix $\mathcal{F}(\boldsymbol{\theta}_r)$ [38]. While for general PQCs the QFIM is *a priori* not known, a type of PQC called NPQC has the special property that the QFIM takes a simple form with $\mathcal{F}(\boldsymbol{\theta}_r) = I$, where I is the identity matrix and $\boldsymbol{\theta}_r$ a particular reference parameter, which we will choose in the following for the NPQC (see [40] and appendix A). The NPQC forms an approximate isotropic RBF kernel that can serve as a well characterised basis for quantum machine learning. We also study another commonly used type of hardware efficient circuit (YZ-CX PQC) composed of single qubit rotations and CNOT gates arranged in a one-dimensional nearest-neighbor chain with a non-trivial QFIM $\mathcal{F}(\boldsymbol{\theta}_r) \neq I$. For the YZ-CX PQC we choose a randomly drawn $\boldsymbol{\theta}_r$, we find that the overall performance is nearly independent of the choice.

Further details on the NPQC and YZ-CX PQC are shown in the appendix A. The scaling factor c controls the scale of the resulting values of the quantum kernel. Too small kernel values can impede learning as the

model becomes too constrained. We can restrict the kernel from below $K_{\min} < K(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j)$ for all i, j by choosing c as

$$c^2 < \frac{-4\log(K_{\min})}{\min_{i,j}(\boldsymbol{x}_i - \boldsymbol{x}_j)^{\mathrm{T}}\mathcal{F}(\boldsymbol{\theta}_{\mathrm{r}})(\boldsymbol{x}_i - \boldsymbol{x}_j)}\,. \tag{6}$$

4. Measurement

We calculate the L quantum kernels using randomized measurements [41–43] by measuring quantum states in r randomly chosen single qubit bases. We first choose r sets $n = \{1, \ldots, r\}$ of transformations $V^{(n)} = \otimes_{k=1}^{N} V_k^{(n)}$, composed of random single qubit rotations $V_k^{(n)}$ drawn according to the Haar measure $\mathbb{SU}(2)$ acting on each qubit k. Then, we prepare the quantum state $\rho(\boldsymbol{\theta}_i)$ and rotate into a random basis $V^{(n)}\rho(\boldsymbol{\theta}_i)V^{(n)\dagger}$. Then, we measure s samples of the rotated state in the computational basis and estimate the probability $P_i^{(n)}(\nu_q)$ of measuring the computational basis state $\nu \in \{0,1\}^N$ for state $\rho(\boldsymbol{\theta}_i)$ and transformation n. This procedure is repeated for the r transformations and L quantum states. The kernel $K_{\mathrm{b}}(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j) = \mathrm{Tr}(\rho(\boldsymbol{\theta}_i)\rho(\boldsymbol{\theta}_j))$ via randomized measurements is then calculated as [42]

$$\begin{aligned} K_{\mathrm{b}}(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j) &= \mathrm{Tr}(\rho(\boldsymbol{\theta}_i)\rho(\boldsymbol{\theta}_j)) \\ &= \sum_{\nu,\nu'}(-2)^{-D(\nu,\nu')}\sum_{n=1}^{r} P_i^{(n)}(s)P_j^{(n)}(s')\,, \end{aligned} \tag{7}$$

where $D(\nu, \nu')$ is the Hamming distance that counts the number of bits that differ between the computational states ν and ν'.

To measure all entries of the kernel, we perform $N_{\mathrm{R}} = srL$ measurements in total. The error ΔK of estimating a single kernel entry scales as $\Delta K \propto 1/(s\sqrt{r})$ [42]. Thus, for a fixed error it is beneficial to choose the number of bases r to a relatively small number compared to s. Note that for sufficient accuracy a minimal number of r is needed which increases with N. Overall, the number of measurements needed to estimate the kernel scales as $N_{\mathrm{R}} \propto 2^{aN}L$, with a factor $a \lesssim 1$ that depends on the type of state being measured [41, 42] and can be improved by importance sampling [44]. While for large N, the exponential measurement cost is prohibitive, for intermediate qubit number on the order of ten qubits the measurement cost is moderate. With our method, the number of measurements needed to determine the full kernel matrix scales only linearly with the dataset size $N_{\mathrm{R}} \propto L$, a quadratic speedup in contrast to other methods. Other commonly used measurement strategies such as the swap test [45, 46] or the inversion test [18, 20] have to explicitly prepare both states $\rho(\boldsymbol{\theta}_i)$ and $\rho(\boldsymbol{\theta}_j)$ on the quantum computer. Thus, they scale unfavorably with the square $N_{\mathrm{R}} \propto L^2$ of the dataset size (see appendix B). While randomized measurements requires an overhead compared to standard methods, we find that for relatively small datasets, $L > 21$, randomized measurement requires less measurements for our experimental parameters (see appendix D). For $L = 10^3$, we find that randomized measurement requires a factor 100 lower number of measurements compared to the parameters used in previous works. A further advantage is found in error mitigation. For standard measurement methods on noisy quantum computers, error mitigation adds a substantial cost to the measurement budget [29]. In contrast, randomized measurement can mitigate errors without further measurement cost as we show in the following.

5. Error mitigation

In general, quantum computers are affected by noise, which will turn the prepared pure quantum state into a mixed state and may negatively affect the capability to learn. For depolarizing noise, we can use the information gathered in the process to mitigate its effect and infer the noiseless value of the kernel.

For global depolarizing noise, with a probability p_i the pure quantum state $|\psi(\boldsymbol{\theta}_i)\rangle$ is replaced with the completely mixed state $\rho_{\mathrm{m}} = I/2^N$, where I is the identity matrix. The resulting quantum state is the density matrix $\rho(\boldsymbol{\theta}_i) = (1-p_i)|\psi(\boldsymbol{\theta}_i)\rangle\langle\psi(\boldsymbol{\theta}_i)| + p_i(2-p_i)\rho_{\mathrm{m}}$. The purity can be determined from the randomized measurements $\mathrm{Tr}(\rho(\boldsymbol{\theta}_i)^2) = K_{\mathrm{b}}(\boldsymbol{\theta}_i, \boldsymbol{\theta}_i) = (1-p_i)^2 + \frac{p_i}{2^N}$ by reusing the same data used to compute the kernel entries. Using these purities, the depolarization probability p_i can be calculated by solving a quadratic equation [23, 47]. With p_i and the measured kernel $K_{\mathrm{b}}(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j)$ affected by depolarizing noise, the mitigated kernel is approximated by

$$K_{\mathrm{m}}(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j) \approx \frac{K_{\mathrm{b}}(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j)}{\sqrt{\mathrm{Tr}(\rho(\boldsymbol{\theta}_i)^2)\mathrm{Tr}(\rho(\boldsymbol{\theta}_j)^2)}}\,. \tag{8}$$

