Quantum kernels through the lens of entangled tensor kernels

Seongwook Shin [1,2,*] Ryan Sweke [3,4,5,†] and Hyunseok Jeong [6,‡]

[1]*Department of Quantum Information Science and Engineering, Sejong University, Seoul 05006, Republic of Korea*
[2]*Dahlem Center for Complex Quantum Systems, Freie Universität Berlin, 14195 Berlin, Germany*
[3]*African Institute for Mathematical Sciences, Cape Town 7945, South Africa*
[4]*Department of Mathematical Sciences, Stellenbosch University, Stellenbosch 7600, South Africa*
[5]*National Institute for Theoretical and Computational Sciences (NITheCS), Stellenbosch 7602, South Africa*
[6]*NextQuantum Innovation Research Center, Department of Physics and Astronomy, Seoul National University, Seoul 08826, Republic of Korea*

(Received 29 April 2025; accepted 25 April 2026; published 19 May 2026)

Quantum kernel methods are one of the most explored approaches to quantum machine learning. However, the structural properties and inductive bias of quantum kernels are not fully understood. In this work, we introduce the notion of entangled tensor kernels—a generalization of product kernels from classical kernel theory—and show that all embedding quantum kernels can be understood as an entangled tensor kernel. We discuss how this perspective allows one to gain insights into both the unique inductive bias of quantum kernels and potential methods for their dequantization.

DOI: 10.1103/c53t-rybw

I. INTRODUCTION

It is currently an open question to understand the extent to which quantum computers can offer meaningful advantages for practically relevant classical machine learning problems. To this end, a significant amount of effort has been invested in better understanding both the potential and limitations of a wide variety of approaches to quantum machine learning [1–4]. One such approach is that of *quantum kernel methods*, which are simply standard kernel-based learning algorithms [5,6], in which the classical kernel is replaced with a quantum kernel [7].

The first proposals for quantum kernels [8,9] were for what is now referred to as either *embedding quantum kernels* [10,11] or *fidelity quantum kernels* [12]. For these quantum kernels, the high-level idea is as follows: First, use some data-dependent quantum circuit to map the classical data to quantum feature states (i.e., "embed" the classical data into Hilbert space). Then, use a quantum device to measure the fidelity between the quantum feature states. In light of these proposals, the natural question is the following: Given a specific problem, how should one choose an appropriate quantum kernel? Ideally, one would like a kernel that is efficient to evaluate with a quantum device, hard to evaluate with a classical device, and guaranteed to generalize well for the problem of interest after training on a polynomially sized dataset (i.e., has a suitable *inductive bias*).

Toward this end, one can show that for specially designed artificial problems, it is possible to construct embedding quantum kernels, which yield a rigorous advantage over classical learning algorithms, under cryptographic assumptions [13]. However, these results give little insight into how one should choose a quantum kernel for more practically relevant problems, and we are now aware of a variety of potential obstacles in this regard. In particular, one can in fact show that, without taking care, the exponential size of Hilbert space can be a distinct *disadvantage* for quantum kernels. More specifically, it can lead to both poor generalization [14,15] and the requirement of exponentially many measurements for evaluating the kernel [16].

Given these obstacles, a variety of approaches have been suggested. One such approach is to introduce a *bandwidth* parameter, which can be shown to facilitate generalization of embedding quantum kernels if chosen appropriately [17,18]. Another approach is to use *trainable* embedding quantum kernels [10,19], which are optimized to match the problem at hand. Finally, *projected quantum kernels* have been suggested [15], in which local quantum feature (reduced) states are used, as opposed to the global fidelity.

With the goal of further enriching our evolving understanding of quantum kernels, in this work we introduce the notion of an *entangled tensor kernel* (ETK)—a generalization of classical product kernels—and show that all embedding quantum kernels are entangled tensor kernels. We then show how this characterization of quantum kernels can be used as lens for their analysis, and, in particular, to gain a variety of insights into both the potential advantages of quantum kernels and methods for their dequantization. We provide a more detailed sketch of our contributions in Sec. I A below. While this work certainly does not solve the problem of quantum kernel

*Contact author: seongwook_shin@sejong.ac.kr
†Contact author: rsweke@aims.ac.za
‡Contact author: h.jeong37@gmail.com

Published by the American Physical Society under the terms of the Creative Commons Attribution 4.0 International license. Further distribution of this work must maintain attribution to the author(s) and the published article's title, journal citation, and DOI.

selection or provide any definitive answers to the question of whether or not one can obtain meaningful advantages with quantum kernels, we hope that the framework of entangled tensor kernels can be a useful tool for the further construction and analysis of both classical and quantum kernels.

