Training quantum embedding kernels on near-term quantum computers

Thomas Hubregtsen,[1,*] David Wierichs,[2] Elies Gil-Fuster,[1] Peter-Jan H. S. Derks,[1] Paul K. Faehrmann,[1] and Johannes Jakob Meyer[1,3]

[1]*Dahlem Center for Complex Quantum Systems, Freie Universität Berlin, 14195 Berlin, Germany*
[2]*Institute for Theoretical Physics, University of Cologne, 50937 Cologne, Germany*
[3]*QMATH, Department of Mathematical Sciences, University of Copenhagen, 2100 Copenhagen, Denmark*

(Received 25 May 2021; accepted 12 September 2022; published 20 October 2022)

Kernel methods are a cornerstone of classical machine learning. The idea of using quantum computers to compute kernels has recently attracted attention. *Quantum embedding kernels (QEKs)*, constructed by embedding data into the Hilbert space of a quantum computer, are a particular quantum kernel technique that is particularly suitable for noisy intermediate-scale quantum devices. Unfortunately, kernel methods face three major problems: Constructing the kernel matrix has quadratic computational complexity in the number of training samples, choosing the right kernel function is nontrivial, and the effects of noise are unknown. In this work, we addressed the latter two. In particular, we introduced the notion of trainable QEKs, based on the idea of classical model optimization methods. To train the parameters of the QEK, we proposed the use of kernel-target alignment. We verified the feasibility of this method, and showed that for our experimental setup we could reduce the training error significantly. Furthermore, we investigated the effects of device and finite sampling noise, and we evaluated various mitigation techniques numerically on classical hardware. We took the best performing strategy and evaluated it on data from a real quantum processing unit. We found that using this mitigation strategy demonstrated an increased kernel matrix quality.

DOI: 10.1103/PhysRevA.106.042431

I. INTRODUCTION

Over the past decade, machine learning (ML) has taken great steps in tackling some of the world's most important challenges across industries and organizations. Neural networks are at the forefront of this development, being the state-of-the-art solution in fields such as image recognition, speech-to-text, and self-driving cars. They have also shown to achieve impressing feats that were hard to imagine years ago [1–3]. These advancements overshadow the fact that, for many use-cases found in industries as diverse as finance, biology or genetics, other ML methods, such as Bayesian methods or kernel methods, are still the go-to solution.

Classical computers are hitting fundamental limits in their scaling and manufacturing. This fact urged many researchers to explore alternative ways of computing. The intricate control of physical systems at their smallest scales allows the exploitation of the peculiarities of quantum mechanics to perform computations. Quantum computing promises to harness these effects to solve problems that are currently intractable for classical computers. While long considered more a dream than a reality, recent efforts have succeeded in constructing quantum devices able to perform computations intractable for classical computers [4]. This generation of quantum devices is referred to as *noisy intermediate-scale quantum* (NISQ) devices. Exploiting the nonclassical capabilities of NISQ devices to solve practically relevant problems is a rapidly growing field of research [5,6].

Because of the importance of both fields, the application of quantum computing to machine learning has received a lot of attention recently, resulting in many research activities in the field dubbed *quantum machine learning* (QML). The most prominent approach to construct learning models using NISQ devices relies on the use of *parametrized quantum circuits* (PQCs) [7–10]. Many works have likened the properties of PQCs to those of *properties of PQCs to those of* (NNs), even calling these *quantum neural networks* (QNNs). However, also in QML, kernel methods remain prominent, and received a significant deal of attention [11–17]. Furthermore, it was recently shown that other types of variational quantum learning models are fundamentally related to quantum kernel methods [18,19] and that quantum kernels enable the construction of learning problems that prove a separation between classical and quantum machine learning [16,20].

To perform prediction, one would map datapoints in a nonlinear fashion from the input space to a feature space, where a linear separation assigns the output label. Kernel methods, thanks to the kernel trick, can bypass this, allowing for efficient computation. However, they come with three major causes of concern. The first concern is that constructing the kernel matrix has quadratic computational complexity in the number of training samples. Second, choosing the right kernel function is nontrivial. Finally, the effects of noise, inherently present in NISQ device, are unknown, as are potential mitigation strategies. We discuss kernel methods and their disadvantages in more detail in Sec. II.

*thubregtsen@zedat.fu-berlin.de

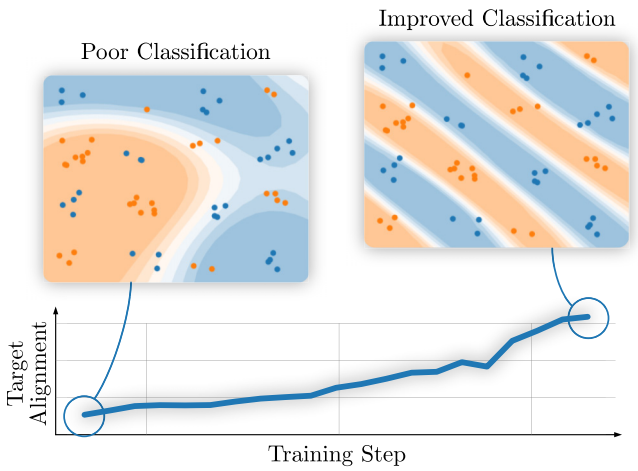

FIG. 1. Quantum embedding kernels allow for classification of datasets through the use of a support vector machine. The quality of classification can be significantly improved by optimizing the parameters of the quantum embedding to increase the kernel-target alignment introduced in Sec. III.

To combat the issue of choosing the right kernel function, we port the notion of model optimization from the classical world to the quantum world in the form of trainable quantum embedding kernels (QEKs). To train the parameters of the QEKs, we propose the use of kernel-target alignment, another method ported from the classical domain. We verify the feasibility to our proposed technique by means of a numerical experiment, that is tailored to evaluate the trainability of trainable QEKs. We illustrated the effects of training a QEKs using kernel-target alignment in Fig. 1. We formally introduce and discuss the trainable QEK in Sec. III

Finally, we discuss the different types of noise in Sec. IV. Here, we look into device noise and finite sampling noise, as well as mitigation strategies. We simulate our proposed strategies on classical hardware. We will then evaluate the best performing strategy on data from a real *quantum processing unit* (QPU). We finish this paper with a summary and outlook in Sec. V

II. KERNEL METHODS

In this section, we will revisit classical kernel methods, quantum embedding kernels, and the associated disadvantages.

A. What are kernel methods

To understand what a kernel method does, let us first revisit one of the simplest methods to assign binary labels to datapoints: *linear classification*. Imagine a set of points that lie in different parts of a plane. We want to construct a classifier that successfully predicts the classes of the datapoints. A linear classifier corresponds to drawing a line and assigning different labels $y = \pm 1$ representing the two classes to the opposing sides of the line. Mathematically, this notion can be formalized by introducing a vector $\boldsymbol{w}$ orthogonal to the line, thus determining its direction. We can then assign the class

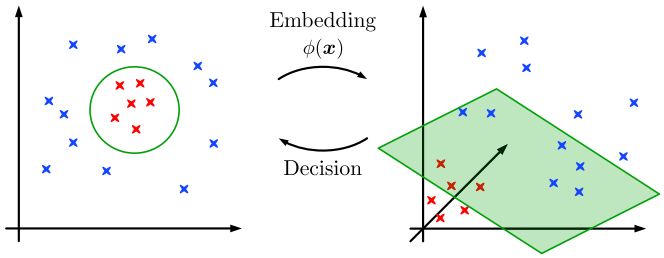

FIG. 2. A nonlinear embedding can be used to enhance the capabilities of a linear classifier. Linear classification in the embedding space can realize nonlinear decision boundaries in the original space.

label y to a datapoint $\boldsymbol{x}$ via

$$y(\boldsymbol{x}) = \mathrm{sgn}(\langle \boldsymbol{w}, \boldsymbol{x} \rangle + b), \tag{1}$$

where the *intercept* term b specifies the position of the line in the plane. The same works for higher dimensional spaces too, where the vector $\boldsymbol{w}$ does not just define a line, but a hyperplane. It is immediately clear that this method is not very powerful, as datasets that are not separable by a hyperplane cannot be classified with high accuracy using this scheme.

There is, however, an ingenious way to enhance the capabilities of a linear classifier: One can specify a *feature map* $\phi(\boldsymbol{x})$ that takes datapoints and embeds them into a larger *feature space* and then perform linear classification in this feature space. In doing so, we can actually realize nonlinear classification in the original space of our datapoints. This strategy is visualized in Fig. 2. We can modify the linear classifier of Eq. (1) to include the feature map:

$$y(\boldsymbol{x}) = \mathrm{sgn}(\langle \boldsymbol{w}', \phi(\boldsymbol{x}) \rangle + b), \tag{2}$$

where $\boldsymbol{w}'$ lives in the feature space corresponding to the feature map ϕ.

A major result in kernel theory is the *representer theorem* [21]. It states that one can write the vector $\boldsymbol{w}'$ that defines an optimal decision boundary as a sum of the embedded datapoints with real coefficients: $\boldsymbol{w}' = \sum_i \alpha_i \phi(\boldsymbol{x}_i)$.[1] Inserting this into Eq. (2) yields

$$y(\boldsymbol{x}) = \mathrm{sgn}\left[\sum_i \alpha_i \langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}) \rangle + b \right]. \tag{3}$$

While this might not seem useful at first, notice that the above formula only contains inner products between vectors in the embedding space,

$$k(\boldsymbol{x}, \boldsymbol{x}') = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x}') \rangle. \tag{4}$$

We call the function k the *kernel* associated to the feature map ϕ. But why do kernels deserve all the attention they get?

The relevant insight is that we can often find a formula for the kernel k without explicitly performing the feature map ϕ. Consider, for example, the embedding

$$\phi : (x_1, x_2) \mapsto \left(x_1^2, \sqrt{2} x_1 x_2, x_2^2\right), \tag{5}$$

[1]The representer theorem makes mild assumptions about the way we measure "optimal," but for our applications these are always fulfilled.

whose associated kernel can be explicitly calculated:

$$k(\boldsymbol{x}, \boldsymbol{x}') = \langle \phi(x_1, x_2), \phi(x_1', x_2') \tag{6}$$

$$= x_1^2 x_1'^2 + 2x_1 x_2 x_1' x_2' + x_2^2 x_2'^2 \tag{7}$$

$$= (x_1 x_1' + x_2 x_2')^2 \tag{8}$$

$$= \langle \boldsymbol{x}, \boldsymbol{x}' \rangle^2. \tag{9}$$

We find that it can be obtained by simply computing the inner product of two vectors *in the initial space* and squaring it. Implicitly, we are however computing the inner product relative to the embedding ϕ, i.e., in feature space! This is the central property of kernel-based methods. In many relevant cases the embedding will require a much higher cost to compute than the kernel, while one still gains access to the larger feature space through the kernel. This implicit use of the embedding through its associated kernel is known as the *kernel trick*.

If we do not need the embedding at all, then how can we determine if a given function k is actually a kernel for some feature map? This question is answered by checking the *Mercer condition*, which states that any function fulfilling

$$\sum_{i,j} c_i c_j k(\boldsymbol{x}_i, \boldsymbol{x}_j) \geqslant 0 \tag{10}$$

for *all* possible sets of real coefficients $\{c_i\}$ and sets of datapoints $\{\boldsymbol{x}_i\}$ is a kernel. Alternatively, we can check whether the kernel matrix K with entries

$$K_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j) \tag{11}$$

associated with *any* dataset $\{\boldsymbol{x}_i\}$ is always *positive semidefinite*.

If we now come back to the example of linear classification in a feature space that motivated our introduction to kernel methods, then it is natural to ask how we can best choose the separating hyperplane. The most common strategy and application for kernel methods is the support vector machine *support vector machine* (SVM). The idea behind SVMs is to find the hyperplane with the maximal *margin*. The margin describes the distance of the dataset on either side of the hyperplane. Intuitively, a larger margin is better, as less ambiguity exists for borderline cases. As input, the SVM takes the kernel matrix from Eq. (11) and delivers the values α_i and b for Eq. (3) that correspond to the decision boundary with the maximal margin.

To predict a class label for a new datapoint, we need to calculate the kernel with respect to the training datapoints and decide on a class label, as shown in Eq. (3). A strong advantage of SVMs is that usually only few weights $\{\alpha_i\}$ contribute significantly to the sum in Eq. (3). We can thus make a prediction by calculating the kernel with respect to these datapoints from the training dataset. The corresponding datapoints are the eponymous *support vectors*—as they support the decision boundary. Intuitively, we can imagine that comparing a new datapoint only to points close to the decision boundary will give important information about the class label.

Kernel methods are not limited to classification. In fact, one can take any ML technique that can be reformulated in terms of inner products and replace the inner products by kernel functions to get a "kernelized" variant. This leads to interesting applications such as kernel principal component analysis [22] or kernel ridge regression [23].