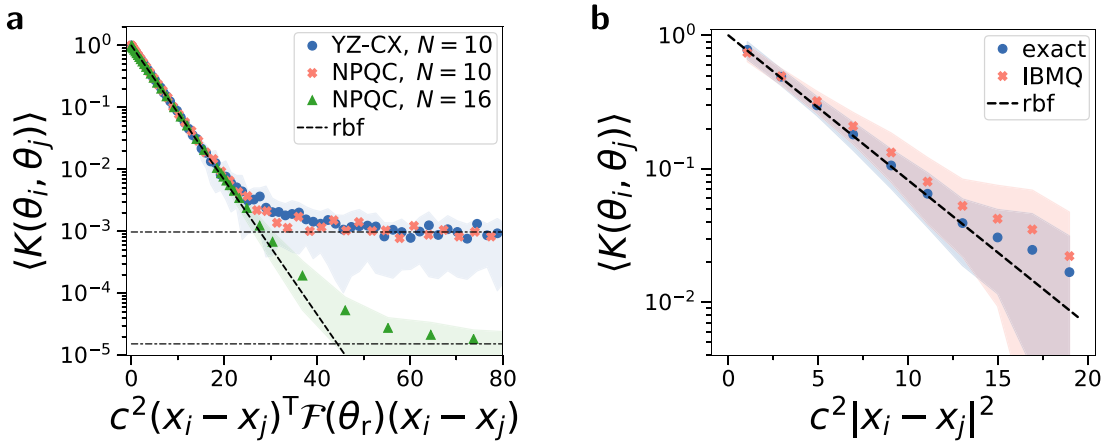

Figure 2. (a) Simulated kernel $K(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j)$ as function of $(\boldsymbol{x}_i - \boldsymbol{x}_j)^{\mathrm{T}}\mathcal{F}(\boldsymbol{\theta}_r)(\boldsymbol{x}_i - \boldsymbol{x}_j)$, which is the distance in feature space weighed with the QFIM $\mathcal{F}(\boldsymbol{\theta}_r)$. The feature vectors $\boldsymbol{x}_i$ are sampled uniformly from $[-1, 1]^M$, then the norm of $\boldsymbol{x}_i$ is rescaled to a randomly chosen $|\boldsymbol{x}_i|$ and encoded with equation (2) into the PQC. We show two types of hardware efficient PQCs, namely YZ-CX PQC and NPQC (see appendix A). Shaded area is the standard deviation of the kernel. The quantum kernels are well approximated by radial basis function kernels (RBF, dashed line, equation (5)) until reaching very small values $K_{\min} = 2^{-N}$ (dash-dotted lines). PQCs have $N = 10$ qubits, $d = 10$ layers and we average over 50 random feature vectors. (b) Experimental kernel $K(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j)$ as function of distance between the feature vectors. We encode uniformly sampled $\boldsymbol{x}_i$ in the NPQC with the QFIM $\mathcal{F}(\boldsymbol{\theta}_r) = I$. The quantum kernel generated by theory (blue dots) and via experimental results with IBM quantum computer (orange crosses) follows approximately the isotropic radial basis function kernel ($K(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j) = \exp(-\frac{c^2}{4}|\boldsymbol{x}_i - \boldsymbol{x}_j|^2)$, black line). Shaded area is standard deviation of the kernel. The NPQC has $N = 8$ qubits, $M = 36$ features and $d = 4$ layers. Experimental results from *ibmq_guadalupe* were performed with $r = 50$ randomly chosen measurement settings, $s = 8192$ measurement samples and error mitigation with equation (8).

6. Results

We now proceed to numerically and experimentally demonstrate our methods. First, we investigate the kernel of our encoding. In figure 2(a) we numerically simulate [48, 49] two types of hardware efficient PQCs (YZ-CX PQC and NPQC) and show that the quantum kernel is well described by a RBF kernel (equation (5), dashed line). The kernel diverges from the RBF kernel for exponentially small values of the kernel and reaches a plateau at $K_{\min} = \frac{1}{2^N}$, which is the fidelity of Haar random states [50]. In figure 2(b), we experimentally measure the kernel of the NPQC with an IBM quantum computer (*ibmq_guadalupe* [51]) using randomized measurements and error mitigation (equation (8)). We find that the mean value of the kernel matches well with the isotropic RBF kernel. See appendix E for details on the experiment and appendix C for results regarding the YZ-CX PQC.

Next we address the statistical error introduced by estimating the kernel using randomized measurements and global depolarizing noise p. In figure 3(a) we simulate the average error

$$\Delta K = \frac{2}{L(L-1)} \sum_{i=1}^{L} \sum_{j=i+1}^{L} \left| K_{\mathrm{m}}(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j) - K(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j) \right| \tag{9}$$

of measuring the mitigated kernel $K_{\mathrm{m}}(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j)$ using randomized measurements with respect to its exact value $K(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j)$ as function of number of measurement samples s. We find that there is a threshold of samples where the error becomes minimal. This threshold depends on the choice of the number of measurement settings r and number of qubits N. We find that the choice $r = 8$ provides sufficient accuracy for our experiments. We are able to mitigate depolarizing noise to a noise-free level even for high p. In figure 3(b), we show the minimal number of samples $s_{\min}$ required to measure the kernel with an average error of at most $\Delta K < 0.1$ as function of depolarization noise p. The randomized measurement scheme works well even with substantial noise p, where we find a power law $s_{\min} \propto (1-p)^{-2}$.

Now we assess the overall performance of our approach on a practical task. We learn to classify handwritten 2D images of digits ranging from 0 to 9. The dataset contains $L = 1797$ images of 8×8 pixels, where each pixel has an integer value between 0 and 16 [52]. We map the image to $M = 64$ dimensional feature vectors. For the YZ-CX PQC, we use all $M = 64$ features, whereas for the NPQC we perform a principal component analysis to reduce it to $M = 36$ features. We calculate the kernel of the full dataset and use a randomly drawn part of it as training data for optimizing the SVM with Scikit-learn [53]. The accuracy of the SVM is defined as the percentage of correctly classified test data, which are $L_{\text{test}} = 200$ images that have not been used for training. The dataset is rescaled using the training data such that each feature has mean

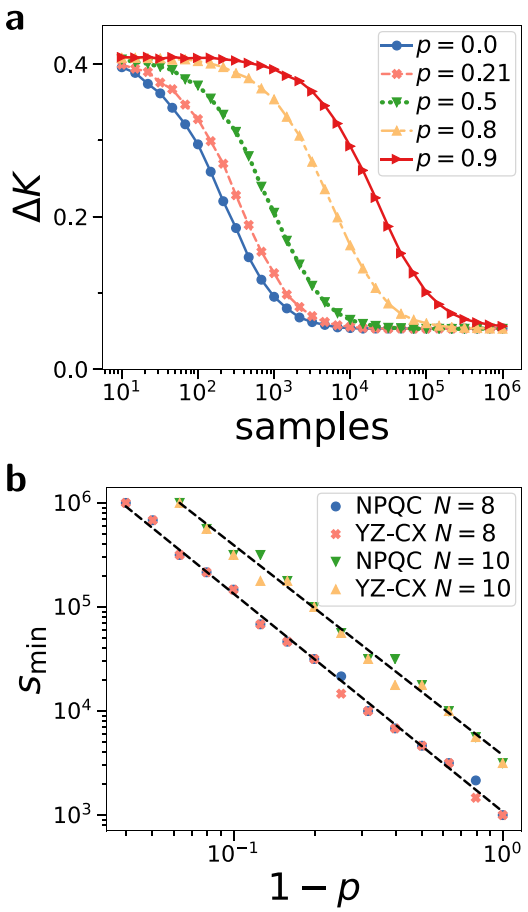

Figure 3. (a) Average error for measuring the kernel with randomized measurements ΔK as function of number of measurement samples s and the global depolarizing probability p. Simulation with $r = 8$ measurement settings, $N = 8$ qubits and the YZ-CX PQC. (b) Minimal number of measurement samples s_{min} needed to achieve an average error of at most $\Delta K < 0.1$ for varying depolarizing noise p. Dashed line is the power law $s_{\text{min}} \propto (1-p)^{-2}$. Number of measurement settings is $r = 8$ for $N = 8$, and $r = 16$ for $N = 10$.

value zero and its variance is given by $\frac{1}{\sqrt{M}}$. We encode the feature vectors $\boldsymbol{x}_i$ via equation (2) with $c = 1$, where for the YZ-CX PQC we choose $\boldsymbol{\theta}_{\text{r}}$ randomly and for the NPQC we define $\boldsymbol{\theta}_{\text{r}}$ such that the QFIM is given by $\mathcal{F}(\boldsymbol{\theta}_{\text{r}}) = I$ (see appendix A).