A. Contributions

Entangled tensor kernels. Our first contribution is to define the notion of an ETK, a natural generalization of classical product kernels. More specifically, any entangled tensor kernel is specified by *local* feature maps (typically into low-dimensional feature spaces) and a core tensor that "entangles" the local features. We analyze the complexity of evaluating ETKs and provide sufficient conditions for their efficient classical evaluation. Specifically, whenever the local feature maps can be efficiently evaluated and the core tensor has a (known) representation as a polynomial bond-dimension matrix product operator (MPO), the ETK can be evaluated efficiently classically. While our primary motivation for introducing ETKs is to provide a tool for the analysis of quantum kernels, they are potentially interesting in their own right, as a tensor network-based approach to constructing classical kernels. As such, ETKs complement the growing body of work aimed at both developing tensor network-based machine learning methods and using tensor networks to gain insight into existing machine learning approaches [20–24].

Quantum kernels are entangled tensor kernels. Having defined the notion of an entangled tensor kernel, the main result of this work is the observation that all *embedding* quantum kernels are entangled tensor kernels, whose local feature maps depend only on the data-encoding strategy and whose core tensor depends only on the data-independent quantum gates, in a well-defined way. As such, we show both that entangled tensor kernels provide a lens for the analysis and understanding of quantum kernels and that quantum kernels provide a means for the implementation of ETKs.

Inductive bias of quantum kernels through the ETK lens. Two necessary conditions for achieving a meaningful quantum advantage with a quantum kernel are that it should not be efficiently classically simulable, and when using the quantum kernel one should be able to obtain good generalization for the problem of interest from polynomially many samples—i.e., the quantum kernel should have the right *inductive bias*. Equipped with the characterization of quantum kernels as ETKs and our understanding of the complexity of evaluating ETKs, we can immediately see that a necessary (but not sufficient) condition for the hardness of simulating a quantum kernel is that the core tensor of its ETK representation is of superpolynomial bond dimension. As such, we can identify efficient quantum kernels that generate ETKs whose core tensor is of superpolynomial bond dimension, as a natural class of quantum kernels for further study. Additionally, the inductive bias of a kernel is determined by its *Mercer decomposition*, and we show how the ETK representation of quantum kernels can facilitate the derivation of their Mercer decomposition and subsequent analysis of their inductive bias and generalization performance.

Dequantization through the ETK lens. The fact that all quantum kernels admit an ETK representation naturally suggests ETKs with polynomial bond-dimension MPO core tensors—i.e., those that can be evaluated efficiently classically—as candidate *classical surrogates* for quantum kernels. We develop this idea and discuss both the potential and limitations of ETKs as classical surrogates for the dequantization of quantum kernels.

B. Structure of this paper

We begin in Sec. II by providing the preliminary material necessary to understand the results of this work. Having done this, we proceed in Sec. III to define the notion of an entangled tensor kernel. With this in hand, we then show in Sec. IV that all quantum kernels are entangled tensor kernels. In Sec. V, we then discuss the landscape of entangled tensor kernels. More specifically, we use this to discuss when a quantum kernel may provide a potential advantage and when ETKs may be used as classical surrogates for the dequantization of a quantum kernel. To further showcase the utility of the ETK lens for the analysis of quantum kernels, we then provide in Sec. VI an example of a family of quantum kernels for which we are able to use the ETK representation to obtain a Mercer decomposition. This decomposition then allows one to make concrete statements concerning the inductive bias of these kernels, their generalization capacity, and sample complexity for learning. Finally, we conclude with a summary and a sketch of interesting directions in Sec. VII.