B. Quantum embedding kernels

On NISQ hardware, we make use of quantum gates, like Pauli rotations, to load data onto the quantum computer. This constitutes a quantum circuit that is represented by a unitary operation dependent on the specific datapoint, $U(\boldsymbol{x})$. As soon as the data is loaded, the quantum system is in a state that depends on the datapoint in question,

$$|\phi(\boldsymbol{x})\rangle = U(\boldsymbol{x})|0\rangle. \tag{12}$$

This approach is also known as a *quantum feature map* [12] because we are effectively embedding the datapoint in the Hilbert space of quantum states. As we learned in Sec. II, it is no large step from a feature map to a kernel function. We only need to take the inner product between quantum states, which is given by the *overlap*

$$k(\boldsymbol{x}, \boldsymbol{x}') = |\langle \phi(\boldsymbol{x}')|\phi(\boldsymbol{x})\rangle|^2. \tag{13}$$

This is the QEK associated to the quantum feature map $|\phi(\boldsymbol{x})\rangle$.

In general, we are not able to avoid noise, which means that we cannot assume that the embedded quantum state is pure. In this case, the quantum embedding is realized by a data-dependent density matrix $\rho(\boldsymbol{x})$, which for a pure state reduces to $\rho(\boldsymbol{x}) = |\phi(\boldsymbol{x})\rangle\langle\phi(\boldsymbol{x})|$. The inner product is given by

$$k(\boldsymbol{x}, \boldsymbol{x}') = \mathrm{Tr}\{\rho(\boldsymbol{x})\rho(\boldsymbol{x}')\}. \tag{14}$$

This inner product is also known as the Hilbert-Schmidt inner product for matrices. For pure quantum states, this reduces to Eq. (13).

In summary, any quantum feature map induces a QEK. We can use this kernel as a subroutine in a classical kernel method, for example, the SVM, which yields a hybrid quantum-classical approach. In this case, the separating hyperplane is constructed in a purely classical manner, only the kernel function between the training datapoints is evaluated on the quantum computer.

To be able to use QEKs in this way, we need to evaluate the overlap of two quantum states on near-term hardware. There are a number of advanced algorithms to estimate the overlap of two quantum states [24–28]. All these algorithms work for arbitrary states, and so they are agnostic to how the states were obtained by necessity. By exploiting the structure and specifics of QEKs, though, we can do better.

For unitary quantum embeddings, i.e., embeddings resulting in a pure quantum state, this is straightforward if we can construct the adjoint of the data-encoding circuit, $U^\dagger(\boldsymbol{x})$. In this case, we can rewrite the overlap as

$$|\langle \phi(\boldsymbol{x}')|\phi(\boldsymbol{x})\rangle|^2 = |\langle 0|U^\dagger(\boldsymbol{x}')U(\boldsymbol{x})|0\rangle|^2. \tag{15}$$

This is nothing but the probability of observing the $|0\rangle$ state when measuring the state $U^\dagger(\boldsymbol{x}')U(\boldsymbol{x})|0\rangle$ in the computational basis. To obtain an estimate, we can therefore initialize the quantum system in the $|0\rangle$ state, apply the unitary operation $U(\boldsymbol{x})$ followed by $U^\dagger(\boldsymbol{x}')$, and finally measure in the computational basis. From there, we only need to record the frequency with which the prepared state is found in the $|0\rangle$ state to obtain our estimate. The circuit diagram for this *adjoint* approach

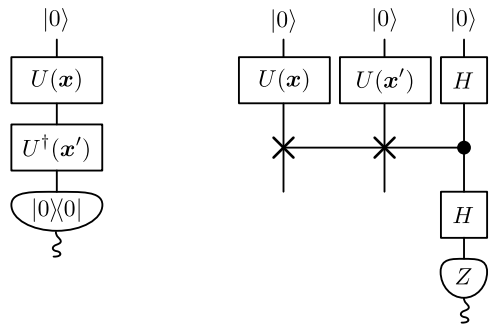

FIG. 3. There are two principal ways to compute the overlap between embedded states. On the left, we see the adjoint approach valid for pure states. This approach results in a doubled circuit depth but does not need auxiliary qubits. The overlap between the embedded states can also be computed via the SWAP test which is depicted on the right. This approach results in a doubled circuit width and requires one additional qubit and controlled SWAP operations but it also works mixed states.

can be found on the left of Fig. 3. This scheme does not need auxiliary qubits, yet it applies the data-encoding circuit twice (or the adjoint thereof). This way, while the width of the circuit for the adjoint approach does not increase from the data-encoding one, the depth is doubled.

Another alternative approach to estimate the overlap between two quantum states is the *SWAP test*. The SWAP test is based on the *SWAP trick*, a mathematical gimmick that allows us to transform the product of the density matrices in Eq. (14) into a tensor product:

$$k(\boldsymbol{x}, \boldsymbol{x}') = \mathrm{Tr}\{\rho(\boldsymbol{x})\rho(\boldsymbol{x}')\} \tag{16}$$

$$= \mathrm{Tr}\{(\rho(\boldsymbol{x}) \otimes \rho(\boldsymbol{x}'))\mathbb{S}\}, \tag{17}$$

where $\mathbb{S}$ denotes the SWAP operation between the two quantum systems in the states $\rho(\boldsymbol{x})$ and $\rho(\boldsymbol{x}')$. To extract this quantity, the SWAP test makes use of an auxiliary qubit that controls the SWAP operation. The exact circuit is depicted on the right of Fig. 3. While this approach needs auxiliary qubits and a quantum computer roughly twice as wide, its depth increases only ever so slightly, and it also works for mixed quantum states. If the deeper requirements of the adjoint approach were too limiting, then the SWAP test would be a natural alternative.

As quantum feature maps occur in many applications of NISQ computing, it is no surprise that QEKs are instrumental beyond their use as subroutines for classical machine learning methods. In Ref. [18] it was shown that most variational quantum learning methods boil down to kernel methods, providing an opening for kernel theory to explore the properties of these learning models.

C. Shortcoming of current kernel methods

While we have now seen that kernel methods can enhance the power of many ML techniques, kernel methods have three downsides.

The first downside stems from computational complexity theory. For the application of kernel methods, the kernel matrix with respect to the input data needs to be constructed—which has quadratic complexity in the number of datapoints. This can already constitute a substantial impediment in the world of big data where the number of datapoints can be in the millions.

A second downside is that the selection of a suitable kernel for a given problem is a nontrivial task. For any given task, different kernel functions yield varying performance. The kernel function ought to be a measure of similarity, so a wrong choice can mark similar points as nonsimilar, or the other way around. Errors at this level will invariably translate into poorer classification. Two extreme examples for such under-performing kernel functions would be

$$k_1(\boldsymbol{x}, \boldsymbol{x}') = 1, \tag{18}$$

$$k_\delta(\boldsymbol{x}, \boldsymbol{x}') = \delta_{\boldsymbol{x},\boldsymbol{x}'}, \tag{19}$$

here δ is Kronecker's δ, which is 1 when both arguments are equal, and 0 otherwise. The first kernel k_1 will mark every pair of points as similar. The second one k_δ will mark them all as nonsimilar. The *radial basis function* (RBF) *kernel* also known as *Gaussian kernel* given by

$$k_{\mathrm{RBF}}(\boldsymbol{x}, \boldsymbol{x}') = \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{x}'\|^2}{2\sigma^2}\right) \tag{20}$$

is often a generally accepted starting point, but even there the parameter σ that quantifies how close datapoints need to be considered similar needs to be fine-tuned. In the next sections, we will introduce trainable QEKs, a notion that is known as model optimization in the classical world, but has not yet been applied in the quantum domain. It is our hope that training the kernel on specific data will improve the modeling capabilities of the algorithm.

The third downside stems from the need of a kernel matrix to remain positive semidefinite, something that is hard to guarantee under the presence of noise from NISQ devices. In following sections, we will investigate noise mitigation strategies to address this issue.

III. TRAINABLE QUANTUM EMBEDDING KERNELS

In the previous section we discussed QEKs. To address the difficulty of finding the right kernel, we propose the use of trainable QEKs, a concept known as model optimization in the classical world. In this section, we discuss how to adjust these parameters to improve the classification capabilities of quantum embedding kernels.

Adjusting the variational parameters of a quantum feature map—and therefore of its associated QEK—can be seen as a particular instance of *model selection*. This process consists of two steps, *kernel selection* and *kernel optimization*. Kernel selection is choosing a particular quantum circuit *layout*, or *ansatz*, with certain variational parameters. Here, we refer to a *variational family* of kernels, much like the family of RBF kernels defined in Eq. (20) where the standard deviation σ is varied. Kernel optimization corresponds to the process of fixing the variational parameters to ensure a good classification performance on our target data.

Because of the unfavorable scaling, exhaustive parameter search procedures are only suitable for optimizing few parameters. It is therefore sensible to resort to a proxy quantity that

is easier to compute but still acts as a predictor for classification accuracy. Reference [29] provides an overview of such measures as provided in Refs. [30–35], using the *kernel-target alignment* from Ref. [36] as central building block. We propose to use kernel-target alignment for optimizing the QEK.

A. Kernel-target alignment

We will first explain this procedure for *balanced* datasets, i.e., datasets that have an equal number of elements in each class. At the end of this section, we will generalize to unbalanced datasets. The central idea behind the kernel-target alignment is that the *labels* for the training set can be seen as an instance of a very particular kernel, which acts as an oracle that always outputs the correct similarity for two datapoints:

$$k^*(\boldsymbol{x}, \boldsymbol{x}') = \begin{cases} 1 & \text{if } \boldsymbol{x} \text{ and } \boldsymbol{x}' \text{ in same class,} \\ -1 & \text{if } \boldsymbol{x} \text{ and } \boldsymbol{x}' \text{ in different classes.} \end{cases} \tag{21}$$

Of course, in general we do not have access to this *ideal* kernel, but on the training data it is given by the training labels and the kernel matrix predicted by this ideal kernel has entries

$$K^*_{ij} = y_i y_j. \tag{22}$$

This means that if we place the labels into a vector $\boldsymbol{y}$, we can express the ideal kernel matrix as the outer product of that vector with itself:

$$K^* = \boldsymbol{y}\boldsymbol{y}^T. \tag{23}$$

To get to a measure of how well a kernel captures the nature of the training dataset we need a way to compare the kernel matrix with the ideal one. To obtain the kernel-target alignment we will make use of geometric reasoning. Remember that we can measure the *alignment* of two vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ by evaluating and normalizing the inner product:

$$\mathrm{A}(\boldsymbol{a}, \boldsymbol{b}) = \frac{\langle \boldsymbol{a}, \boldsymbol{b} \rangle}{\sqrt{\langle \boldsymbol{a}, \boldsymbol{a} \rangle \langle \boldsymbol{b}, \boldsymbol{b} \rangle}}. \tag{24}$$

The alignment is related to the angle $\sphericalangle(\boldsymbol{a}, \boldsymbol{b})$ between the vectors as $\cos \sphericalangle(\boldsymbol{a}, \boldsymbol{b}) = \mathrm{A}(\boldsymbol{a}, \boldsymbol{b})$, which means that the alignment is a quantity that ranges from -1 for vectors pointing in exactly opposite directions to $+1$ for vectors pointing in exactly the same direction.

We can apply the same reasoning to kernel matrices. To do so, we need to define an inner product between two matrices. For the definition of the kernel-target alignment we will use the *Frobenius inner product*. For that, we simply treat the matrices as if they were column vectors, with every entry of the matrix being a separate entry of the vector. This means that the inner product is just

$$\langle A, B \rangle_F = \sum_{ij} A_{ij} B_{ij} = \mathrm{Tr}\{A^T B\}, \tag{25}$$

from where one defines the alignment of two matrices B and B' as

$$\mathrm{A}(B, B') = \frac{\langle B, B' \rangle_F}{\sqrt{\langle B, B \rangle_F \langle B', B' \rangle_F}}. \tag{26}$$

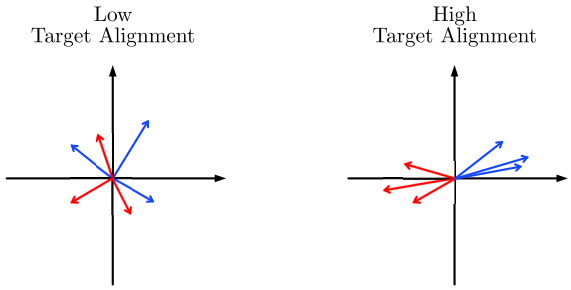

FIG. 4. The kernel-target alignment is high if the feature vectors corresponding to datapoints in the same class cluster together and the feature vectors of points in the opposite class lie exactly opposite to them. This illustrates that a high kernel-target alignment allows for easier linear separability.