In figure 4(a), we classify the data by measuring the quantum kernel with a single quantum computer. We plot the accuracy of classifying test data with the SVM against the size of the training data for the YZ-CX PQC and the NPQC. As a classical baseline, we show the RBF kernel. Further, we show a simulations of the exact quantum kernel (exact) and a noiseless simulation of the randomized measurements (noiseless). For experimental data, we use an IBM quantum computer (*ibmq_guadalupe* [51], see appendix E for more details) to perform randomized measurements with error mitigation (mitigated) and without error mitigation (unmitigated). The accuracy improves steadily with increased number of training data for all kernels. Our error mitigation scheme (equation (8)) substantially improves the accuracy of the SVM trained with experimental data to nearly the level of the noiseless simulation of the randomized measurements. The randomized measurements have a lower accuracy compared to the exact quantum kernel as we use only a relatively small number r of randomized measurement settings. For the NPQC, the exact quantum kernel shows nearly the same accuracy as the classical RBF kernel, whereas for the YZ-CX PQC the quantum kernel performs slightly worse compared to the classical kernel, likely indicating that its QFIM does not optimally capture the structure of the data. The depolarizing probability of the IBM quantum computer is estimated as $p \approx 0.36$ for the NPQC and $p \approx 0.39$ for the YZ-CX. To measure the kernel of the dataset with $L_{\text{train}} = 1597$ and $L_{\text{test}} = 200$, we require in total $N_{\text{R}} = s(L_{\text{train}} + L_{\text{test}})r \approx 1.2 \times 10^8$ measurements. For the inversion test, one would require $N_{\text{R}} = s_{\text{c}} L_{\text{train}}(L_{\text{train}} - 1)/2 + s_{\text{c}} L_{\text{train}} L_{\text{test}} \approx 0.8 \times 10^{10}$ experiments, where we have set the number of measurements per kernel entry to $s_{\text{c}} = 5000$ as chosen in past experiments [18]. Thus, we estimate that our method yields a reduction in total measurements by more than factor 60. We find that our method already yields a lower measurement cost for $L_{\text{train}} > 21$ as shown in appendix D.

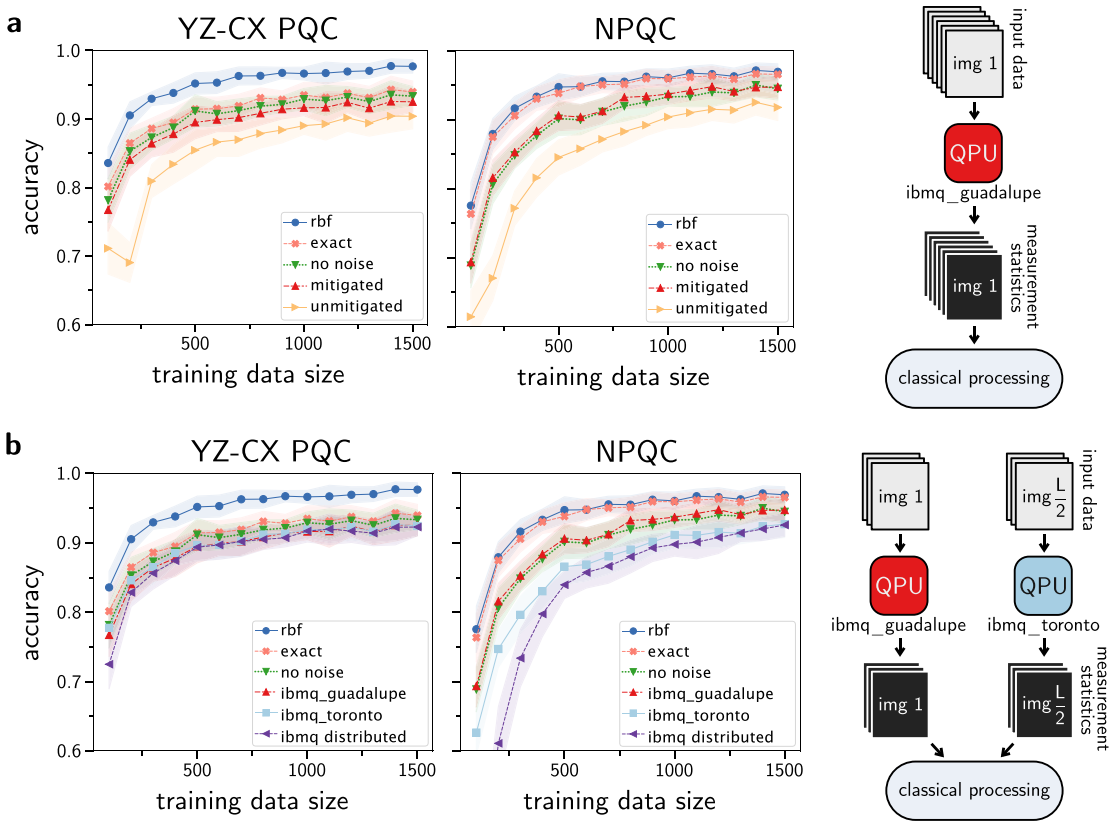

Figure 4. Accuracy of classifying previously unseen handwritten digits correctly as function of the size of the training data. (a) SVM trained with experimental quantum kernel measured on a single quantum computer (*ibmq_guadalupe*) with randomized measurements using error mitigation (red, equation (8)) and no error mitigation (yellow). The shaded area is the standard deviation of the accuracy. As a classical baseline, we show the isotropic radial basis function kernel (blue). Simulations of quantum kernels are the exact quantum kernel (orange) and noiseless simulation of randomized measurements (green). (b) We distribute the measurements on two different quantum computers (*ibmq_guadalupe* and *ibmq_toronto*, purple curve) and post-process the combined measurement results with error mitigation. As reference, we show the accuracy of quantum kernel measured on a single quantum computer for *ibmq_guadalupe* (red) and *ibmq_toronto* (light blue). We encode the data into the YZ-CX PQC with $M = 64$ features and the NPQC with $M = 36$ features. Experiments are performed using $s = 8192$ measurement samples, $N = 8$ qubits and $r = 8$ randomized measurement settings. The test data contains $L_{\text{test}} = 200$. To calculate mean and standard deviation of the accuracy, we randomly draw test and training data from the full dataset 20 times for each training data size.

Finally, in figure 4(b) we distribute the measurements between two quantum computers. We split the dataset into two halves, where one half is measured using randomized measurements with *ibmq_guadalupe* and the other half with *ibmq_toronto* [51] (see appendix E for more details). The measurement outcomes from both machines are then combined for the post-processing on the classical computer to calculate the kernel matrix of the full dataset. Here, we also apply error mitigation. As reference, we also plot the accuracy achieved with a single quantum computer. For the YZ-CX PQC, we find nearly equal accuracy with the distributed and single quantum computer approach. For the NPQC, the accuracy of the distributed approach is slightly lower. The performance highly depends on the noise and calibration of the IBM quantum computers, which can fluctuate over time and highly depends when an experiment is performed. We attribute the lower performance of the distributed YZ-CX approach with a higher noise level present while the experiment was performed on *ibmq_toronto*. As the randomized measurement method correlates measured samples, differences in the respective noise model of the two quantum computers can have a negative effect on the resulting quantum kernel. In the appendices F and G, we show the accuracy of the training data and the confusion matrices.

7. Discussion

Our work demonstrates a practical method to learn large datasets on noisy quantum computers with intermediate qubit numbers. Randomized measurement enables a linear scaling in dataset size L and encodes high-dimensional data with number of features scaling linearly with quantum circuit depth d. We show our encoding can be characterized by the QFIM and its eigenvalues and eigenvectors [35]. As the behavior of the

kernel is crucial for effectively learning and generalizing data, future work could design the QFIM to improve the capability of quantum machine learning models. We demonstrated the NPQC with a simple and exactly known QFIM, which could be a useful basis to study quantum machine learning on large quantum computers.