II. PRELIMINARIES

A. Notation

Given a Hilbert space $\mathcal{F}$, we use the notation $\langle\cdot|\cdot\rangle_{\mathcal{F}}$ for the inner product associated with $\mathcal{F}$. We use both bold font and bra-ket notation to represent vectors in $\mathbb{C}^d$, and we denote the standard dot product in $\mathbb{C}^d$ simply via $\langle\cdot|\cdot\rangle$. For any function $F : \mathbb{C}^{d'} \mapsto \mathbb{C}^d$, we represent the k'th component of the vector $|F(\mathbf{x})\rangle \in \mathbb{C}^d$ as $F_k(\mathbf{x})$. In other words, $|F(\mathbf{x})\rangle = [F_1(\mathbf{x})\ F_2(\mathbf{x})\ \cdots\ F_d(\mathbf{x})]^\top$. Additionally, we use the notation $L^2(\mathcal{X}, \mu)$ to denote the set of square integrable functions over $\mathcal{X}$ with respect to measure μ. We use $\langle\cdot,\cdot\rangle$ to denote the associated inner product on $L^2(\mathcal{X}, \mu)$—i.e., for any $f, g \in L^2(\mathcal{X}, \mu)$, we have $\langle f, g\rangle = \int \bar{f}(\mathbf{x})g(\mathbf{x})\mathrm{d}\mu(\mathbf{x})$.

Additionally, we note that we will make extensive use of tensor network notation. Given the wealth of excellent resources on tensor networks, we do not provide an introduction to them here, but refer the reader to Refs. [25,26].

B. Supervised learning

In this work, we are concerned with algorithms for *supervised learning*. To formalize such algorithms, we start by letting $\mathcal{X}$ denote the set of all possible data points and $\mathcal{Y}$ a set of all possible labels. We assume the existence of a probability distribution P over $\mathcal{X} \times \mathcal{Y}$, which defines the learning problem and from which data are drawn. For this work, we will always take $\mathcal{X} = \mathbb{R}^d$ and use d to represent the "size" of the learning problem. When $\mathcal{Y}$ is a discrete set (for example, $\{0, 1\}$), we call the associated learning problem a classification problem, and when $\mathcal{Y}$ is continuous (for example, $\mathbb{R}$) we refer to the problem as a regression problem.

We will denote the marginal distribution of P on $\mathcal{X}$ with $P_{\mathcal{X}}$. Additionally, we associate with the learning algorithm a set of functions $\mathcal{H} \subseteq \mathcal{Y}^{\mathcal{X}}$, which we call the hypothesis class. Given access to some finite dataset $S = \{(\mathbf{x}_i, y_i) \sim P\}|_{i=1}^{n}$, the goal of the learning algorithm for the supervised learning problem specified by P is to identify the optimal hypothesis $f^* \in \mathcal{H}$, i.e., the hypothesis that minimizes the *true risk*, defined via

$$R(f) := \mathop{\mathbb{E}}_{(\mathbf{x},y)\sim P}[\mathcal{L}(y, f(\mathbf{x}))], \tag{1}$$

where $\mathcal{L} : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ is some loss function. We note that the true risk cannot be calculated exactly from any finite set of samples without knowledge of P and so we also define the *empirical risk* with respect to the dataset S as

$$\hat{R}_S(f) := \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}(y_i, f(\mathbf{x}_i)). \tag{2}$$

As the empirical risk can be calculated from available data, most supervised learning algorithms proceed by minimizing the empirical risk plus a regularization term, which is added to prevent over fitting to the training data S. Moreover, given a learning algorithm with an associated hypothesis class $\mathcal{H}$, one would ideally like to obtain a *generalization bound*, which allows one to put an upper bound on the difference between the true risk and the empirical risk $|\hat{R}_S(f) - R(f)|$ in terms of properties of the hypothesis class, algorithm, dataset, S and problem $\mathcal{P}$.