Now we have all the ingredients to define the kernel-target alignment:

$$\mathrm{TA}(K) = \mathrm{A}(K, K^*) = \frac{\langle K, K^* \rangle_F}{\sqrt{\langle K, K \rangle_F \langle K^*, K^* \rangle_F}}. \tag{27}$$

We can equivalently express this in terms of the kernel function and the training dataset:

$$\mathrm{TA}(k) = \frac{\sum_{ij} y_i y_j k(\boldsymbol{x}_i, \boldsymbol{x}_j)}{\sqrt{\left(\sum_{ij} k(\boldsymbol{x}_i, \boldsymbol{x}_j)^2\right)\left(\sum_{ij} y_i^2 y_j^2\right)}} \tag{28}$$

$$= \frac{\sum_{ij} y_i y_j k(\boldsymbol{x}_i, \boldsymbol{x}_j)}{n\sqrt{\sum_{ij} k(\boldsymbol{x}_i, \boldsymbol{x}_j)^2}}, \tag{29}$$

where we used the fact that for all labels $y_i^2 = 1$ and n denotes the number of points in the training set. Note that $\mathrm{TA}(k) \geqslant 0$ because of the positive semidefinite nature of the kernel function.

At the beginning of this section we assumed that the training set was balanced, i.e., that it contains the same number of datapoints for each class. If this were not the case, then the approach we just outlined would run into problems, because the contributions from one class would dominate the kernel-target alignment. We could however mitigate this by simply rescaling the labels, dividing them by the number of samples available in their class. In this case, we cannot use Eq. (29) but have to stay with Eq. (28).

We can gather further intuition why the kernel-target alignment is a meaningful measure by looking at the numerator of Eq. (28), $\sum_{ij} y_i y_j k(\boldsymbol{x}_i, \boldsymbol{x}_j)$. This quantity is also known as the *kernel polarity*. Each term $y_i y_j k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ in the sum is a product of the kernel function of two points and their labels. If both points belong to the same class, $y_i y_j = +1$, then the kernel value increases the kernel-target alignment, whereas if the labels are different $y_i y_j = -1$, then the term decreases it. Increasing the kernel-target alignment therefore means both increasing the kernel values for datapoints from the same class and decreasing them for datapoints in different classes. Figure 4 visually illustrates how a large kernel-target alignment allows for easier linear classification of the training data. Beyond this intuition, the kernel-target alignment profits from theoretical guarantees regarding both its high concentration

about the expected value and its good generalization behavior in labeling previously unseen data [35–37].

With the kernel-target alignment we now have a measure that we can use as a cost function to maximize through a hybrid quantum-classical optimization loop [38]. At every iteration, the QPU is used to evaluate the kernel matrix K, recall constructing K has quadratic complexity in the number of datapoints.

It turns out that optimizing the kernel-target alignment is closely related to the "quantum metric learning" approach put forward in Ref. [39] that analyzed different strategies to optimize quantum feature maps. Indeed, the Hilbert-Schmidt distance-based method is the same as optimizing the unnormalized kernel-target alignment, the polarity. We detail the connection in Appendix A.

B. Trainability

To verify the feasibility of our proposed setup, and to perform a numerical analysis of the trainability of a trainable QEK, we have run a set of experiments. In these experiments, we look at two scenarios. First, we look at the error percentage of an optimized SVM on a QEK with random parameter values. We will refer to this as the untrained QEK. Second, we look at the error percentage of an optimized SVM on a trained QEK. We will refer to this as the trained QEK. From this, we expect to see that a trained QEK is able to better fit the training data compared to an untrained QEK. Below, we give details on the particular numerical experiments conducted.

1. Dataset

For our dataset, we have used the "Bank Marketing Data Set" [40]. We preprocess the data in various steps. We first convert all nonnumerical columns by assigning an increasing integer value to every new label, e.g., a column containing {"yes," "no," "maybe," "yes"} is converted to {0, 1, 2, 0}. We then normalize all columns to the range $[-1, 1]$. To reduce dimensionality, we then perform Principal Component Analysis to reduce the dataset to 12 features. This is the minimal amount of features that still ensures no two entries are reduced to a single sample. Finally, we sample 30 datapoints in a balanced way, both for our training and validation set.

2. Ansatz

As explained above, a QEK is specified by a PQC that embeds one input datapoint. The PQCs we use are organized in layers with fixed gate layout, which are implemented sequentially. Each layer starts with a block of Hadamard gates; then, a block of Pauli-Z rotations whose rotation angles are the data coordinates; next, one block of trainable Pauli-Y rotations; and finally, one *ring* of controlled Pauli-Z rotations. A sketch of one such layer can be seen in Fig. 5. This ansatz can be used for arbitrary number of qubits and layers. We use this layout following the data re-uploading ideas from Ref. [41], which shows that ansätze of this type form rich function families. It is important to use expressive enough designs so that we can potentially achieve the linear separability of the data.

We select an ansatz to solve both: kernel selection and kernel optimization. Kernel selection is addressed by having chosen a parametrized kernel that can be scaled up both in number of qubits and number of layers. Then, maximizing the target-kernel alignment corresponds to kernel optimization. The flexibility in the number of qubits and layers allows us to increase the model expressivity in a controlled way. That is, we are more interested in making sure the models are expressive enough than in encoding a specific inductive bias.

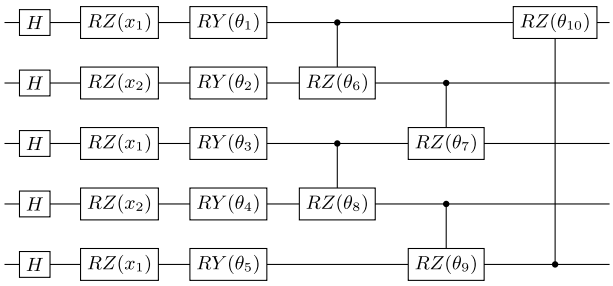

FIG. 5. Circuit diagram of the elementary building layer used in the QEK ansatz for $N = 5$ qubits and $m = 2$ features. The trainable parameters of the ansatz are denotes as $\{\theta_j\}_{j=1}^{s=10}$ and $x_{1,2}$ are the data features.

3. Hyperparameters

In every of our 51 runs, we use a different seed to sample 30 datapoints in a balanced fashion. We embed the data in an ansatz consisting of five qubits and six layers. To train the QEK, we use 2000 optimization iterations, in which we optimize using six samples. We report the error percentage, defined as $1 - \frac{\text{misclassifications}}{\text{total samples}}$.

4. Results

The raw data and code used to run the experiment can be found on GitHub [42]. The untrained QEK has an error percentage of 18.8%. The trained QEK has an error percentage of 13.3%. This is a reduction in error percentage of 29.3%. Both experiments had a variance of 0.003.

5. Discussion and conclusion

To evaluate the trainability of a trainable QEK, we performed the experiment discussed above. This experiment serves as a first proof of concept. As with all numerical experiments, changes to the ansatz used, hyperparameters set, and data selected, caution needs to be taken in interpreting the outcome. However, for our specific setting, we can conclude that the performance significantly improved.

As a side remark, we also wanted to express our intent to evaluate generalization capability. However, the 30 samples of training data that our computational resources could process, proved not to be sufficient to learn any meaningful information regarding validation data. When sanity checked with a classical linear, RBF and sigmoid kernel, all performing close to random guessing.

IV. THE EFFECTS OF NOISE

Noise is one of the namesakes of NISQ devices and considering the effects of noise is therefore of utmost importance. In this section, we discuss how both noise arising from imperfect

quantum operations—device noise—and noise arising from finite sampling of expectation values affects QEKs and how it can be mitigated. We then proceed with a series of simulations that predict the performance of our proposed noise mitigation techniques. Last, we use the best noise mitigation technique, as found in our experiments, and evaluate it on data from a real QPU. The code for these experiments can be found on GitHub [42].

A. Device noise

NISQ devices suffer from unavoidable noise caused by unintentional interactions with the environment or imperfect control. It is thus not possible to prepare pure quantum states with an embedding circuit. This fact has multiple implications for QEKs and their realization.

Noise can be modeled by *quantum channels*. Formally, any map that takes valid quantum states to valid quantum states can be seen as a quantum channel. An example is *depolarizing noise*, which corresponds to a complete loss of information about the underlying quantum state with a certain probability $1-\lambda$. We formally realize it by replacing the system's quantum state with the maximally mixed state with probability $1-\lambda$:

$$\mathcal{D}_\lambda[\rho] = \lambda\rho + (1-\lambda)\frac{\mathbb{I}}{2^N}. \tag{30}$$

Depolarizing noise is a popular model for noise in quantum systems, as it is simple and subsumes other, more nuanced noise models.

As we have seen in Sec. II B, a QEK can also be defined for mixed states for which its output corresponds to the state overlap. The SWAP test can be directly employed to compute the overlap of mixed states, but often we would like to use the adjoint method due to its lower qubit requirements (see Fig. 3). The adjoint method, however, needs more consideration, because the implementation of the adjoint noisy embedding circuit itself is not straightforward. But if we fail to implement the correct adjoint operation, we are no longer computing the overlap of the quantum embedding states and therefore also do not compute a valid kernel!

In the noiseless embedding circuit, all operations are unitary and typically the adjoint of every elementary operation is available. This becomes apparent when we consider that any quantum circuit can be constructed from controlled NOT gates, which are self-adjoint, and single-qubit Pauli rotations, whose adjoint is obtained by performing the same rotation with negated angle.

The device noise, however, can in principle prevent us from implementing the adjoint embedding. As an example, consider a quantum channel that represents a noisy Pauli rotation gate, $\mathcal{V}(\theta)$. We model it by the original rotation gate $R(\theta)$ that is followed by a noise channel $\mathcal{N}$. The channel $\mathcal{N}$ could model imprecision in the control of the rotation angle, unwanted interactions with the environment or other noise processes, but we will leave it arbitrary for this example. The noisy gate is then given by

$$\mathcal{V}(\theta)[\rho] = \mathcal{N}[R(\theta)\rho R^\dagger(\theta)], \tag{31}$$

and its adjoint reads

$$\mathcal{V}(\theta)^\dagger[\rho] = R(-\theta)\mathcal{N}^\dagger[\rho]R^\dagger(-\theta). \tag{32}$$

But how would we implement $\mathcal{V}(\theta)^\dagger$? The very nature of a noise channel implies that we cannot control it or choose at which time the noise occurs. Instead, we have to work with the noisy quantum gates at our disposal, which means that we can only approximate $\mathcal{V}(\theta)^\dagger$ by $\mathcal{V}(-\theta)$:

$$\mathcal{V}(-\theta)[\rho] = \mathcal{N}[R(-\theta)\rho R^\dagger(-\theta)]. \tag{33}$$

In general, this approximation is not equal to the adjoint of the noisy unitary. This only happens to be the case if the noise channel $\mathcal{N}$ both is self-adjoint and commutes with the unitary operation $R(\theta)$. Both conditions hold for the depolarizing noise introduced in Eq. (30).

Let us now take a step back and look at actual NISQ devices. They are usually programed at the gate-level, assuming perfect unitaries. The adjoint of a perfect unitary circuit is readily available, but only if the behavior of the available NISQ device is well-modeled by depolarizing noise can we expect this "naive" adjoining of the unitary gates to still compute the overlap of embedded states for the QEK.

B. Mitigating device noise

Mitigating the effects of device noise is very important to make NISQ computers useful in practice. It is therefore no surprise that the topic has gained a lot of attention and that many techniques have been developed to mitigate device noise [43–49]. In the following, we will complement these with an approach that exploits the very definition of the quantum embedding kernel and that can be freely combined with other mitigation approaches.

We have introduced depolarizing noise as a rather general approach to model the noise in quantum devices. We will model the noise with $\mathcal{D}_\lambda$ as in Eq. (30), where the depolarizing channel is assumed to act homogeneously on the whole system. We will refer to λ—the probability that the depolarizing channel does not cause a loss of information about the underlying state—as *survival probability*. Note that it may well be possible that the probabilities λ_i differ for distinct embedded datapoints $\boldsymbol{x}_i$, as one might need longer pulse sequences to be embedded than the other, causing more noise.