We encode the data in hardware efficient PQCs, which are known to be hard to simulate classically for large numbers of qubits [12]. This type of PQC has been used in quantum machine learning experiments [18]. While sampling from these circuits is difficult to simulate on classical computers, we find that the quantum kernel closely follow the radial basis function kernel up to exponentially small kernel values [38]. Similarly, many other classes of quantum kernels have efficient classical representations [54]. The resemblance with a classical kernel implies that these quantum kernels are unlikely to achieve an advantage over classical methods [8]. However, we note that radial basis function type of kernels have been of interest in quantum optics [55] and can serve as a reliable benchmark of quantum machine learning methods. Further, the non-trivial weight matrix $\mathcal{F}$ could be of independent interest in machine learning [56].

We mitigate the noise occurring in the quantum computer by using data sampled during the measurements of the kernel. We find that the number $s_{\min}$ of measurement samples needed to mitigate depolarizing noise scales as $s_{\min} \propto (1-p)^{-2}$, allowing us to extract kernels even from very noisy quantum computers. We successfully apply this model to mitigate the noise of the IBM quantum computer. While the noise model of quantum computers is known to be complicated involving multiple types of sources of noise, the depolarizing model we use is sufficient to mitigate the noise of kernels measured on IBM quantum computers [47]. This may be the result of the randomized measurements leading to an insensitivity to fixed unitary noise channels. We note that noise induced errors can actually be beneficial to machine learning as the capability to generalize from data can improve with increasing noise [37].

In general, the number of measurements needed for the randomized measurement scheme scales exponentially with the number N of qubits [41, 42]. This makes our method currently practical only for a lower number of qubits. However, various approaches could extend our method to larger qubit numbers. Importance sampling can reduce the number of measurements needed [44]. For particular types of states an exponential reduction in cost has been observed. It would be worthwhile to study how importance sampling can improve the measurement cost for quantum machine learning. In other settings adaptive measurements have been proposed to improve the scaling of measurement costs [57], as well as other approaches such as shadow tomography [58]. The choice of an effective set of measurements could be included in the machine learning task as hyper-parameters to be optimized. To reduce the number of qubits, one could combine our approach with quantum autoencoders to transform the encoding quantum state into a subspace with less qubits that captures the essential information of the kernel [59]. Alternatively, one could trace out most of the qubits of a many-qubit quantum state $\rho(\boldsymbol{\theta}_i)$ such that a subsystem A with a lower number of qubits remains. Then, randomized measurements can efficiently determine the kernel $\mathrm{Tr}(\rho^A(\boldsymbol{\theta}_i)\rho^A(\boldsymbol{\theta}_j))$. It would be worthwhile to investigate the learning power of kernels generated from subsystems of quantum states that possess quantum advantage [7, 8].

Randomized measurements process each of the L quantum states of the dataset separately [42]. The full kernel matrix $K(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j)$ with L^2 elements is then constructed via classical post-processing using equation (7) where the randomized measurement data for state $|\psi(\theta_i)\rangle$, $|\psi(\theta_j)\rangle$ is reused to calculate each entry of the matrix. This gives us the resulting speedup in quantum computational time. As a further advantage, our approach only requires preparing one quantum state at a time, reducing the number of gates by half compared to the inversion test or swap test. Further, we demonstrate how to achieve additional speedups by distributing measurements across different quantum computers.

The quantum computation time scales linearly with dataset size L and provides a quadratic speedup compared to conventional measurement methods such as the inversion test or swap test. Note that the classical post-processing to construct the kernel still scales as L^2. However, we note that current quantum computers perform measurements at a rate of $\sim$5 kHz [12, 13], which is a factor 10^6 slower than commonly available classical computers. Further, using quantum computers is very expensive compared to classical computation. Thus, the main bottleneck for quantum machine learning algorithms on current quantum hardware lies within the quantum part, while the classical part can be easily parallelized and distributed. Therefore, our work opens up benchmarking quantum machine learning with large datasets on intermediate-size quantum computers, which was impractical with previously known methods.

For our encoding equation (2), at small distances the quantum kernel in parameter space can be described by the QFIM via equation (4). We note this relation is general for any type of PQC. The rank of the QFIM $\mathcal{F}$ indicates the number of independent directions in parameter space with equation (4). The maximal number of independent features $M_{\max}$ that can be encoded via equation (2) is thus given by the rank of the QFIM, which is upper bounded by $\mathrm{rank}(\mathcal{F}) = M_{\max} \leqslant 2^{N+1} - 2$ [35]. Thus, even a modest number of qubits

can represent a large number of parameters. The popular MNIST dataset [60] for classifying 2D images of handwritten digits has 28×28 pixels, which could be encoded in only $N = 9$ qubits.

Assuming 5 kHz measurement rate, $s = 8192$ measurement samples and $r = 8$ measurement settings, our method can process the full MNIST training dataset with $L_{\text{train}} = 60\,000$ entries in about 240 h of quantum processing time of a single quantum computer. In contrast, the inversion or swap test would require at least 10 years with $s = 1000$ samples on a quantum computer. With our scheme, we enable quantum machine learning with large datasets on intermediate-sized quantum computers. Future work could benchmark the performance of currently available quantum computers with datasets commonly used in classical machine learning.

Data availability statement

The code to reproduce the experimental results presented in this paper is available from [61] and the experimental data is available from [62].

The data that support the findings of this study are openly available at the following URL/DOI: https://github.com/chris-n-self/large-scale-qml https://doi.org/10.5281/zenodo.5211695.

Acknowledgment

We acknowledge discussions with Kiran Khosla and Alistair Smith. This work is supported by a Samsung GRC project and the UK Hub in Quantum Computing and Simulation, part of the UK National Quantum Technologies Programme with funding from UKRI EPSRC Grant EP/T001062/1. We acknowledge the use of IBM Quantum services for this work. The views expressed are those of the authors, and do not reflect the official policy or position of IBM or the IBM Quantum team.

Appendix A. Parameterized quantum circuits

Here, we describe the two types of PQCs used in the main text. The PQCs are composed of N qubits and d layers of unitaries. The parameters of the PQC are given by the M-dimensional parameter vector $\boldsymbol{\theta} \in \mathbb{R}^M$. Each layer is described by unitary $U_l(\boldsymbol{\theta}_l)$ with the parameter vector of each layer $\boldsymbol{\theta}_l$ with $\boldsymbol{\theta} = \{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_d\}$. The total PQC is given by $U(\boldsymbol{\theta})|0\rangle = \prod_{l=1}^{d} U_l(\boldsymbol{\theta}_l)|0\rangle^{\otimes N}$. Each layer unitary $U_l(\boldsymbol{\theta}^l)$ is is composed of parameterized single qubit rotations and an unparameterized entangling gate. For each layer, we denote each parameter entry by $\theta_{l,\alpha}^{(k)}$, where l denotes the layer, $\alpha \in \{x, y, z\}$ the type of rotation and k the qubit number. Note this notation is different from the main text.

In figure 5(a), we show the first circuit we use, which we call the NPQC. The first layer is composed of $2N$ single qubit rotations around the y and z axis for each qubit n with $U_1 = \prod_{k=1}^{N} R_z^{(k)}(\theta_{1,z}^{(k)}) R_y^{(k)}(\theta_{1,y}^{(k)})$. Here, $R_\alpha^{(k)}(\theta) = \exp(-i\frac{\theta}{2}\sigma_n^\alpha)$, $\alpha \in \{x, y, z\}$ and σ_n^x, σ_n^y, σ_n^z are the Pauli matrices acting on qubit k. Each additional layer $l > 1$ is a product of two qubit entangling gates and N parameterized single qubit rotations defined as $U_l(a_l) = \prod_{k=1}^{N/2} [R_z^{(2k-1)}(\theta_{l,z}^{(2k-1)}) R_y^{(2k-1)}(\theta_{l,y}^{(2k-1)})] U_{\text{ent}}(a_l)$, where $U_{\text{ent}}(a_l) = \prod_{k=1}^{N/2} \text{CPHASE}\,(2k-1, 2k+2a_l)$ $R_y^{(2k-1)}(\pi/2)$ and CPHASE(n, m) is the controlled σ^z gate for qubit index n, m, where indices larger than N are taken modulo. The entangling layer $U_l(0)$ is shown as example in figure 5(b). The shift factor $a_l \in \{0, 1, \ldots, N/2 - 1\}$ for layer l is given by the recursive rule shown in the following. Initialize a set $A = \{0, 1, \ldots, N/2 - 1\}$ and $s = 1$. In each iteration, pick and remove one element r from A. Then set $a_s = r$ and $a_{s+q} = a_q$ for $q = \{1, \ldots, s - 1\}$. As the last step, we set $s = 2s$. We repeat this procedure until no elements are left in A or a target depth d is reached. One can have maximally $d_{\max} = 2^{N/2}$ layers with in total $M = N(d + 1)$ parameters. The NPQC has a QFIM $\mathcal{F}(\boldsymbol{\theta}_r) = I$, I being the identity matrix, for the reference parameter $\boldsymbol{\theta}_r$ given by