C. Kernel methods for supervised learning

Some of the simplest learning algorithms possible are those that use *linear functions* as a hypothesis class. In the case of regression, this is the set of functions

$$\mathcal{H}_{\text{lin}} = \{f_{\mathbf{c}}(\mathbf{x}) = \langle \mathbf{c}|\mathbf{x}\rangle \,|\, \mathbf{c} \in \mathbb{R}^d\}. \tag{3}$$

However, for many problems linear functions are not sufficient to obtain a suitably good solution, and a natural next step is to consider linear functions *in a feature space* $\mathcal{F}$. More specifically, one starts by defining a feature map $F : \mathcal{X} \to \mathcal{F}$, and then considering the class of functions

$$\mathcal{H}_F = \{f_{\mathbf{c},F}(\mathbf{x}) = \langle \mathbf{c}|F(\mathbf{x})\rangle_{\mathcal{F}} \,|\, \mathbf{c} \in \mathcal{F}\}. \tag{4}$$

For any such feature map, one can define an associated *kernel* $K_F : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ via

$$K_F(\mathbf{x}, \mathbf{x}') = \langle F(\mathbf{x})|F(\mathbf{x}')\rangle_{\mathcal{F}}. \tag{5}$$

We note that for real (complex) kernels the feature space $\mathcal{F}$ can be any real (complex) Hilbert space, even infinite dimensional. However, whenever $\mathcal{F}$ is a finite d-dimensional real Hilbert space, there exists an equivalent feature map $\tilde{F} : \mathcal{X} \mapsto \mathbb{R}^d$ satisfying

$$\langle F(\cdot)|F(\cdot)\rangle_{\mathcal{F}} = K_F(\cdot,\cdot) = K_{\tilde{F}}(\cdot,\cdot) = \langle \tilde{F}(\cdot)|\tilde{F}(\cdot)\rangle, \tag{6}$$

where $\langle\cdot|\cdot\rangle$ is the standard dot product in $\mathbb{R}^d$. To construct $\tilde{F}$, let $\{v_i\}$ be an orthonormal basis for $\mathcal{F}$, and define $\varphi : \mathcal{F} \to \mathbb{R}^d$ via $\varphi(\sum_i a_i v_i) = [a_1 \cdots a_d]^\top$. Then, $\tilde{F} = \varphi \circ F$ can be seen to satisfy Eq. (6). We will use this observation in Sec. III to restrict ourselves to considering feature maps to $\mathbb{R}^d$ for finite d.

In many settings, access to the kernel function K_F is sufficient to identify the optimal hypothesis in $\mathcal{H}_F$, with respect to a suitably regularized empirical risk. In this work, we call learning algorithms that only use access to the kernel function *kernel methods*. While we refer to Refs. [5,6] for more detailed information on such algorithms, which include standard methods such as support vector machines and kernel ridge regression (KRR), we mention two important facts here:

(1) For certain feature maps, evaluating the kernel function K_F can be done much more efficiently than by explicitly constructing the vectors $F(\mathbf{x})$ and $F(\mathbf{x}')$ then taking the inner product. Using feature maps, which allow one to exploit this, is called *the kernel trick*.

(2) It is in fact not even necessary to start by specifying a feature map. Indeed, for *any* symmetric positive semidefinite (PSD) function $K : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$, there is some feature map for which K is the kernel. As such, in practice, one typically simply chooses a suitable kernel, as opposed to a feature map.

Inductive bias of kernel methods

The first step in any kernel method is choosing a kernel function, and as such one would ideally like to understand the *inductive bias* associated with a specific kernel—or in other words, the set of problems for which a specific kernel is well suited. To this end, it is useful to analyze the *kernel integral operator* associated with a kernel. To be more specific, for any non-negative measure $P_{\mathcal{X}}$ on $\mathcal{X}$, we define the kernel integral operator $T_{(K,P_{\mathcal{X}})}$ via

$$T_{(K,P_{\mathcal{X}})}(f)(\mathbf{x}) := \int_{\mathcal{X}} K(\mathbf{x}, \mathbf{y}) f(\mathbf{y}) \mathrm{d}P_{\mathcal{X}}(\mathbf{y}). \tag{7}$$

It then follows from *Mercer's theorem* that there exists a (possibly infinite) spectral decomposition of $T_{(K,P_{\mathcal{X}})}$, with eigenvalues $\{\gamma_i\}$ and associated eigenfunctions $\{e_i\}$, which satisfies the following:

(1) All eigenvalues are non-negative—i.e., $\gamma_i \geqslant 0$ for all i.

(2) The eigenfunctions $\{e_i\}$ form an orthonormal basis for $L^2(\mathcal{X}, P_{\mathcal{X}})$.