We now assume that the embedding is composed of this noise channel and the noiseless unitary embedding:

$$\rho_\lambda(\boldsymbol{x}) = \mathcal{D}_\lambda[|\phi(\boldsymbol{x})\rangle\langle\phi(\boldsymbol{x})|]. \tag{34}$$

We can then explicitly compute the impact of the depolarizing noise on a kernel matrix entry:

$$K_{ij}^{(\mathrm{dev})} = \mathrm{Tr}\left\{\rho_{\lambda_i}(\boldsymbol{x}_i)\rho_{\lambda_j}(\boldsymbol{x}_j)\right\} \tag{35}$$

$$= \mathrm{Tr}\left\{\lambda_i\rho(\boldsymbol{x}_i)\lambda_j\rho(\boldsymbol{x}_j) + (1-\lambda_i)\lambda_j\frac{\rho(\boldsymbol{x}_j)}{2^N} + \lambda_i(1-\lambda_j)\frac{\rho(\boldsymbol{x}_i)}{2^N} + (1-\lambda_i)(1-\lambda_j)\frac{\mathbb{I}}{2^{2N}}\right\} \tag{36}$$

$$= \lambda_i\lambda_j K_{ij} + (1-\lambda_i\lambda_j)\frac{1}{2^N}. \tag{37}$$

Here we used $\mathrm{Tr}\{\rho(\boldsymbol{x})\} = 1$ and $\mathrm{Tr}\{\mathbb{I}\} = 2^N$.

We can exploit the fact that all diagonal entries of the noiseless kernel matrix K are known to be 1. While we could use this knowledge to save quantum computational cost, we propose to instead use it to gather information about the device noise. We can use Eq. (37) to infer the survival probability λ_i from the diagonal element of the noisy kernel matrix $K_{ii}^{(\mathrm{dev})}$:

$$\lambda_i = \sqrt{\frac{K_{ii}^{(\mathrm{dev})} - 2^{-N}}{1 - 2^{-N}}}. \tag{38}$$

With those values at hand we can recover the noiseless kernel matrix entries

$$K_{ij} = \frac{K_{ij}^{(\mathrm{dev})} - 2^{-N}(1 - \lambda_i\lambda_j)}{\lambda_i\lambda_j}. \tag{39}$$

We denote this mitigation strategy as M-SPLIT. We can distill two even simpler mitigation strategies from this approach by assuming that all λ_i have the same value. This value can then be estimated by averaging multiple of the λ_i obtained from Eq. (38), a strategy which we denote as M-MEAN and which requires fewer diagonal elements to be measured. Alternatively, we can choose to further save resources and only measure one diagonal entry to estimate the survival probability, which we denote as M-SINGLE. There are a number of options for which entry to use, for the sake of simplicity and reproducibility we always use the first entry.

More details on the presented mitigation strategies can be found in Appendix C 1.

C. Finite sampling noise

Recall Eq. (13), where we first introduced the definition of QEKs. And, critically, notice from Eq. (15) that we proposed the so-called adjoint method for estimating the overlap: a frequentist way of approximating the kernel function from quantum circuit evaluations. Measuring such kernel functions results in *independent and identically distributed* (i.i.d) Bernoulli random variables $\hat{k}_{ij}$, since each circuit evaluation outputs either a 1, in case the observed state is $|0\rangle$, or a 0 otherwise. By construction, the theoretical kernel value K_{ij} is the true mean of this random variable, i.e., $\mathbb{E}(\hat{k}_{ij}) = K_{ij}$. Since it follows from Born's rule that an infinite number of circuit evaluations would be required to pin down the exact kernel value, there exists a second source of noise originating from using a finite number of samples.

In reality, we can only estimate the kernel function from a finite number of experimental runs. How many runs we can afford is limited by our experimental resources. This incurs uncertainty beyond the device noise, especially if the number of runs is small.

Going one step further, notice that the pipeline involves estimating the entire kernel matrix, comprising $n(n-1)/2 = O(n^2)$ independent entries. To gauge the number of required circuit evaluations M_{tot} to reach a desired error ϵ in operator distance of an estimator to the target kernel matrix, we can use results from random matrix theory.

The difference between an estimator constructed using M circuit evaluations *per entry*, $(\bar{K}_M)_{ij} = \sum_{s=1}^{M} \hat{k}_{ij}^{(\mathrm{s})}/M$, and the target kernel matrix K is given by

$$(\bar{E}_M)_{ij} = (\bar{K}_M)_{ij} - (K)_{ij} = \frac{1}{M}\sum_{s=1}^{M} \hat{k}_{ij}^{(\mathrm{s})} - K_{ij}. \tag{40}$$

Remembering that

$$\mathbb{E}\{(\bar{E}_M)_{ij}\} = 0, \tag{41}$$

$$\mathbb{E}\{[(\bar{E}_M)_{ij}]^2\} = O\left(\frac{1}{M}\right), \tag{42}$$

$$\mathbb{E}\{[(\bar{E}_M)_{ij}]^4\} = O\left(\frac{1}{M^2}\right) \tag{43}$$

allows us to make use of the following result:

Theorem 1 (Latala's theorem [50,51]). Let A be a random matrix whose entries a_{ij} are independent centered random variables with finite fourth moment. Then, for $C > 0$,[2]

$$\mathbb{E}\{\|A\|\} \leqslant C\left[\max_i \left(\sum_j \mathbb{E}\{a_{ij}^2\}\right)^{1/2} + \max_j \left(\sum_i \mathbb{E}\{a_{ij}^2\}\right)^{1/2} + \left(\sum_{ij} \mathbb{E}\{a_{ij}^4\}\right)^{1/4}\right]. \tag{44}$$

For the $n \times n$-dimensional error matrix $\bar{E}_M$ with moments as in Eqs. (41)–(43), this leads to

$$\mathbb{E}\{\|\bar{E}_M\|\} = O\left(\frac{\sqrt{n}}{\sqrt{M}}\right). \tag{45}$$

Consequently, $M = O(n/\epsilon^2)$ measurements per kernel matrix entry are required to ensure an error of ϵ in operator distance. As a result, since we need to estimate $O(n^2)$ entries of the kernel matrix, we require a total of

$$M_{\mathrm{tot}} = O\left(\frac{n^3}{\epsilon^2}\right) \tag{46}$$

circuit evaluations to reach the desired accuracy. A short calculation for Gaussian variables using Bai-Yin's law [51,52] verifies that this scaling is indeed asymptotically optimal, since the error of kernel matrix entries converges to the Gaussian distribution according to the central limit theorem.

The corresponding constant prefactors can be obtained via involved methods using results from random matrix theory for matrices with sub-Gaussian rows [51]

Although having results for the required number of circuit evaluations to ensure a desired error of the kernel matrix is sufficient for the presented scheme to work, another important quantity we need to estimate for quantum embedding kernels is the kernel target alignment introduced in Eq. (28), which we use as loss function in the training phase of our algorithm.

Allowing for an error of at most ϵ in the kernel target alignment, propagation of uncertainty via partial derivatives suggests that $O(1/\epsilon^2)$ measurements per kernel entry are required, leading to $O(n^2/\epsilon^2)$ circuit evaluations in total (see

[2] C is a constant depending only on the sub-Gaussian norm of the entries.

Appendix E. This is a direct consequence of the fact that kernel matrix entries enter the definition of the kernel target alignment both in the nominator and denominator.

D. Mitigating finite sampling noise

Due to the imperfect sampling outcome for the kernel matrix and the device noise mitigation techniques introduced in Sec. IV B, the obtained matrix might not be positive semidefinite. However, we know the exact kernel matrix to be positive semidefinite and this property is a requirement for the matrix to be used in a classification task. We may therefore regularize the obtained matrix, validating it as kernel matrix and bringing it closer to the perfect outcome.

We discuss three methods to find a positive semidefinite matrix close to a symmetric matrix A: In the first method called *Tikhonov regularization*, we displace the spectrum of A by its smallest eigenvalue $\sigma_{\min}$ if it is negative, by subtracting it from all eigenvalues or equivalently from the diagonal [53]:

$$\text{R-TIK}(A) = \begin{cases} A - \sigma_{\min}\mathbb{I} & \text{if } \sigma_{\min} < 0 \\ A & \text{else} \end{cases}, \tag{47}$$

which yields a positive semidefinite matrix. While being formally the same as the original method by Tikhonov [54], we use it here to assure positive semidefiniteness instead of nonsingularity of the matrix.

The second method called *thresholding* only changes the negative eigenvalues of A by setting them to zero [55]. This is done via a full eigenvalue decomposition, adjustment of the negative eigenvalues and composition of the adjusted spectrum and the original eigenvectors:

$$D = V^T A V, \tag{48}$$

$$D'_{ij} = \max\{D_{ij}, 0\}, \tag{49}$$

$$\text{R-THR}(A) = V D' V^T. \tag{50}$$

This approach is equivalent to finding the positive semidefinite matrix closest to A in any unitarily invariant norm. It is also equivalent to finding the positive semidefinite matrix which has the largest alignment [see Eq. (27)] with A.

The third method extends this reasoning by searching the closest matrix in Frobenius norm, but with the additional requirement that the diagonal elements of the regularized matrix need be one, incorporating our knowledge about the exact kernel as a constraint. This approach constitutes a *semidefinite program* (SDP) and is therefore efficiently computable:

$$\text{R-SDP}(A) = \text{argmin}\{\|A' - A\|_F : A' \succcurlyeq 0,\ A'_{ii} = 1\}. \tag{51}$$

For further details on the computational cost and properties of the regularized matrices please refer to Appendix C 1.

We note that other mitigation techniques proposed in the literature may be combined with the ones discussed here, as they function on different levels of abstraction. An established method to reduce the impact of noise is zero noise interpolation to first order [11,43,56], which makes use of additional circuit evaluations at increased noise rates. Furthermore, techniques to suppress errors by duplicating the circuit have been proposed recently [48,49]. Both of these methods may be used to greatly reduce the noise on the kernel matrix before treating it with regularization and mitigation techniques.

It has been shown that regularization techniques such as those presented here can change some properties of the regularized matrix; see, for example, Ref. [57]. If the matrices underwent very dramatic changes, for example, in regimes of very large noise, then the optimization cycle for the kernel alignment could be disrupted. The study of these effects and when they start to appear lays beyond the scope of this paper. Along the same lines, there are a few different ways in which our error mitigation techniques could be used to label new data after training. In this work, regarding corrections due to noise, we do not tread further than the SVM step of the pipeline.

During the preparation of this work, Wang *et al.* [58] demonstrated that regularization methods can improve the classification accuracy of noisy circuits significantly, more concretely R-TIK, R-THR and flipping the negative eigenvalues of the kernel matrix were covered.

E. Evaluating noise mitigation techniques through simulation

We now investigate the effect of the regularization and device noise mitigation techniques introduced in Secs. IV B and IV D. To this end, we simulate device noise with a model based on gate-level, local depolarizing noise (see Appendix D for details) and test the post-processing performance on the *checkerboard* dataset (see Appendix B 2).

For a range of base survival probabilities λ_0 and measurements M per kernel matrix entry, we first compute 100 noisy, sampled kernel matrices $\{\bar{K}_{M,\ell}\}_{\ell=1}^{100}$. We then consider any combination of up to three methods with the order *regularize-mitigate-regularize*, including combinations that skip one or two of these steps, to post-process each $\bar{K}_{M,\ell}$ into $K_\ell^{(\text{post})}$. The quality of each post-processing strategy is finally assessed via the average change in the alignment with K [see Eq. (26)], relative to the distance of $\bar{K}_M$ from perfect alignment:

$$q = \sum_{\ell=1}^{100} \frac{\text{A}\big(K_\ell^{(\text{post})}\big) - \text{A}(\bar{K}_{M,\ell})}{1 - \text{A}(\bar{K}_{M,\ell})}, \tag{52}$$

where in a slight abuse of notation we abbreviated $\text{A}(K_\ell) \coloneqq \text{A}(K_\ell, K)$, i.e., we skipped the dependence on the exact kernel matrix K.

The best-performing technique per base survival probability λ_0 and number of measurements M is shown together with the achieved improvement q in Fig. 6. For large base survival probabilities λ_0 (low device noise), the combination of M-MEAN and R-SDP consistently is the best approach[3] and yields significantly improved alignments by up to 82.8% in the domain marked in Fig. 6. For smaller λ_0 (higher device noise), however, other post-processing methods become favourable and M-MEAN, R-SDP results in reduced alignments (negative q).

The trivial combination that only uses R-THR performed best for low M (high shot noise) and simultaneously smaller

[3]A minimal manual filter was applied to the results to improve readability, for details please refer to Appendix C 2.