$$\mathcal{F}(\boldsymbol{\theta}_r) = I \quad \text{for} \quad \theta_{r,l,y}^{(k)} = \pi/2\,, \quad \theta_{r,l,z}^{(k)} = 0\,, \tag{A1}$$

where $\theta_{r,l,z}^{(k)}$ is the reference parameter for layer L, qubit k and rotation around z-axis. Close to this reference parameter, the QFIM remains approximately close to being an identity matrix. When implementing the NPQC for the IBM quantum computer, we choose the sift factor a_l such that only nearest-neighbor CPHASE gates arranged in a chain appear. To match the connectivity of the IBM quantum computer, we removed one entangling gate and its corresponding single qubit rotations which require connection between the first and the last qubit of the chain.

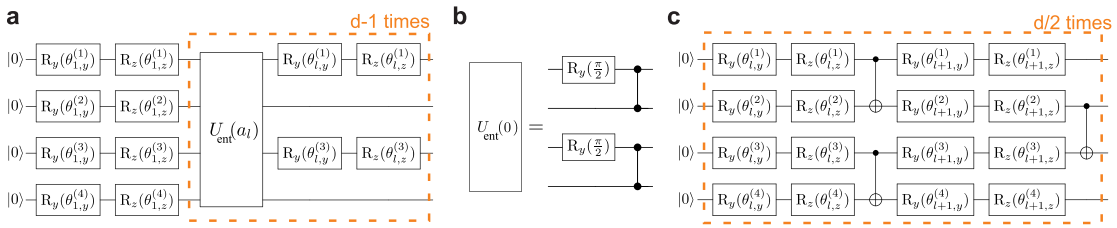

Figure 5. (a) The NPQC for N qubits and d layers, which is a hardware efficient PQC composed of single qubit rotations and CPHASE gates. For the reference parameter $\boldsymbol{\theta}_r$, the QFIM is the identity matrix. (b) Example of the entangling layer for the NPQC, which is composed of $N/2$ non-overlapping CPHASE gates and y rotations by $\pi/2$. (c) YZ-CX PQC, which is a hardware efficient circuit consisting of single qubit rotations and CNOT gates arranged in an alternating fashion in even and odd layers l.

The second type of PQC used is shown in figure 5(c), which we call YZ-CX. It consists of d layers of parameterized single qubit y and z rotations, followed by CNOT gates. The CNOT gates arranged in a one-dimensional chain, acting on neighboring qubits. Every layer l, the CNOT gates are shifted by one qubit. Redundant single qubit rotations that are left over at the edges of the chain are removed.

Appendix B. Methods to measure quantum kernels

In figure 6, we explain the different methods to measure kernels of L quantum states. In this paper, we use the randomized measurements method shown in figure 6(a). The number of required measurements to measure all possible pairs of kernels scales linearly with dataset size L.

The inversion test is shown in figure 6(b). To measure the kernel between two quantum states, it uses the unitary of the first state combined the with inverse unitary of the second state. Then, the kernel is given by the probability of measuring the zero state. Here, the number of measurements scales with the square L^2 of the dataset size.

The swap test is shown in figure 6(c). It prepares both states for the kernel, requiring two times the amount of qubits as with the other tests. Then, a controlled SWAP gate is applied, with the control being on an ancilla qubit. Then, the kernel is given by the measurement of the ancilla. As with the inversion test, the number of required measurements scales with the square L^2 of the dataset size. Further, the controlled SWAP gate can require substantial quantum resources.

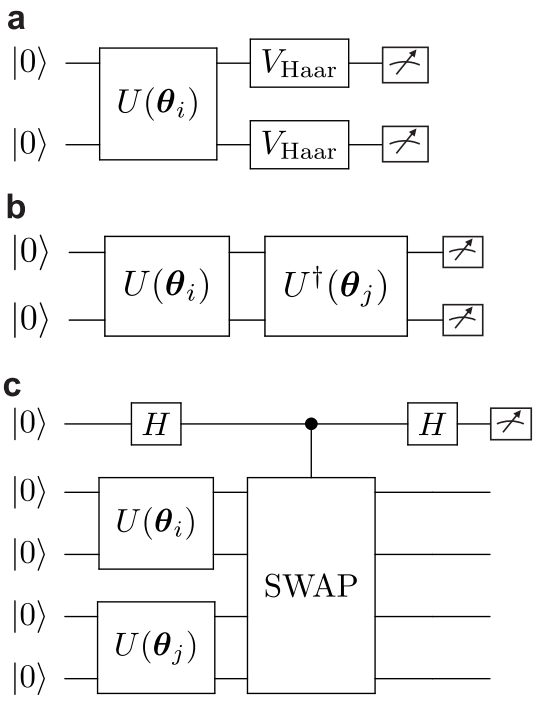

Figure 6. Quantum circuits to measure kernel $K(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j)$ of L quantum states $U(\boldsymbol{\theta}_i)|0\rangle$. (a) Randomized measurement scheme. Prepare state $|\psi(\boldsymbol{\theta}_i)\rangle$, rotate into randomized basis given by single qubit Haar random unitaries V_{Haar} and measure in computational basis. By post-processing sampled states one can extract the kernel. The number of measurements scales linearly with L. (b) Inversion test. Prepare $U^\dagger(\boldsymbol{\theta}_j)U(\boldsymbol{\theta}_i)|0\rangle$ and measure probability of zero state $|0\rangle$. The number of measurements scale with L^2. (c) Swap test. Prepare $U(\boldsymbol{\theta}_j) \otimes U(\boldsymbol{\theta}_i)|0\rangle$ on twice the number of qubits and perform controlled SWAP gate with an ancilla. Kernel is determined by measuring ancilla only. The number of measurements scale with L^2.

Appendix C. Experimental kernel of YZ-CX PQC

We provide further data on the experimental quantum kernel measured on the IBM quantum computer. We measure the kernel using randomized measurements for randomly chosen feature vectors. In figure 7, we show experimental data of the kernel for the YZ-CX PQC using *ibmq_guadalupe*. We find that the experimental data and numerical simulations match well.