(3) The kernel K admits a *Mercer decomposition*

$$K(\mathbf{x}, \mathbf{x}') = \sum_{i}^{d} \gamma_i e_i(\mathbf{x}) e_i(\mathbf{x}') \tag{8}$$

$$= \langle e(\mathbf{x})|\Gamma|e(\mathbf{x})\rangle, \tag{9}$$

where d is the number of nonzero eigenvalues, $|e(\mathbf{x})\rangle = |e_1(\mathbf{x}) \cdots e_d(\mathbf{x})\rangle$, and Γ is the diagonal matrix with $\Gamma_{i,i} = \gamma_i$.

For certain kernel methods, such as KRR, the inductive bias of the method when using kernel K is closely related to the spectral decomposition of $T_{(K,P_{\mathcal{X}})}$ (which for convenience we sometimes refer to as the Mercer decomposition of K). More specifically, as $\{e_i\}$ is a basis for $L^2(\mathcal{X}, P_{\mathcal{X}})$, we can decompose any $f \in L^2(\mathcal{X}, P_{\mathcal{X}})$ as $f = \sum_i a_i e_i$. At a high level, then it is biased toward functions that are supported on eigenfunctions e_i corresponding to large eigenvalues γ_i. Said another way, KRR is biased toward functions whose components $\{a_i\}$ are "well aligned" with the spectrum of $T_{(K,P_{\mathcal{X}})}$.

These high-level statements can be made more precise through notions such as *kernel-target alignment* [27] and *task-model alignment* [18,28], both of which provide different

measures of the alignment between the spectrum of the kernel integral operator and the principal components of the target function, when written in the eigenbasis of $T_{(K,P_{\mathcal{X}})}$. The latter, in particular, is useful, as one can prove generalization bounds in terms of the task-model alignment, which in turn allows one to bound the sample complexity required to learn specific functions in a meaningful sense.

D. Quantum kernel methods

As discussed in Sec. I, quantum kernel methods are kernel-based learning algorithms in which the kernel function is evaluated using a quantum computational device [7–9]. In this work, we will consider *embedding quantum kernels* [11], which have the form

$$K_U(\mathbf{x}, \mathbf{x}') := |\langle 0|^{\otimes n} U^\dagger(\mathbf{x}) U(\mathbf{x}')|0\rangle^{\otimes n}|^2, \tag{10}$$

where $U(x)$ is some data-dependent quantum circuit that serves as a feature map, "embedding" classical data into n-qubit quantum states. To see that this is indeed a valid (real) kernel function, first recall that the set of Hermitian matrices $\mathrm{Herm}(2^n)$, equipped with the Hilbert-Schmidt inner product, defined via $\langle A|B\rangle_{\mathrm{HS}} = \mathrm{Tr}[A^\dagger B]$, is a real Hilbert space. Then, note that if we define the feature map $\mathcal{U} : \mathcal{X} \mapsto \mathrm{Herm}(2^n)$ via

$$\mathcal{U}(\mathbf{x}) = U(\mathbf{x})|0\rangle^{\otimes n}\langle 0|^{\otimes n} U^\dagger(\mathbf{x}) \tag{11}$$

$$:= \rho(x), \tag{12}$$

then have that

$$K_U(\mathbf{x}, \mathbf{x}') = \langle \mathcal{U}(\mathbf{x})|\mathcal{U}(\mathbf{x}')\rangle_{\mathrm{HS}} \tag{13}$$

$$= \mathrm{Tr}[\rho(\mathbf{x})\rho(\mathbf{x}')]. \tag{14}$$

Said another way, $K_U(\mathbf{x}, \mathbf{x}')$ is simply the Hilbert-Schmidt inner product of the (pure) *feature states* $\rho(\mathbf{x})$ and $\rho(\mathbf{x}')$, and therefore a valid real kernel. Additionally, we note that Eq. (10) is the *fidelity* of the feature states $\rho(\mathbf{x})$ and $\rho(\mathbf{x}')$, which motivates the name of *fidelity quantum kernels* [12].