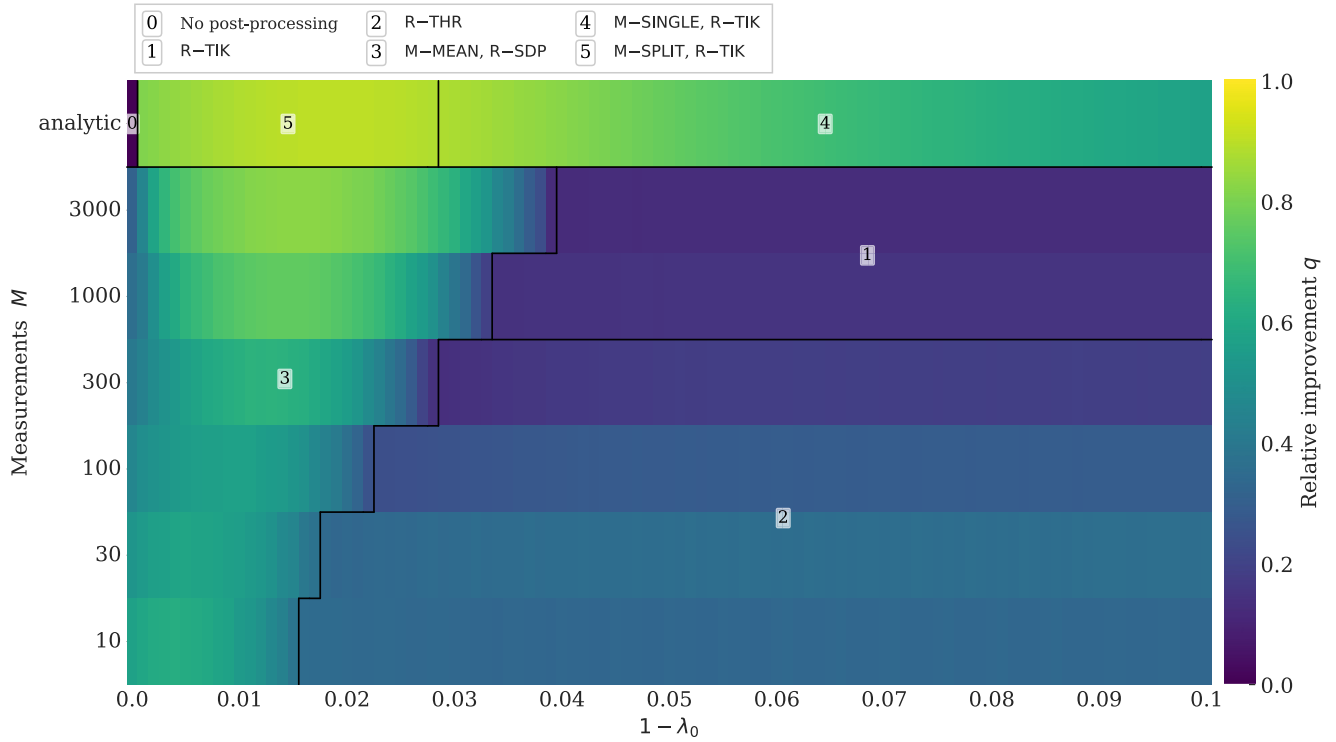

FIG. 6. Relative improvement q in alignment [see Eq. (52)] for various base survival probabilities λ_0 and numbers of circuit evaluations per matrix entry M under a noise model based on local depolarizing noise (see Appendix D). For each λ_0 and M, and for each investigated post-processing combination, we sample 100 kernel matrices, perform the post-processing and compute the average improvement q across the 100 samples (see Appendix C). The labels indicate the best-performing combination for each area[b] and the color map shows the corresponding best average improvement q.

λ_0 (high device noise). It achieved improvements in alignment of up to 38.0% and never decreased the quality of the kernel matrix, that is $q \geqslant 0$ for all tested noise parameters. When increasing M (lower shot noise) while keeping λ_0 small, Tikhonov regularization R-TIK performed better, with the caveat that it decreased the alignment for high λ_0 (low device noise). For the analytic case without shot noise, a combination of either M-SINGLE or M-SPLIT with R-TIK performs best, with q ranging from 54.6% to 90.2%. We include additional details on the performance of all 42 combinations of mitigation and regularization strategies to Appendix C 2.

We observe that post-processing can systematically and significantly enhance the quality of the obtained kernel matrix, in addition to other possible mitigation techniques that may reduce the effective sampling and device noise strengths [11,43,56] or even at the hardware level [48,49].

At the same time, it is important to choose the mitigation and regularization technique adequately as is apparent from the different best post-processing approaches per noise regime in Fig. 6 and the fact that the methods are not guaranteed to not decrease the alignment with the noiseless matrix K. In this regard, R-THR has the advantage of changing the spectrum of $\bar{K}_M$ as little as possible while ensuring positive semidefiniteness.

For NISQ applications we find M-MEAN, R-SDP to be the best choice, assuming that the device noise is not too large. In the following, we will confirm that the above post-processing approach deals well with experimental data from a QPU.

F. Evaluating the proposed noise mitigation technique on a QPU

So far, all experiments in this section were run on classical simulators. We will now proceed to evaluate the post-processing techniques in real-world conditions. To this end, we have computed the kernel matrix for the *symmetric donuts* dataset (see Appendix B 1) using three qubits on an ion trap QPU by IonQ.

For the computation we have used $M = 175$ circuit evaluations per kernel matrix entry and because we measured the diagonal entries for mitigation purposes, the total number of circuit evaluations sums up to about 3.2×10^5. In addition, we sampled kernel matrices for several smaller M from the measured distribution.[4]

Figure 7 shows the alignment $\mathrm{A}(\bar{K}_M)$ between the obtained kernel matrix $\bar{K}_M$ and the noiseless matrix K, as well as the alignment $\mathrm{A}(K^{(\mathrm{post})})$ between the post-processed matrix $K^{(\mathrm{post})}$ and K. For each number of circuit evaluations M, we plot the two best out of the 42 post-processing combinations. Note that various of these combinations yield a quality similar to the best choice. As expected, the quality of the kernel matrix improves with the number of circuit evaluations and as predicted by our simulation results (see Appendix IV E), the

[4]Note that this is not the same as a proper computation on the quantum device with decreased M because we sample from a sample and not from the true distribution directly.

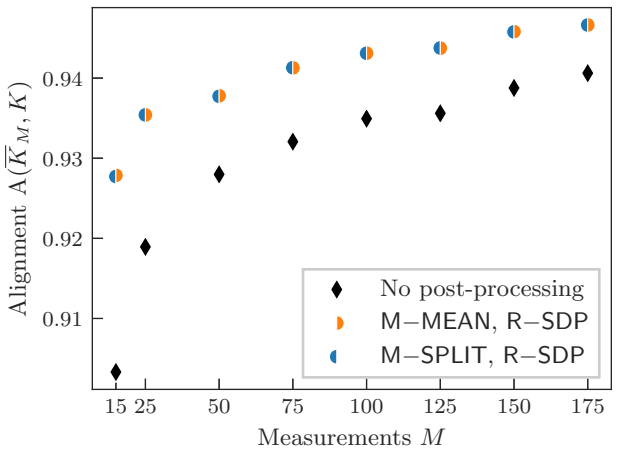

FIG. 7. Alignment A of the kernel matrix measured on the ion trap QPU with the simulated, noiseless kernel matrix K for various numbers of circuit evaluations per matrix entry M, with and without the two best post-processing strategies. Applying our device noise mitigation techniques M-MEAN/M-SPLIT (see Sec. IV B), which assume a simple, global depolarizing noise model, followed by matrix regularization R-SDP results in the highest improvement of the alignment. Here, M-MEAN, R-SDP is the strategy that performed best in the numerical simulations for low device noise rates (see Fig. 6)

post-processing methods increase the alignment significantly. The achieved values for the relative improvement q range between 10.1% and 25.4% with a mean of 14.9%.

We observe that the combination M-MEAN, R-SDP, which is either best or second best by a small margin, was correctly predicted for small device noise levels by our simulations of depolarizing noise. This is remarkable as the simulations were performed on a different dataset, with different circuit depth and width (see Appendix C) and using a different elementary gate set. This indicates that the depolarizing noise model captures properties of the noise in the QPU that are significant for the kernel matrix computation, and suggests that these post-processing methods show robust performance across different circuit depths, qubit numbers and datasets.

In contrast to the above, the method M-SPLIT, R-SDP yields better results than expected for the hardware results, as it showed significantly worse performance than M-MEAN, R-SDP in the simulated experiments for similar numbers of shots (see Appendix C 2 for details). However, we stress that this does not refute the prediction of the latter method as strongest post-processing strategy.

In conclusion, our results on the actual quantum device demonstrate an increased kernel matrix quality when using post-processing, which may allow for improved classification accuracy (also see [58]) or alternatively for a reduced number of circuit evaluations while maintaining a given level of classification performance.

V. SUMMARY AND OUTLOOK

In this work, we have studied the concept of quantum embedding kernel (QEK). To address the difficulty of choosing the right kernel, we transferred the concept of model optimization from the classical world, and introduced trainable QEKs. To optimize variational parameters of the QEKs, we transferred the concept of kernel target alignment to the quantum setting.

To summarize this concept, we summarize our work into a holistic pipeline for working with quantum embedding kernels as depicted in Fig. 8. We start from a parameterized quantum circuit that represents a quantum feature map with variational parameters set to some initial values. To adjust the parameters for a specific dataset, a training loop is run: First, a kernel matrix is obtained from the underlying NISQ device. As an alternative next step, a mitigation and regularization strategy can be applied to improve the quality of the kernel matrix. The kernel matrix is then used to calculate the kernel-target alignment and its gradient with respect to the variational parameters of the parameterized quantum circuit. Gradient descent is then used to update the variational parameters and hence the quantum embedding kernel. This process is repeated until the desired kernel-target alignment is reached.

To perform a classification of new data, a support vector machine is trained using the post-processed kernel matrix of the optimized quantum embedding kernel. The support vectors of the SVM are then extracted and can be used in a support vector classifier to predict labels for new datapoints. To this end, the quantum embedding kernel between the new datapoints and the support vectors have to be computed, but the training of the SVM itself is purely classical.

As QEKs run on noisy quantum devices, they are necessarily affected by two sources of noise—device noise and statistical fluctuations. Using a simple, global depolarizing noise model we proposed device noise mitigation techniques specific for kernel matrices and combined them with matrix regularization methods that exploit positive semidefiniteness of the exact kernel matrix. We tested a total of 42 combinations both on kernel matrices computed with a mixed-state simulator with a gate-based noise model and on matrices measured on quantum hardware. The former provides systematic insights into the performance of the mitigation and regularization techniques for a large range of device noise levels and statistical noise strengths, and indeed shows that the optimal choice varies with these experimental parameters. For low device noise levels, the combination of mitigating via M-MEAN and regularizing with R-SDP is predicted as best choice, using a simple estimate for global depolarizing noise in the circuit and an advanced thresholding of the kernel matrix spectrum. This prediction is confirmed using the matrices measured on a quantum processing unit, demonstrating that post-processing methods can partly recover the noiseless kernel matrix to improve its accuracy or reducing the required number of measurements.

There are two immediate challenges remaining when applying post-processing methods. On the one hand, access to the noiseless matrix is required to rate the methods, requiring us to extrapolate their performance from small to large systems. On the other hand, the impact of the methods on the classification accuracy remains to be investigated.