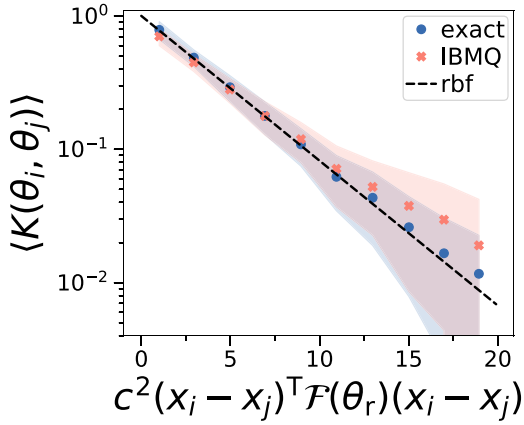

Figure 7. Experimental kernel $K(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j)$ as function of distance between the feature vectors $\boldsymbol{x}_i$, $\boldsymbol{x}_j$. We encode randomly chosen feature vectors via $\boldsymbol{\theta}_i = \boldsymbol{\theta}_r + c\boldsymbol{x}_i$ in the YZ-CX PQC, where $\boldsymbol{\theta}_r \in [0, 2\pi]$ is a randomly chosen reference parameter. The QFIM $\mathcal{F}(\boldsymbol{\theta}_r)$ is calculated numerically. The quantum kernel generated by theory (blue dots) and via experimental results with IBM quantum computer (orange crosses) follows approximately a weighted radial basis function kernel $K(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j) = \exp[-\frac{c^2}{4}(\boldsymbol{x}_i - \boldsymbol{x}_j)^T \mathcal{F}(\boldsymbol{\theta}_r)(\boldsymbol{x}_i - \boldsymbol{x}_j)]$ (black line). Shaded area is standard deviation of the kernel. The YZ-CX PQC has $N = 8$ qubits, $M = 72$ features and $d = 5$ layers. Experimental results from *ibmq_guadalupe* were performed with $r = 50$ randomized measurement settings, $s = 8192$ measurement samples and error mitigation.

Appendix D. Measurement cost

Here, we compare the measurement cost when learning from our dataset for varying number of training data used. For randomized measurements, the number of measurements is given by $N_{\text{meas}}^{\text{random}} = s(L_{\text{train}} + L_{\text{test}})r$. For conventional methods such as SWAP or inversion test, we have $N_{\text{meas}}^{\text{inv}} = s_c L_{\text{train}}(L_{\text{train}} - 1)/2 + s_c L_{\text{train}} L_{\text{test}}$. We now assume that $L_{\text{test}} = L_{\text{train}}/5$. We assume $s = 8192$ and $r = 8$ with the same values as used for experiment of $N = 9$ qubits in the main text. For the conventional approach, we choose $s_c = 5000$ as used in [18] for an experiment with comparable feature vector size. The measurement cost is plotted in figure 8, where we find that randomized measurements is advantageous with $N_{\text{meas}}^{\text{random}} < N_{\text{meas}}^{\text{inv}}$ for $L_{\text{train}} > 21$.

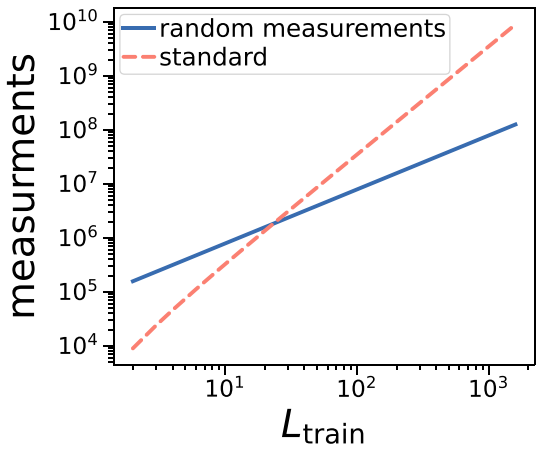

Figure 8. Number of measurements for randomized and standard measurement methods as function of training data L_{train} measured on the quantum computer. We set test data to $L_{\text{test}} = L_{\text{train}}/5$ and use $s = 8192$, $r = 8$ and $s_c = 5000$.

Appendix E. IBM Quantum implementation details

Our PQC circuits are constructed as parameterized circuits with Qiskit [63]. These parameterized circuits are first transpiled then bound for each data point and randomized measurement unitary, ensuring that all circuits submitted have the same structure and use the same set of device qubits. Transpiling is handled by the pytket Python package [64] using *rebase*, *placement* and *routing* passes with no additional optimizations (IBMQ default passes with optimization level 0).

The *ibmq_guadalupe* [51] results presented in figures 2 and 4 were collected between 22 July 2021 and 30 July 2021. The *ibmq_toronto* [51] results presented in figure 4 were collected between 23 July 2021 and 9 August 2021. Figure 2 required the execution of $100 \times 50 = 5000$ circuits and figure 4 involved $1790 \times 8 = 14\,320$ circuits, each with 8192 measurement shots. For comparison, applying the inversion test to the same handwritten digit dataset used for figure 4 would have required the execution of $1790 \times 1790 \approx 3.2 \times 10^6$ circuits. Circuits were executed on IBM quantum devices using the circuit queue API. Job submissions were batched in such a way that all measurement circuits for a data point were submitted and executed together.

Beyond the error mitigation procedure described in the main text we carry out no further error mitigation. In particular, we find that within our experiments readout error mitigation does not yield any significant advantages. We attribute this two possible origins. Our randomized measurement scheme applies random unitaries, which effectively twirl the noise into a form which is easier to mitigate and has been shown to substantially reduce errors [65]. Further, data collected from application of random Pauli operators subject to the same noise has been shown to efficiently correct read-out errors [66]. Similar to this approach, it is possible that our error mitigation scheme also corrects read-out errors at the same time.

Appendix F. Training accuracy

In figure 9, we plot the accuracy of classifying the training data with the SVM for the YZ-CX PQC and NPQC. We show the accuracy for processing on *ibmq_guadalupe*, *ibmq_toronto* and distributing the dataset between both quantum computers. The accuracy is defined as the percentage of training data that is correctly identified. We find that error mitigation substantially increases the accuracy in all cases.

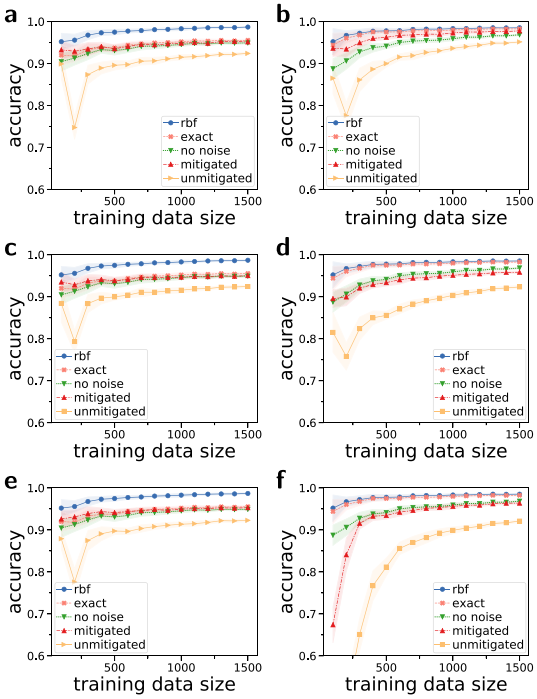

Figure 9. Accuracy of classifying training data as function of the size of the training data. The shaded area is the standard deviation of the accuracy. We use same data as in the main text. We compare the radial basis function kernel (blue dots), exact quantum kernel (orange), noiseless simulation of randomized measurements (green), kernel of IBM quantum computer with randomized measurements using error mitigation (red) and no error mitigation (yellow). (a), (c), (e) YZ-CX PQC with $M = 64$ features and (b), (d), (f) NPQC with $M = 36$ features. We use (a), (b) *ibmq_guadalupe*, (c), (d) *ibmq_toronto* and (e), (f) equally distributing measurements between aforementioned quantum computers. We have $s = 8192$ measurement samples, $N = 8$ qubits and $r = 8$ randomized measurement settings. The training data is randomly drawn from the full dataset, which is repeated 20 times for each training data size.

Appendix G. Confusion matrix

We now show the confusion matrices for the test data. The confusion matrix shows what label is predicted by the SVM in respect to its true label of the test data. The diagonal are the correctly classified digits, whereas the off-diagonals show the number of times a digit was miss-classified. In figure 10, we show the confusion matrix for the NPQC, and in figure 11. We show the confusion matrix for the YZ-CX PQC. We find that the actual digit 8 is often predicted to be the digit 1. Then, likely confusions are that digit 3 is assumed to be 8 and digit 9 is assumed to be 8. We find these confusions consistently in all kernels. While for the NPQC, radial basis function kernel and quantum kernel give nearly the same confusion matrix, we find substantial differences for the YZ-CX PQC. The reason is that while NPQC is an approximate isotropic radial basis function kernel, the YZ-CX PQC is an approximate radial basis function kernel with a weight matrix given by the QFIM. The weight matrix of the YZ-CX seems to reduce the accuracy of the trained SVM.