Given this, we see that to specify an embedding quantum kernel, one needs to define a data-dependent quantum circuit $U(x)$. Such circuits will in general consist of both data-dependent gates and fixed nonparametrized gates. There also exist recent proposals for *trainable* quantum kernels [10,19], including gates parametrized by trainable parameters, for which everything that follows would hold once these trainable parameters are fixed. Given such a circuit, the kernel function $K_U(\mathbf{x}, \mathbf{x}')$ can then be approximated in multiple ways. One way is via repetition of the quantum circuit illustrated in Fig. 1. Specifically, one first implements the circuit $U(\mathbf{x})$, then the circuit $U^\dagger(\mathbf{x}')$, followed by a measurement in the computational basis. Another method is to prepare both $\rho(\mathbf{x})$ and $\rho(\mathbf{x}')$, then use a swap test to evaluate their inner product. These methods are both efficient whenever $U(\mathbf{x})$ can be implemented efficiently, and we refer to Ref. [29] for a comparison of their relative advantages and disadvantages.

Finally, we stress that the essential difference between quantum and classical kernel methods is the way in which the kernel function is evaluated. As such, all theory pertaining to classical kernel methods—i.e., inductive bias, sample complexity, generalization, etc.—is also applicable to quantum kernel methods. Indeed, this classical theory has already been used to gain significant insight into the inductive bias of quantum kernels [14,15,17,18]. Additionally, we stress that in this work we are uniquely concerned with *embedding* quantum kernel methods as described above, and do not consider alternative quantum kernel methods such as *projected* quantum kernels [15]. For a detailed benchmarking study of both embedding and projected quantum kernels, we refer to Ref. [30].

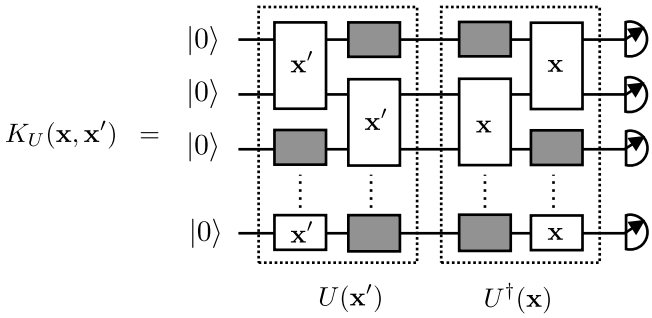


FIG. 1. A circuit diagram for quantum kernel evaluation—i.e., for the evaluation of Eq. (10). White boxes represent data-dependent quantum gates, while gray ones are nonparametrized gates.

III. ENTANGLED TENSOR KERNELS

There are a variety of situations in which one might want to build kernels out of existing kernels. In this work, we consider the following two scenarios:

Extending the data domain: Imagine one has two kernels $K_1 : \mathcal{X}_1 \times \mathcal{X}_1 \mapsto \mathbb{R}$ and $K_2 : \mathcal{X}_2 \times \mathcal{X}_2 \mapsto \mathbb{R}$, for different data domains $\mathcal{X}_1 \neq \mathcal{X}_2$. Sometimes, one would like a kernel that can be used for learning on the product domain $\mathcal{X} = \mathcal{X}_1 \times \mathcal{X}_2$. This would be the case when a dataset is enriched by the addition of distinct features, either through additional data collection or the availability of data (for example, when a version of a census contains new categories).

Enhancing the feature space: Imagine one has two different kernels $K_1 : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ and $K_2 : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$, for the *same* data domain $\mathcal{X}$. Let us denote the feature map associated with K_i via $F^{(i)} : \mathcal{X} \mapsto \mathcal{F}_i$. Sometimes, the feature spaces $\mathcal{F}_i$ may not be "expressive enough" to provide good solutions, and one would like to build a kernel $K : \mathcal{X} \times \mathcal{X}$, using a feature map with the higher-dimensional feature space $\mathcal{F}_1 \otimes \mathcal{F}_2$.

In what follows, we will introduce the notion of *entangled tensor kernels*, which can be used in both of the above situations, in the case when one knows and can evaluate the feature maps of the constituent kernels used as building blocks. In order to gain some intuition, we start with the simple existing method of *product kernels* [31].