In the design and training of QEKs, one could explore various aspects. A clear question would be the choice of ansatz families. Some key objects of study for this would be the expressivity of different circuits, the dependence

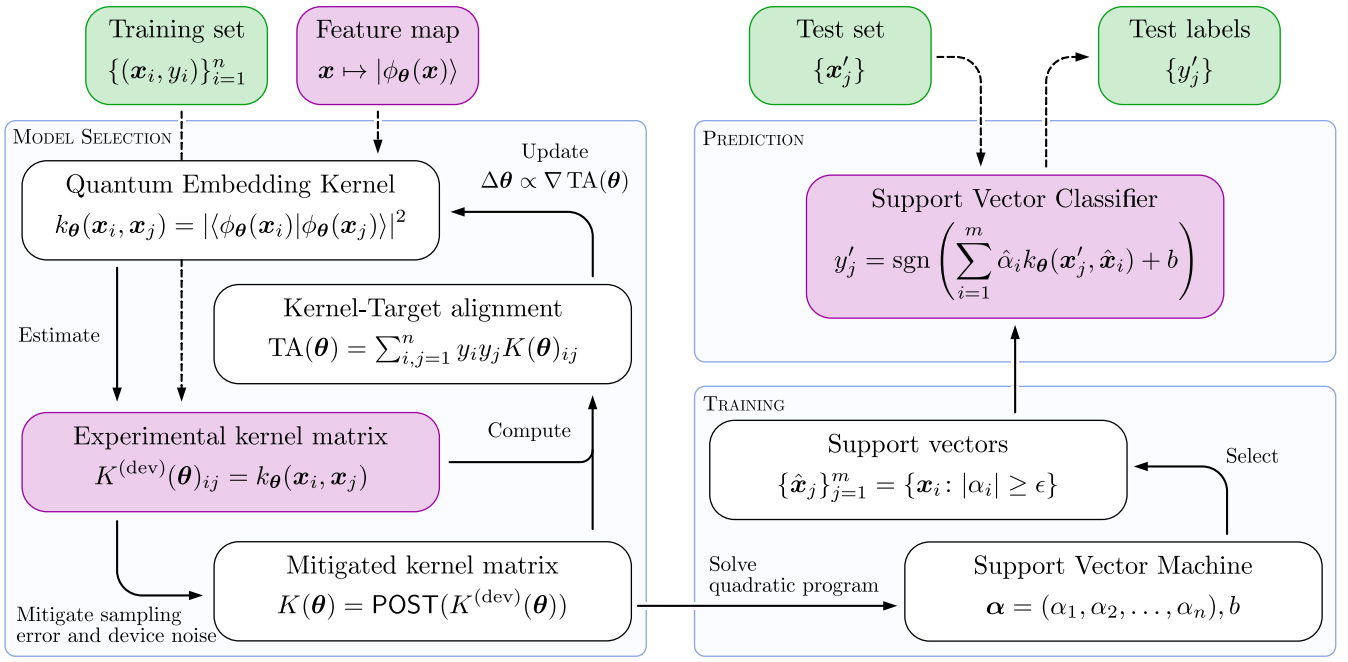

FIG. 8. Schematic of the pipeline used in this work. Green boxes indicate data, purple boxes indicate process steps that are executed on quantum hardware. The pipeline used in this work can be split into three separate parts. In the model selection part, depicted on the left, the parameters of the feature map are adjusted to increase the kernel-target alignment. To calculate the alignment, the kernel matrix is computed and may afterwards be post-processed to mitigate sampling and device noise. After a sufficient target-alignment is reached, the kernel is used to train a support vector machine. The resulting support vector classifier is used in the prediction step to predict labels of new datapoints.

on the dataset, the optimal choice of hyperparameters (or, alternatively, how one could perform empirical risk minimization successfully), or how one would build gauge invariant kernel functions [59]. Another major topic is investigating the effect of the barren plateau phenomenon [60–62] in the kernel setting, and subsequently the study of (quantum-aware) cost function alternatives to the target alignment.

Finally, one could explore whether the proposed model generalizes well to unseen data, and if the model can be transferred to more general tasks such as unbalanced binary classification, multiclass classification, or regression.

ACKNOWLEDGMENTS

The authors thank Xanadu for organizing QHack 2021, where the foundations of this work were laid as part of the Open Hackathon Challenge and the resulting funding. We further thank the AWS team for their support and funding that provided us access to the Rigetti and IonQ devices, as well as Sandbox@Alphabet for alpha access to the Floq cloud service, yielding access to the TPU-based quantum simulator. Additionally, we thank Richard Kueng for valuable input on bounds, as well as Jens Eisert and Maria Schuld for valuable feedback. We endorse Scientific CO_2nduct [63] and provide a CO_2 emission Table I in Appendix F. This work was supported by the BMWi under the PlanQK initiative, the BMBF under the RealistiQ initiative, the Cluster of Excellence MATH+ Project No. EF1-7, the European Flagship project PasQuanS and the DFG under Germany's Excellence Strategy Cluster of Excellence Matter and Light for Quantum Computing (ML4Q) Grant No. EXC2004/1 390534769 and the CRC 183 Project No. B01.

T.H., J.J.M., E.G.F., and P.K.F. worked on trainable QEKs. D.W., P.K.F., and J.J.M. built the theory on noise mitigation. D.W. and P.J.H.S.D. ran the numerics for noise mitigation. J.J.M. supervised the project. All authors contributed to the discussions and to writing the manuscript.

APPENDIX A: CONNECTION TO QUANTUM FEATURE MAP OPTIMIZATION

Optimizing kernels using the kernel-target alignment as a cost function is closely related to the "metric learning" approach put forward for the training of quantum feature embeddings in Ref. [39].

To understand this approach, we first introduce some notation. We consider a dataset $\mathcal{S} = \{(\boldsymbol{x}_i, y_i)\}$ that we split in two parts corresponding to the two classes labeled as ± 1. We denote these subsets as $\mathcal{S}_+$ and $\mathcal{S}_-$, respectively. For a given embedding $|\phi_{\boldsymbol{\theta}}(\boldsymbol{x})\rangle$, we can identify both classes with quantum states—we will refer to them as *class states*—simply by averaging the embedded quantum states

$$\rho_\pm(\boldsymbol{\theta}) = \frac{1}{|\mathcal{S}_\pm|} \sum_{\boldsymbol{x}\in\mathcal{S}_\pm} |\phi_{\boldsymbol{\theta}}(\boldsymbol{x})\rangle\langle\phi_{\boldsymbol{\theta}}(\boldsymbol{x})| \tag{A1}$$

$$= \frac{1}{|\mathcal{S}_\pm|} \sum_{\boldsymbol{x}\in\mathcal{S}_\pm} \phi_{\boldsymbol{\theta}}(\boldsymbol{x}). \tag{A2}$$

Here, we denoted the density matrix of the embedding as $\phi_{\boldsymbol{\theta}}(\boldsymbol{x}) = |\phi_{\boldsymbol{\theta}}(\boldsymbol{x})\rangle\langle\phi_{\boldsymbol{\theta}}(\boldsymbol{x})|$. The state $\rho_\pm$ models an approach

where the the encoded datapoint $\boldsymbol{x}$ is uniformly sampled from $\mathcal{S}_\pm$.

Reference [39] suggests to optimize the embedding $|\phi_{\boldsymbol{\theta}}(\boldsymbol{x})\rangle$ by maximizing the Hilbert-Schmidt distance of the class states, i.e.,

$$P(\boldsymbol{\theta}) = \mathrm{Tr}\{[\rho_+(\boldsymbol{\theta}) - \rho_-(\boldsymbol{\theta})]^2\}. \tag{A3}$$

Its relation to kernel-target alignment becomes apparent if we rewrite the numerator of the kernel-target alignment—the polarity—in terms of these density matrices. We therefore consider the polarity for imbalanced datasets, where we rescale the labels with the number of datapoints in the class. The rescaled labels are denoted as $\hat{y}_j$:

$$\sum_{i,j=1}^{N} \hat{y}_i \hat{y}_j k_{\boldsymbol{\theta}}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sum_{i,j=1}^{N} \hat{y}_i \hat{y}_j \langle \phi_{\boldsymbol{\theta}}(\boldsymbol{x}_i), \phi_{\boldsymbol{\theta}}(\boldsymbol{x}_j)\rangle \tag{A4}$$

$$= \left\langle \sum_{i=1} \hat{y}_i \phi_{\boldsymbol{\theta}}(\boldsymbol{x}_i), \sum_{i=1} \hat{y}_i \phi_{\boldsymbol{\theta}}(\boldsymbol{x}_i) \right\rangle \tag{A5}$$

$$= \left\| \sum_{i=1} \hat{y}_i \phi_{\boldsymbol{\theta}}(\boldsymbol{x}_i) \right\|^2. \tag{A6}$$

The polarity is therefore nothing else but the squared norm of $\sum_{i=1} \hat{y}_i \phi_{\boldsymbol{\theta}}(\boldsymbol{x}_i)$, which is a weighted sum of the embedded datapoints. For QEKs, this is equal to the difference of the two class matrices introduced above:

$$\sum_{i=1} \hat{y}_i \phi_{\boldsymbol{\theta}}(\boldsymbol{x}_i) = \sum_{\boldsymbol{x}_+ \in \mathcal{S}_+} \frac{\phi_{\boldsymbol{\theta}}(\boldsymbol{x}_+)}{|\mathcal{S}_+|} - \sum_{\boldsymbol{x}_- \in \mathcal{S}_-} \frac{\phi_{\boldsymbol{\theta}}(\boldsymbol{x}_-)}{|\mathcal{S}_-|} \tag{A7}$$

$$= \rho_+(\boldsymbol{\theta}) - \rho_-(\boldsymbol{\theta}). \tag{A8}$$

This means that the polarity is equal to the Hilbert-Schmidt distance introduced in Ref. [39], as found in Eq. (A3)

As already noted in Ref. [39], the polarity can be rewritten as

$$P(\boldsymbol{\theta}) = \mathrm{Tr}\{\rho_+(\boldsymbol{\theta})^2 + \rho_-(\boldsymbol{\theta})^2 - 2\rho_+(\boldsymbol{\theta})\rho_-(\boldsymbol{\theta})\}. \tag{A9}$$

Consequently, increasing the polarity translates to an increase in the *purity* of the class states $\mathrm{Tr}\{\rho_\pm(\boldsymbol{\theta})\}^2$, thereby encouraging points in the dataset to cluster closer together in feature space. At the same time, this cost function decreases the overlap of the two data embedding states, thereby encouraging them to reside in different corners of the Hilbert space.

However, we are of the opinion that the kernel-target alignment—representing the normalized polarity—is a measure that is easier to interpret and more accessible to numerical optimization than the pure polarity. Reference [39] proposes a classifier where the overlap of the embedded datapoint with the two class states is computed. The label of the class state with the larger overlap is then assigned to the new datapoint. This corresponds to a kernelized nearest-centroid classification. We conclude that the use of the embedding in a support vector machine allows for more sophisticated decision boundaries than the method proposed in Ref. [39].

APPENDIX B: DATASET DETAILS

In this Appendix, we will discuss the details of the datasets used.

1. Donut dataset

The symmetric donuts dataset is an artificial dataset that has 60 training and 60 test datapoints. The datapoints are generated by sampling points uniformly at random from a circle of radius $\sqrt{2}/2$ and then labeling them according to whether they fall within an inner circle of radius $1/2$ or without. We do this one time centering the circles on the x axis, on the point (1,0), giving the inner points label 1 and the outer ones label -1. Next, we repeat the process for circles centered about the point $(-1, 0)$ and this time exchange the labels: the inner point class is now -1 and the outer $+1$. This way we obtain a dataset contained in the domain $[-(3+\sqrt{2})/2, (3+\sqrt{2})/2] \times [-\sqrt{2}/2, \sqrt{2}/2]$.

2. Checkerboard dataset

The checkerboard dataset contains 30 train and and 30 test datapoints, and represents a 4×4 grid of alternating classes, where the elements of the checkerboard are drawn from a continuous uniform distribution centered in the tiles of the checkerboard. In particular, we defined a 4×4 grid in the domain $[0, 1]^2$ with sites i, j at coordinates $[(2i+1)/8, (2j+1)/8]$ to prevent overlap between centroids and spilling out of the fixed domain. Next, we sampled points uniformly centered about each grid site. At the end, we assigned alternating classes to each of the sites, and finished by assigning all swarms of points the class corresponding to their centroid.

APPENDIX C: DETAILS ON POST-PROCESSING METHODS

1. Runtimes and output properties

The post-processing methods we introduced in Secs. IV D and IV B differ in their classical and quantum computational cost and in the properties of the output matrix.

The regularization methods R-TIK and R-THR require the computation of the smallest eigenvalue and of the full eigenvalue decomposition respectively, which has classical complexity $O(n^3)$ with naive methods but more realistically scales like matrix multiplication for relevant sizes with $O(n^{2.8})$ (Strassen algorithm [64]).[5] The worst case scaling for R-SDP is $O(n^{3.8})$, again assuming the Strassen algorithm for matrix multiplication and considering that we use n constraints to fix the diagonal entries [64,65]. In our experiments on datasets with 60 datapoints, the former two methods had negligible computational cost, whereas R-SDP took 0.5 s on average for this rather small matrix. In addition to this large difference for the used matrix size, some additional tests for larger random matrices confirmed a significantly worse scaling of the cost for R-SDP compared to R-TIK and R-THR.

As they only act on the spectrum of the kernel matrix, R-TIK and R-THR preserve its eigenbasis, a potentially relevant property for the classification task. On the contrary, R-SDP does not preserve the eigenbasis but ensures that the output kernel matrix has the correct diagonal entries.

[5] If this was to be a bottle neck, then the full matrix multiplication may be skipped when multiplying the kernel matrix with vectors only.

For the proposed mitigation methods, additional quantum computation is required to determine the diagonal entries, which in turn are used to estimate the depolarizing survival probabilities. The number of required entries is 1, $n_{\text{mean}} \in [1, n]$, and n for M-SINGLE, M-MEAN, and M-SPLIT, respectively, which then should be measured as often as the other matrix entries. While estimating the probabilities has negligible cost, the modification of the matrix requires $O(n^2)$ classical computation resources.[6]

Considering Eq. (38), we see that our mitigation methods estimate the survival probability λ_i to be larger than 1 for $K_{ii}^{(\text{dev})} > 1$ and to be imaginary if $K_{ii}^{(\text{dev})} < 2^{-N}$, both being unreasonable estimates. The first will only ever occur if a previous post-processing method increased the diagonal element $K_{ii}^{(\text{dev})}$ too far, as a QPU itself will not output measurement probabilities above 1. The second may occur in the presence of very strong noise that suppresses the exact value of 1 to 2^{-N}, which would presumably imply the QPU output to be impracticably flawed anyways. For M-SINGLE (M-MEAN), the same reasoning holds for the single measured entry (for the average of the considered diagonal entries), i.e., in particular for M-MEAN we are unlikely to run into either of the above problems.