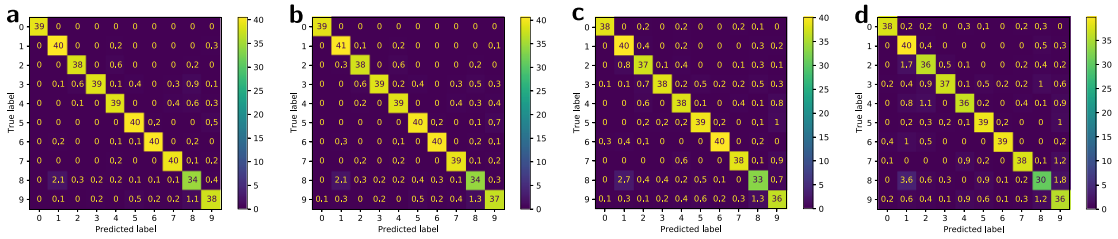

Figure 10. Confusion matrix for the NPQC for (a) radial basis function kernel (b) exact quantum kernel (c) mitigated IBM results with *ibmq_guadalupe* and (d) unmitigated IBM results. We use 1300 training data and 200 test data, where we average the confusion matrix over 100 randomly sampled instances of the data.

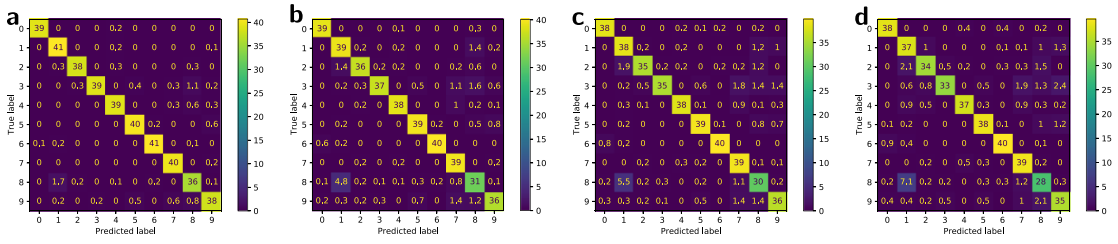

Figure 11. Confusion matrix for the YZ-CX PQC for (a) radial basis function kernel (b) exact quantum kernel (c) mitigated IBM results with *ibmq_guadalupe* and (d) unmitigated IBM results. We use 1300 training data and 200 test data, where we average the confusion matrix over 100 randomly sampled instances of the data.

Appendix H. Product state as analytic radial basis function kernel

As an analytic example, we show that product states form an exact radial basis function kernel. We use the following N qubit quantum state

$$|\psi(\boldsymbol{\theta})\rangle = \bigotimes_{n=1}^{N} \left(\cos\left(\frac{\theta_n}{2}\right)|0\rangle + \sin\left(\frac{\theta_n}{2}\right)|1\rangle \right). \tag{H1}$$

The QFIM is given by $\mathcal{F}(\boldsymbol{\theta}) = I$, where I is the M-dimensional identity matrix and $\mathcal{F}_{ij} = 4[\langle\partial_i\psi|\partial_j\psi\rangle - \langle\partial_i\psi|\psi\rangle\langle\psi|\partial_j\psi\rangle]$. The kernel of two states parameterized by $\boldsymbol{\theta}, \boldsymbol{\theta}'$ is given by

$$K(\boldsymbol{\theta},\boldsymbol{\theta}') = \left|\langle\psi(\boldsymbol{\theta})|\psi(\boldsymbol{\theta}')\rangle\right|^2 = \prod_{n=1}^{N}(1+\frac{1}{2}\cos(\Delta\theta_n)) \tag{H2}$$

where we define $\Delta\boldsymbol{\theta} = \boldsymbol{\theta} - \boldsymbol{\theta}'$ as the difference between the two parameter sets. We now assume $|\Delta\theta_n| \ll 1$ and that all the differences of the parameters are equal $\Delta\theta_1 = \cdots = \Delta\theta_N$. We then find in the limit of many qubits N

$$K(\boldsymbol{\theta},\boldsymbol{\theta}') \approx \prod_{n=1}^{N}(1-\frac{1}{4}\Delta\theta_n^2) \xrightarrow[N\to\infty]{} \exp\left(-\frac{1}{4}\sum_{n=1}^{N}\Delta\theta_n^2\right), \tag{H3}$$

which gives us the radial basis function kernel.

ORCID iD

Tobias Haug https://orcid.org/0000-0003-2707-9962

References

[1] Biamonte J, Wittek P, Pancotti N, Rebentrost P, Wiebe N and Lloyd S 2017 Quantum machine learning *Nature* **549** 195–202
[2] Schuld M and Petruccione F 2018 *Supervised Learning With Quantum Computers* vol 17 (Berlin: Springer)
[3] Schuld M and Killoran N 2019 Quantum machine learning in feature Hilbert spaces *Phys. Rev. Lett.* **122** 040504
[4] Schuld M, Sweke R and Meyer J J 2021 Effect of data encoding on the expressive power of variational quantum-machine-learning models *Phys. Rev.* A **103** 032430
[5] Lloyd S, Schuld M, Ijaz A, Izaac J and Killoran N 2020 Quantum embeddings for machine learning (arXiv:2001.03622)