A. Warm up: Product kernels

Let us assume that we are in the first situation described above. More specifically, let us assume that we are given two kernels $K_1 : \mathcal{X}_1 \times \mathcal{X}_1 \to \mathbb{R}$ and $K_2 : \mathcal{X}_2 \times \mathcal{X}_2 \to \mathbb{R}$, and we would like to use these to construct a kernel $K : (\mathcal{X}_1 \times \mathcal{X}_2) \times (\mathcal{X}_1 \times \mathcal{X}_2) \to \mathbb{R}$ for the domain $\mathcal{X} = \mathcal{X}_1 \times \mathcal{X}_2$. One way to do this is to define

$$K((\mathbf{x}_1, \mathbf{x}_2), (\mathbf{x}_1', \mathbf{x}_2')) := K_1(\mathbf{x}_1, \mathbf{x}_1') K_2(\mathbf{x}_2, \mathbf{x}_2'). \tag{15}$$

To see that this is a valid kernel and to provide a foundation for the entangled tensor kernels we introduce in Sec. III B, it is instructive to think about this kernel from the feature map perspective. To this end, let $F^{(i)} : \mathcal{X}_i \mapsto \mathcal{F}_i$ be a feature map for K_i. We can then easily see that $F : \mathcal{X}_1 \times \mathcal{X}_2 \to \mathcal{F} = \mathcal{F}_1 \otimes \mathcal{F}_2$ defined as

$$|F(\mathbf{x}_1, \mathbf{x}_2)\rangle = |F^{(1)}(\mathbf{x}_1)\rangle \otimes |F^{(2)}(\mathbf{x}_2)\rangle \tag{16}$$

satisfies

$$\begin{aligned}
&\langle F(\mathbf{x}_1, \mathbf{x}_2)|F(\mathbf{x}_1', \mathbf{x}_2')\rangle_{\mathcal{F}} \\
&\quad = (\langle F^{(1)}(\mathbf{x}_1)| \otimes \langle F^{(2)}(\mathbf{x}_2)|)(|F^{(1)}(\mathbf{x}_1')\rangle \otimes |F^{2}(\mathbf{x}_2')\rangle) && (17)\\
&\quad = \langle F^{(1)}(\mathbf{x}_1)|F^{(1)}(\mathbf{x}_1')\rangle_{\mathcal{F}_1} \langle F^{(2)}(\mathbf{x}_2)|F^{(2)}(\mathbf{x}_2')\rangle_{\mathcal{F}_2} && (18)\\
&\quad = K_1(\mathbf{x}_1, \mathbf{x}_1')K_2(\mathbf{x}_2, \mathbf{x}_2') && (19)\\
&\quad = K((\mathbf{x}_1, \mathbf{x}_2), (\mathbf{x}_1', \mathbf{x}_2')). && (20)
\end{aligned}$$

In particular, we see from Eq. (16) that, for any kernel defined via Eq. (15), the feature vector $|F(\mathbf{x}_1, \mathbf{x}_2)\rangle$ is simply the tensor product of feature vectors $|F^{(1)}(\mathbf{x}_1)\rangle$ and $|F^{(2)}(\mathbf{x}_2)\rangle$. This motivates the name *product kernels* [31] for such kernels. We note that the construction above clearly generalizes to N kernels $\{K_i \,|\, i \in [N]\}$. Additionally, we also note that the Mercer decomposition of this product kernel can be obtained easily from the Mercer decompositions of the constituent kernels. In particular, one has

$$\begin{aligned}
K((\mathbf{x}_1, \mathbf{x}_2), (\mathbf{x}_1', \mathbf{x}_2')) = \sum_{i}^{d_1}\sum_{j}^{d_2} &\gamma_i^{(1)}\gamma_j^{(2)} e_i^{(1)}(\mathbf{x}_1)e_j^{(2)}(\mathbf{x}_2) \\
&\times e_i^{(1)}(\mathbf{x}_1')e_j^{(2)}(\mathbf{x}_2'),
\end{aligned} \tag{21}$$

where $\{\gamma_i^{(1,2)}\}$ and $\{e_i^{(1,2)}\}$ are the nonzero eigenvalues and corresponding eigenfunctions for the respective kernels, and $d_{1,2}$ is used to denote the number of nonzero eigenvalues. In particular, one can notice that here the eigenfunctions and eigenvalues of the product kernel are simply the products of those of the individual kernels.