Even if the estimated survival probabilities λ_i lie in the physically meaningful range [0,1], the mitigation might still produce kernel matrix entries that are not valid probabilities and thus can impossibly be the result of a real QEK evaluation. For a given noisy matrix entry $K_{ij}^{(\text{dev})}$, this happens if

$$K_{ij}^{(\text{dev})} \notin [\epsilon, \lambda_i \lambda_j + \epsilon], \tag{C1}$$

where we abbreviated $\epsilon = 2^{-N}(1 - \lambda_i \lambda_j)$ and the estimated probabilities fulfill $\lambda_i = \lambda_j$ for M-SINGLE and M-MEAN. Note that $\lambda_i \approx 1$ and $\epsilon \ll 1$ for reasonable survival rates.

In conclusion, even though there are extreme cases in which our methods might transform the noisy matrix into an invalid kernel matrix, we do not expect these problems to play any role because such extreme noise levels likely would render the QPU output useless.

Note that the error bound in operator distance derived in Sec. IV C is valid for the deviation of the statistical estimator from the noiseless kernel matrix K or from the device-noisy kernel matrix $K^{(\text{dev})}$. When applying post-processing methods however, this bound may not transfer to the output $K^{(\text{post})}$ in general. Consequently, while being designed to counter device and finite sampling noise, the analytic error bound might become worse.

For R-THR, however, this bound is provably maintained [66]: Splitting the indefinite matrix $\bar{K}_M$ into the difference of two positive semidefinite matrices K_+ and K_- with disjoint support, identifying R-THR$(\bar{K}_M) = K_+$ and calculating the distance between the approximand[7] and $\bar{K}_M$ yields

$$\begin{aligned}
\bar{K}_M &=: K_+ - K_-, && \text{(C2)}\\
\|K - \bar{K}_M\|_\infty &= \|K - K_+ + K_-\|_\infty && \text{(C3)}\\
&= \max_{\|x\|_2=1} [x^T (K - K_+)^2 x \\
&\quad + \underbrace{x^T (K K_- + K_- K + K_-^2) x}_{\geqslant 0}] && \text{(C4)}\\
&\geqslant \max_{\|x\|_2=1} x^T (K - K_+)^2 x && \text{(C5)}\\
&= \|K - K_+\|_\infty && \text{(C6)}\\
&= \|K - \text{R-THR}(\bar{K}_M)\|_\infty, && \text{(C7)}
\end{aligned}$$

where we used the positive semidefiniteness of K_- and that $K_\pm K_\mp = 0$ due to the disjoint support.

2. Comparison of post-processing strategies

There are many combinations of the discussed post-processing techniques to choose from to counter both device noise and finite sampling noise. We will consider the following subset of combinations:

First, we apply a regularization R_1, second, we perform device noise mitigation M and third, we regularize again with R_2. For each step we include the option to not modify $K^{(\text{dev})}$ at all (Id).

For the two regularization steps $\text{R}_{1,2}$, we may apply Tikhonov regularization (R-TIK), thresholding (R-THR) or the *semidefinite program* (SDP) that fixes the diagonal while thresholding (R-SDP), see Sec. IV D. For the mitigation step, we choose from estimating a single survival probability based on a single (M-SINGLE) or the mean (M-MEAN) diagonal entry of $\bar{K}_M$, or estimating survival probabilities per feature embedding (M-SPLIT), see Sec. IV B.

Naively, this yields $4^3 = 64$ combinations when including the trivial transformation Id, out of which some are identical, e.g., Id, Id, R and R, Id, Id. In addition, there are special combinations in which methods effectively act like Id: First, combinations of the form SDP, M, R_2 for which M already receives a positive semidefinite input matrix with correct diagonal entries and thus will estimate the survival probability to be 1. Second, some combinations without mitigation (namely, TIK/THR, Id, TIK/THR) in which R_2 would be redundant. Here we already excluded the combinations obeying the first pattern. Excluding duplicates and these "reducible" combinations, we obtain 42 reasonable, distinct strategies (including Id, Id, Id) and for each of the outcomes $K^{(\text{post})}$ we compute the kernel alignment [see Eq. (27)] with the noiseless matrix K.

In Fig. 6 in the main text we showed the best of the 42 combinations for each pair of noise parameters M and λ_0. However, to obtain a clear analysis, we reduced the set of combinations to the ones that performed best in a wide range of noise parameters. This manual filtering step changed the optimal post-processing method for 22 configurations, but

[6]This may again be improved if we are not interested in the fully computed matrix but, e.g., in multiplying it with vectors, should it ever become a relevant resource requirement.

[7]We here show the calculation when approximating K. It has to be replaced by $K^{(\text{dev})}$ accordingly when approximating the device-noisy matrix by sampling.

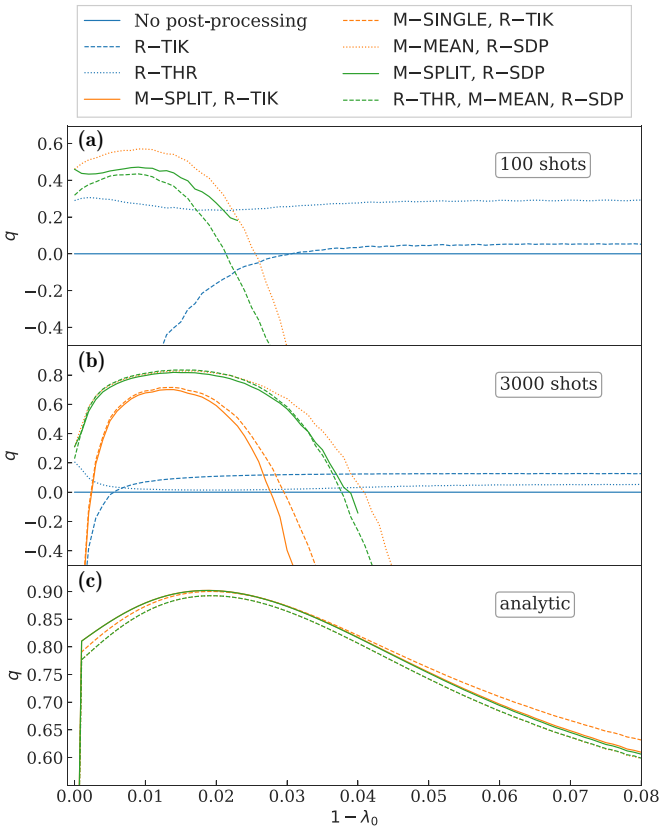

FIG. 9. Comparison of the relative improvement q for all regularization methods that are best for at least ten configurations of shots M and base survival probabilities λ_0, shown for three different shot numbers and averaged over 100 kernel matrix samples. Note the differing q axes and that missing post-processing methods either performed poorly or did not converge at all due to large noise. We include the strategy M-SPLIT, R-SDP in addition to allow the comparison to the hardware results. The transition to negative q for a variety of methods is shifted to stronger device noise when increasing M, as also is visible in Fig. 6 in the main text. For $M \to \infty$, the transition is no longer within the range of investigated λ_0.

only reduced the attained performance by 0.76% on average and at most 0.94%. 19 of these changes exchanged the slightly more complex combination R-THR, M-MEAN, R-SDP at $M = 3000$ for the combination M-MEAN, R-SDP.

The behavior of the regularization methods is shown in more detail for $M = 100$, $M = 3000$, and $M \to \infty$ in Fig. 9. We observe that while M-MEAN, R-SDP, which was identified as best approach for low device noise, outperforms all other methods for $M = 100$, there are several equally performing methods for $M = 3000$. This justifies the reduction of the best-performing methods discussed above. For the analytic case $M \to \infty$, we observe different methods to perform best, as seen in the first row of Fig. 6 in the main text. However, even in this case, the combination M-MEAN, R-SDP performs almost as well as these methods, making it a strong candidate for a wide range of noise scenarios.

To allow for comparison to the hardware results in Sec. IV F, we included the strategy M-SPLIT, R-SDP in Fig. 9. We observe that it does not perform as well as M-MEAN, R-SDP and does not even yield valid kernel matrices for large device noise, with both effects being more severe for small shot numbers. This is in contrast to the hardware results, for which this method performs very well for shot numbers up to 175.

APPENDIX D: SIMULATING DEVICE NOISE BY DEPOLARIZATION

For the simulation of device noise in Sec. IV E we use the following noise model: After each unitary gate we apply single-qubit depolarizing noise channels $\mathcal{D}_\lambda$ to each qubit the gate acted on [see Eq. (30) with $N = 1$].

Recalling the discussion in Sec. IV A, we remark that the qubitwise depolarizing channel does commute with single-qubit but not with multiqubit gates like the ring of controlled gates in our embedding circuit. In this sense our model properly captures the case in which the device noise invalidates the adjoint approach, potentially destroying the positive semidefiniteness of the kernel matrix, and our post-processing strategies are challenged to correct this deviation.

The *base survival probability* λ_0 quantifies the overall noise strength. However, it is reasonable to expect that the noise strength for a specific gate on a QPU depends on the duration of the pulses that implement the gate, leading to different effective noise levels for different embedded datapoints. To capture this dependence, we rescale the base survival probability λ_0 for a rotation gate about the angle θ according to

$$\lambda = \left(1 - \frac{\theta}{2\pi}\right) + \lambda_0 \frac{\theta}{2\pi} \tag{D1}$$

and fix the survival probability of the Hadamard and idling gate to $(1 + \lambda_0)/2$ and $(1 + 3\lambda_0)/4$, respectively.

We do not simulate any device readout error explicitly but assume the presented implementation of depolarizing noise to represent the full device noise closely enough. This assumption seems to be valid considering our results in Secs. IV E and IV F and the accordance between them.

APPENDIX E: MEASUREMENT PRECISION OF KERNEL TARGET ALIGNMENT

In this section we calculate the scaling of the number of measurements required to obtain the kernel target alignment to a given precision ϵ, complementing the analysis in Sec. IV C.

First, compute the partial derivative

$$\frac{\partial \mathrm{TA}(K)}{\partial K_{kl}} = \frac{y_k y_l}{n\|K\|_F} - \frac{\langle K, K^* \rangle_F K_{kl}}{n\|K\|_F^3}, \tag{E1}$$

which squares to

$$\left(\frac{\partial \mathrm{TA}(K)}{\partial K_{kl}}\right)^2 = \frac{1}{n^2\|K\|_F^2} + \frac{\mathrm{TA}(K)^2}{\|K\|_F^4} K_{kl}^2 \tag{E2}$$

$$-2\frac{\mathrm{TA}(K)}{n\|K\|_F^3} y_k y_l K_{kl}. \tag{E3}$$

Note that the norm of K is bounded from below via

$$\|K\|_F^2 = \sum_{ij} K_{ij}^2 = \sum_i K_{ii}^2 + \sum_{i \neq j} K_{ij}^2 \geqslant n. \tag{E4}$$

We furthermore know that $y_k \in \{1, -1\}$, $K_{kl} \in [0, 1]$ as well as $\mathrm{TA}(K) \in [0, 1]$, and hence we may bound the above

expression by

$$\left[\frac{\partial \mathrm{TA}(K)}{\partial K_{kl}}\right]^2 \leqslant \frac{1}{n^3} + \frac{1}{n^2} + \frac{2}{n^{5/2}}. \tag{E5}$$

The statistical variance of $\mathrm{TA}(K)$ hence is

$$\sigma_{\mathrm{TA}}^2 = \sum_{kl}\left[\frac{\partial \mathrm{TA}(K)}{\partial K_{kl}}\right]^2 \sigma_{kl}^2 \tag{E6}$$

$$\leqslant \left(\frac{1}{n^2} + \frac{2}{n^{5/2}} + \frac{1}{n^3}\right)\sum_{kl}\sigma_{kl}^2 \tag{E7}$$

$$= O\left(\frac{1}{M}\right), \tag{E8}$$

where we used that the statistical variance of each kernel entry is $O(1/M)$. Note that we only kept the leading order contribution in n, in accordance with the asymptotic analysis in Sec. IV C. In conclusion, the required number of shots per kernel entry scales as $O(1/\epsilon^2)$, the total number of shots as $O(n^2/\epsilon^2)$.

TABLE I. Overview of the kernel hours required by our numerical simulations and their corresponding estimated CO_2 emissions.