[6] Weikang Li and Deng D-L 2022 Recent advances for quantum classifiers *Sci. China Phys. Mech. Astron.* **65** 1–23
[7] Liu Y, Arunachalam S and Temme K 2021 A rigorous and robust quantum speed-up in supervised machine learning *Nat. Phys.* **17** 1013–7
[8] Huang H-Y, Broughton M, Mohseni M, Babbush R, Boixo S, Neven H and McClean J R 2021 Power of data in quantum machine learning *Nat. Commun.* **12** 1–9
[9] Huang H-Y, Kueng R, Torlai G, Albert V V and Preskill J 2022 Provably efficient machine learning for quantum many-body problems *Science* **377** eabk3333
[10] Preskill J 2018 Quantum computing in the nisq era and beyond *Quantum* **2** 79
[11] Bharti K *et al* 2022 Noisy intermediate-scale quantum algorithms *Rev. Mod. Phys.* **94** 015004
[12] Arute F *et al* 2019 Quantum supremacy using a programmable superconducting processor *Nature* **574** 505–10
[13] Yulin W *et al* 2021 Strong quantum computational advantage using a superconducting quantum processor *Phys. Rev. Lett.* **127** 180501
[14] Li Z, Liu X, Xu N and Du J 2015 Experimental realization of a quantum support vector machine *Phys. Rev. Lett.* **114** 140504
[15] Bartkiewicz K, Gneiting C, Černoch A, Jiráková Křina, Lemr K and Nori F 2020 Experimental kernel-based quantum machine learning in finite feature space *Sci. Rep.* **10** 1–9
[16] Blank C, Park D K, Rhee J-K K and Petruccione F 2020 Quantum classifier with tailored quantum kernel *npj Quantum Inform.* **6** 1–7
[17] Guan W *et al* 2020 Quantum machine learning in high energy physics *Mach. Learn.: Sci.Technol.* **2** 011003
[18] Peters E, Caldeira J, Ho A, Leichenauer S, Mohseni M, Neven H, Spentzouris P, Strain D and Perdue G N 2021 Machine learning of high dimensional data on a noisy quantum processor *npj Quantum Inform.* **7** 1–5
[19] Wu S L *et al* 2021b Application of quantum machine learning using the quantum kernel algorithm on high energy physics analysis at the LHC *Phys. Rev. Res.* **3** 033221
[20] Havlíček Vĕch, Córcoles A D, Temme K, Harrow A W, Kandala A, Chow J M and Gambetta J M 2019 Supervised learning with quantum-enhanced feature spaces *Nature* **567** 209–12
[21] Johri S, Debnath S, Mocherla A, Singh A, Prakash A, Kim J and Kerenidis I 2020 Nearest centroid classification on a trapped ion quantum computer (arXiv:2012.04145)
[22] Huang H-L *et al* 2020 Experimental quantum generative adversarial networks for image generation (arXiv:2010.06201)
[23] Hubregtsen T, Wierichs D, Gil-Fuster E, Derks P-J H S, Faehrmann P K and Meyer J J 2021 Training quantum embedding kernels on near-term quantum computers (arXiv:2105.02276)
[24] Kusumoto T, Mitarai K, Fujii K, Kitagawa M and Negoro M 2021 Experimental quantum kernel trick with nuclear spins in a solid *npj Quantum Inform.* **7** 1–7
[25] Dutta T, Pérez-Salinas A, Cheng J P S, Latorre J I and Mukherjee M 2021 Realization of an ion trap quantum classifier (arXiv:2106.14059)
[26] Pérez-Salinas A, Cervera-Lierta A, Gil-Fuster E and Latorre J I 2020 Data re-uploading for a universal quantum classifier *Quantum* **4** 226
[27] Schuld M 2021 Quantum machine learning models are kernel methods (arXiv:2101.11020)
[28] Temme K, Bravyi S and Gambetta J M 2017 Error mitigation for short-depth quantum circuits *Phys. Rev. Lett.* **119** 180509
[29] Endo S, Cai Z, Benjamin S C and Yuan X 2021 Hybrid quantum-classical algorithms and quantum error mitigation *J. Phys. Soc. Japan* **90** 032001
[30] Scholkopf B and Smola A J 2018 *Learning With Kernels: Support Vector Machines, Regularization, Optimization and Beyond* (Cambridge, MA: MIT Press)
[31] Wolkowicz H, Saigal R and Vandenberghe L 2012 *Handbook of Semidefinite Programming: Theory, Algorithms and Applications* vol 27 (New York: Springer)
[32] Brandão F G S L, Kalev A, Li T, Lin C Y-Y, Svore K M and Wu X 2017 Quantum SDP solvers: large speed-ups, optimality, and applications to quantum learning (arXiv:1710.02581)
[33] Bharti K, Haug T, Vedral V and Kwek L-C 2022 Noisy intermediate-scale quantum algorithm for semidefinite programming *Phys. Rev.* A **105** 052445
[34] Kandala A, Mezzacapo A, Temme K, Takita M, Brink M, Chow J M and Gambetta J M 2017 Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets *Nature* **549** 242
[35] Haug T, Bharti K and Kim M S 2021 Capacity and quantum geometry of parametrized quantum circuits *PRX Quantum* **2** 040309
[36] Meyer J J 2021 Fisher information in noisy intermediate-scale quantum applications *Quantum* **5** 539
[37] Banchi L, Pereira J and Pirandola S 2021 Generalization in quantum machine learning: a quantum information standpoint *PRX Quantum* **2** 040321
[38] Haug T and Kim M S 2021 Optimal training of variational quantum algorithms without barren plateaus (arXiv:2104.14543)
[39] Goodfellow I, Bengio Y and Courville A 2016 *Deep Learning* (Cambridge, MA: MIT Press)
[40] Haug T and Kim M S 2022 Natural parameterized quantum circuit *Phys. Rev.* A **106** 052611
[41] Elben A, Vermersch Bît, Roos C F and Zoller P 2019 Statistical correlations between locally randomized measurements: a toolbox for probing entanglement in many-body quantum states *Phys. Rev.* A **99** 052323
[42] Elben A, Vermersch Bît, van Bijnen R, Kokail C, Brydges T, Maier C, Joshi M K, Blatt R, Roos C F and Zoller P 2020 Cross-platform verification of intermediate scale quantum devices *Phys. Rev. Lett.* **124** 010504
[43] Zhu D *et al* 2021 Cross-platform comparison of arbitrary quantum computation (arXiv:2107.11387)
[44] Rath A, van Bijnen R, Elben A, Zoller P and Vermersch Bît 2021 Importance sampling of randomized measurements for probing entanglement *Phys. Rev. Lett.* **127** 200503
[45] Buhrman H, Cleve R, Watrous J and De Wolf R 2001 Quantum fingerprinting *Phys. Rev. Lett.* **87** 167902
[46] Nguyen C-H, Tseng K-W, Maslennikov G, Gan H C J and Matsukevich D 2021 Experimental swap test of infinite dimensional quantum states (arXiv:2103.10219)
[47] Vovrosh J, Khosla K E, Greenaway S, Self C, Kim M S and Knolle J 2021 Simple mitigation of global depolarizing errors in quantum simulations *Phys. Rev.* E **104** 035309
[48] Robert Johansson J, Nation P D and Nori F 2012 Qutip: An open-source Python framework for the dynamics of open quantum systems *Comput. Phys. Commun.* **183** 1760–72
[49] Luo X-Z, Liu J-G, Zhang P and Wang L 2020 Yao. jl: Extensible, efficient framework for quantum algorithm design *Quantum* **4** 341
[50] McClean J R, Boixo S, Smelyanskiy V N, Babbush R and Neven H 2018 Barren plateaus in quantum neural network training landscapes *Nat. Commun.* **9** 4812

[51] *ibmq_guadalupe* (v1.3.4) and *ibmq_toronto* (v1.5.7) 2021 IBM Quantum team (available at: https://quantum-computing.ibm.com)
[52] Kaynak C 1995 Methods of combining multiple classifiers and their applications to handwritten digit recognition *Unpublished Master Thesis* Bogazici University
[53] Pedregosa F *et al* 2011 Scikit-learn: Machine learning in Python *J. Mach. Learn. Res.* **12** 2825–30
[54] Schreiber F J, Eisert J and Meyer J J 2022 Classical surrogates for quantum learning models (arXiv:2206.11740)
[55] Chatterjee R and Ting Y 2016 Generalized coherent states, reproducing kernels, and quantum support vector machines (arXiv:1612.03713)
[56] Musavi M T, Ahmed W, Chan K H, Faris K B and Hummels D M 1992 On the training of radial basis function classifiers *Neural Netw.* **5** 595–603
[57] García-Pérez G, Rossi M A C, Sokolov B, Tacchino F, Barkoutsos P K, Mazzola G, Tavernelli I, and Maniscalco S 2021 Learning to measure: adaptive informationally complete POVMs for near-term quantum algorithms (arXiv:2104.00569)
[58] Huang H-Y, Kueng R and Preskill J 2020 Predicting many properties of a quantum system from very few measurements *Nat. Phys.* **16** 1050–7
[59] Romero J, Olson J P and Aspuru-Guzik A 2017 Quantum autoencoders for efficient compression of quantum data *Quantum Sci. Technol.* **2** 045001
[60] Deng Li 2012 The MNIST database of handwritten digit images for machine learning research [best of the web] *IEEE Signal Process. Mag.* **29** 141–2
[61] Self C and Haug T Code for large-scale quantum machine learning (available at: https://github.com/chris-n-self/large-scale-qml)
[62] Self C and Haug T Data for large-scale quantum machine learning (available at: https://doi.org/10.5281/zenodo.5211695)
[63] Abraham H *et al* 2019 Qiskit: an open-source framework for quantum computing
[64] Sivarajah S, Dilkes S, Cowtan A, Simmons W, Edgington A and Duncan R 2020 tket: a retargetable compiler for NISQ devices *Quantum Sci. Technol.* **6** 014003
[65] Wallman J J and Emerson J 2016 Noise tailoring for scalable quantum computation via randomized compiling *Phys. Rev.* A **94** 052325
[66] van den Berg E, Minev Z K and Temme K 2022 Model-free readout-error mitigation for quantum expectation values *Phys. Rev.* A **105** 032620