Numerical simulations	
Total kernel hours [h]	7250
Thermal design power per kernel [W]	4.6
Total energy consumption simulations [kWh]	32.8
Average emission of CO_2 in Germany/USA [kg/kWh]	0.47
Total CO_2-emission for numerical simulations [kg]	15.5
Estimated CO_2-emission for QPU usage [kg]	21.4
Were the emissions offset?	**Yes**
Total CO_2-emission [kg]	36.9

APPENDIX F: CO_2 EMISSION TABLE

We endorse the scientific CO_2nduct initiative [63] and have listed all carbon costs resulting from this work in Table I.

[1] O. Vinyals *et al.*, Grandmaster level in StarCraft II using multi-agent reinforcement learning, Nature **575**, 350 (2019).

[2] T. B. Brown *et al.*, Language models are few-shot learners, arXiv:2005.14165.

[3] A. W. Senior *et al.*, Improved protein structure prediction using potentials from deep learning, Nature **577**, 706 (2020).

[4] F. Arute *et al.*, Quantum supremacy using a programmable superconducting processor, Nature **574**, 505 (2019).

[5] M. Cerezo, A. Arrasmith, R. Babbush, S. C. Benjamin, S. Endo, K. Fujii, J. R. McClean, K. Mitarai, X. Yuan, L. Cincio, and P. J. Coles, Variational quantum algorithms, Nat. Rev. Phys. **3**, 625 (2021).

[6] K. Bharti, A. Cervera-Lierta, T. H. Kyaw, T. Haug, S. Alperin-Lea, A. Anand, M. Degroote, H. Heimonen, J. S. Kottmann, T. Menke, W.-K. Mok, S. Sim, L.-C. Kwek, and A. Aspuru-Guzik, Noisy intermediate-scale quantum (NISQ) algorithms, Rev. Mod. Phys. **94**, 015004 (2022).

[7] M. Schuld and F. Petruccione, *Supervised Learning with Quantum Computers*, Quantum Science and Technology (Springer International Publishing, Cham, 2018).

[8] P. Wittek, *Quantum Machine Learning: What Quantum Computing Means to Data Mining*, Elsevier Insights (Elsevier Science, Amsterdam, 2014).

[9] S. Lloyd, M. Mohseni, and P. Rebentrost, Quantum principal component analysis, Nat. Phys. **10**, 631 (2014).

[10] M. Schuld, R. Sweke, and J. J. Meyer, Effect of data encoding on the expressive power of variational quantum-machine-learning models, Phys. Rev. A **103**, 032430 (2021).

[11] V. Havlíček, A. D. Córcoles, K. Temme, A. W. Harrow, A. Kandala, J. M. Chow, and J. M. Gambetta, Supervised learning with quantum enhanced feature spaces, Nature **567**, 209 (2019).

[12] M. Schuld and N. Killoran, Quantum Machine Learning in Feature Hilbert Spaces, Phys. Rev. Lett. **122**, 040504 (2019).

[13] T. Kusumoto, K. Mitarai, K. Fujii, M. Kitagawa, and M. Negoro, Experimental quantum kernel machine learning with nuclear spins in a solid, npj Quantum Inf. **7**, 94 (2021).

[14] K. Bartkiewicz, C. Gneiting, A. Cernoch, K. Jiráková, K. Lemr, and F. Nori, Experimental kernel-based quantum machine learning in finite feature space, Sci. Rep. **10**, 12356 (2020).

[15] C. Blank, D. K. Park, J.-K. K. Rhee, and F. Petruccione, Quantum classifier with tailored quantum kernel, npj Quantum Inf. **6**, 41 (2020).

[16] H.-Y. Huang, M. Broughton, M. Mohseni, R. Babbush, S. Boixo, H. Neven, and J. R. McClean, Power of data in quantum machine learning, Nat. Commun. **12**, 2631 (2021).

[17] E. Peters, J. Caldeira, A. Ho, S. Leichenauer, M. Mohseni, H. Neven, P. Spentzouris, D. Strain, and G. N. Perdue, Machine learning of high dimensional data on a noisy quantum processor, arXiv:2101.09581.

[18] M. Schuld, Quantum machine learning models are kernel methods, arXiv:2101.11020.

[19] S. Jerbi, L. J. Fiderer, H. P. Nautrup, J. M. Kubler, H. J. Briegel, and V. Dunjko, Quantum machine learning beyond kernel methods, arXiv:2110.13162.

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

[21] B. Schölkopf and A. J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*, Adaptive Computation and Machine Learning (MIT Press, Cambridge, MA, 2002).

[22] B. Schölkopf, A. Smola, and K.-R. Müller, Kernel principal component analysis, in *Proceedings of the International Conference on Artificial Neural Networks (ICANN'97)*, Vol. 1327, edited by W. Gerstner, A. Germond, M. Hasler, and J.-D. Nicoud (Springer, Berlin, 1997), pp. 583–588.

[23] C. Saunders, A. Gammerman, and V. Vovk, Ridge regression learning algorithm in dual variables, in *Proceedings of the 15th International Conference on Machine Learning(ICML'98)* (Morgan Kaufmann, San Francisco, CA, 1998), pp. 515–521.

[24] M. Fanizza, M. Rosati, M. Skotiniotis, J. Calsamiglia, and V. Giovannetti, Beyond the Swap Test: Optimal Estimation of Quantum State Overlap, Phys. Rev. Lett. **124**, 060503 (2020).

[25] H. Buhrman, R. Cleve, J. Watrous, and R. de Wolf, Quantum Fingerprinting, Phys. Rev. Lett. **87**, 167902 (2001).

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

[27] H.-Y. Huang, R. Kueng, and J. Preskill, Predicting many properties of a quantum system from very few measurements, Nat. Phys. **16**, 1050 (2020).

[28] S. T. Flammia and Y.-K. Liu, Direct Fidelity Estimation from Few Pauli Measurements, Phys. Rev. Lett. **106**, 230501 (2011).

[29] T. Wang, D. Zhao, and S. Tian, An overview of kernel alignment and its applications, Artific. Intell. Rev. **43**, 179 (2015).

[30] C. Cortes, M. Mohri, and A. Rostamizadeh, Algorithms for learning kernels based on centered alignment, J. Mach. Learn. Res. **13**, 795 (2012).

[31] Y. Baram, Learning by kernel polarization, Neural Comput. **17**, 1264 (2005).

[32] T. Wang, S. Tian, H. Huang, and D. Deng, Learning by local kernel polarization, Neurocomputing **72**, 3077 (2009).

[33] L. Wang, Feature selection with kernel class separability, IEEE Trans. Pattern Anal. Mach. Intell. **30**, 1534 (2008).

[34] Huilin Xiong, M. N. S. Swamy, and M. O. Ahmad, Optimizing the kernel in the empirical feature space, IEEE Trans. Neural Netw. **16**, 460 (2005).

[35] C. H. Nguyen and T. B. Ho, An efficient kernel matrix evaluation measure, Pattern Recogn. **41**, 3366 (2008).

[36] N. Cristianini, J. Kandola, A. Elisseeff, and J. Shawe-Taylor, On Kernel Target Alignment, in *Innovations in Machine Learning: Theory and Applications*, Studies in Fuzziness and Soft Computing, edited by D. E. Holmes and L. C. Jain (Springer, Berlin, 2006).

[37] J. Kandola, J. Shawe-Taylor, and N. Cristianini, Optimizing Kernel Alignment over Combinations of Kernels, Tech. Rep. 121, Electronics & Computer Science (2002), https://eprints.soton.ac.uk/259746/.

[38] J. R. McClean, J. Romero, R. Babbush, and A. Aspuru-Guzik, The theory of variational hybrid quantum-classical algorithms, New J. Phys. **18**, 023023 (2016).

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

[40] S. Moro, P. Cortez, and P. Rita, A data-driven approach to predict the success of bank telemarketing, Decis. Supp. Syst. **62**, 22 (2014).

[41] A. Pérez-Salinas, A. Cervera-Lierta, E. Gil-Fuster, and J. I. Latorre, Data re-uploading for a universal quantum classifier, Quantum **4**, 226 (2020).

[42] Github code repository, https://github.com/thubregtsen/qhack (2021).

[43] K. Temme, S. Bravyi, and J. M. Gambetta, Error Mitigation for Short-Depth Quantum Circuits, Phys. Rev. Lett. **119**, 180509 (2017).

[44] S. Endo, S. C. Benjamin, and Y. Li, Practical Quantum Error Mitigation for Near-Future Applications, Phys. Rev. X **8**, 031027 (2018).

[45] A. Lowe, M. H. Gordon, P. Czarnik, A. Arrasmith, P. J. Coles, and L. Cincio, Unified approach to data-driven quantum error mitigation, Phys. Rev. Res. **3**, 033098 (2021).

[46] T. Giurgica-Tiron, Y. Hindy, R. LaRose, A. Mari, and W. J. Zeng, Digital zero noise extrapolation for quantum error mitigation, in *Proceedings of the IEEE International Conference on Quantum Computing and Engineering (QCE'20)* (IEEE, Piscataway, NJ, 2020), pp. 306–316.

[47] A. Strikis, D. Qin, Y. Chen, S. C. Benjamin, and Y. Li, Learning-based quantum error mitigation, PRX Quantum **2**, 040330 (2021).

[48] B. Koczor, Exponential Error Suppression for Near-Term Quantum Devices, Phys. Rev. X **11**, 031057 (2021).

[49] W. J. Huggins, S. McArdle, T. E. O'Brien, J. Lee, N. C. Rubin, S. Boixo, K. B. Whaley, R. Babbush, and J. R. McClean, Virtual Distillation for Quantum Error Mitigation, Phys. Rev. X **11**, 041036 (2021).

[50] R. Latala, Some estimates of norms of random matrices, Proc. Am. Math. Soc. **133**, 1273 (2005).

[51] R. Vershynin, Introduction to the nonasymptotic analysis of random matrices, arXiv:1011.3027.

[52] Z. D. Bai and Y. Q. Yin, Limit of the smallest eigenvalue of a large dimensional sample covariance matrix, Ann. Probab. **21**, 1275 (1993).

[53] V. Roth, J. Laub, M. Kawanabe, and J. Buhmann, Optimal cluster preserving embedding of nonmetric proximity data, IEEE Trans. Pattern Anal. Mach. Intell. **25**, 1540 (2004).

[54] A. N. Tikhonov, On the stability of inverse problems, Proc. USSR Acad. Sci. **39**, 195 (1943).

[55] T. Graepel, R. Herbrich, P. Bollmann-Sdorra, and K. Obermayer, Classification on pairwise proximity data, in *Proceedings of the 1998 Conference on Advances in Neural Information Processing Systems II* (MIT Press, Cambridge, MA, 1999), pp. 438–444.

[56] A. Kandala, K. Temme, A. D. Corcoles, A. Mezzacapo, J. M. Chow, and J. M. Gambetta, Extending the computational reach of a noisy superconducting quantum processor, Nature **567**, 491 (2019).

[57] C. Schwemmer, L. Knips, D. Richart, H. Weinfurter, T. Moroder, M. Kleinmann, and O. Gühne, Systematic Errors in Current Quantum State Tomography Tools, Phys. Rev. Lett. **114**, 080403 (2015).

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

[59] M. M. Bronstein, J. Bruna, T. Cohen, and P. Veličković, Geometric deep learning: Grids, groups, graphs, geodesics, and gauges, arXiv:2104.13478.

[60] J. R. McClean, S. Boixo, V. N. Smelyanskiy, R. Babbush, and H. Neven, Barren plateaus in quantum neural network training landscapes, Nat. Commun. **9**, 4812 (2018).

[61] M. Cerezo, A. Sone, T. Volkoff, L. Cincio, and P. J. Coles, Cost function dependent barren plateaus in shallow parametrized quantum circuits, Nat. Commun. **12**, 1791 (2021).

[62] A. Uvarov and J. Biamonte, On barren plateaus and cost function locality in variational quantum algorithms, J. Phys. A: Math. Theor. **54**, 245301 (2021).

[63] R. Sweke, P. Boes, N. Ng, C. Sparaciari, J. Eisert, and M. Goihl, Transparent reporting of research-related greenhouse gas

emissions through the scientific CO_2nduct initiative, Commun. Phys. **5**, 150 (2022).

[64] V. Strassen, Gaussian elimination is not optimal, Numer. Math. **13**, 354 (1969).

[65] Y. T. Lee, A. Sidford, and S. C.-w. Wong, A faster cutting plane method and its implications for combinatorial and convex optimization, in *Proceedings of the IEEE 56th Annual Symposium on Foundations of Computer Science* (IEEE, Piscataway, NJ, 2015), pp. 1049–1065.

[66] M. Guţă, J. Kahn, R. Kueng, and J. A. Tropp, Fast state tomography with optimal error bounds, J. Phys. A: Math. Theor. **53**, 204001 (2020).