Let us now move on to the second situation. In particular, let us assume that we are given two kernels $K_1 : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ and $K_2 : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$, and we would like to build a kernel $K : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$, with a different feature space. Similarly to before, one option is to simply take the product of the kernels, i.e.,

$$K(\mathbf{x}, \mathbf{x}') := K_1(\mathbf{x}, \mathbf{x}')K_2(\mathbf{x}, \mathbf{x}'). \tag{22}$$

Once again, from the explicit feature map perspective, one has that

$$K(\mathbf{x}, \mathbf{x}') = \langle F(\mathbf{x})|F(\mathbf{x}')\rangle_{\mathcal{F}}, \tag{23}$$

where $F : \mathcal{X} \to \mathcal{F} = \mathcal{F}_1 \otimes \mathcal{F}_2$ via

$$|F(\mathbf{x})\rangle = |F^{(1)}(\mathbf{x})\rangle \otimes |F^{(2)}(\mathbf{x})\rangle. \tag{24}$$

In particular, we see that this kernel has the enhanced feature space $\mathcal{F} = \mathcal{F}_1 \otimes \mathcal{F}_2$. Again, we note that this construction again clearly generalizes to the case of N kernels $\{K_i\}$, with feature maps $\{F^{(i)}\}$. We stress, however, that in this case the Mercer decomposition of the product kernel is not immediately obtained from the Mercer decompositions of the constituent kernels, as the functions $\{e_i(\mathbf{x})e_j(\mathbf{x})\}|_{i,j}$ are not necessarily orthogonal themselves.

In summary, we see that taking the product of two different kernels—or alternatively, the tensor product of feature vectors—allows one to obtain a kernel with either

(1) an extended domain and feature space (if the constituent kernels act on different domains);

(2) an extended feature space (if the constituent kernels act on the same domain).

Computational complexity of evaluating product kernels

Consider a kernel K_F, for which $K_F(\cdot, \cdot) = \langle F(\cdot)|F(\cdot)\rangle$, where $F : \mathcal{X} \mapsto \mathbb{R}^d$. Assuming knowledge of the feature map F, the naive way to evaluate $K_F(\mathbf{x}, \mathbf{x}')$ is to first construct the feature vectors $|F(\mathbf{x})\rangle, |F(\mathbf{x}')\rangle \in \mathbb{R}^d$, and then take the inner product, which requires d multiplications. However, as mentioned in Sec. II C, for certain kernels $K_F(\mathbf{x}, \mathbf{x}')$ can in fact be evaluated much more efficiently than via the naive method. Indeed, this *kernel trick* is central to the success of kernel methods, as it facilitates the use of extremely high-dimensional (even infinite) feature spaces [5,6]. Here, we briefly examine the extent to which a "kernel trick" is possible for product kernels.

To this end, let us consider a product kernel built from N kernels $\{K_i : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}\}$, where each K_i is associated with a feature map $F^{(i)} : \mathcal{X} \to \mathbb{R}^d$. Additionally, let us assume that each kernel K_i admits a kernel trick thatallows the kernel to be evaluated with $T_i < d$ multiplications. We now have the following options for evaluating the product kernel $K(\mathbf{x}, \mathbf{x}')$:

(1) First, explicitly construct the feature vectors $|F(\mathbf{x})\rangle, |F(\mathbf{x}')\rangle \in \mathbb{R}^{d^N}$, then take the inner product. This requires $O(d^N)$ multiplications—i.e., the cost scales exponentially with the number of constituent kernels.

(2) For each $i \in [N]$, evaluate $K_i(\mathbf{x}, \mathbf{x}')$ by first evaluating $|F^{(i)}(\mathbf{x})\rangle, |F^{(i)}(\mathbf{x}')\rangle \in \mathbb{R}^d$, then taking the inner product. Finally, take the product of all constituent kernel evaluations. This requires $O(Nd)$ multiplications.

(3) For each $i \in [N]$, simply use the appropriate kernel trick to evaluate $K_i(\mathbf{x}, \mathbf{x}')$, then take the product of all constituent kernel evaluations. This requires $\prod_i T_i < Nd$ multiplications.

As such, we see that product kernels do indeed admit kernel tricks, which allows for the kernel to be evaluated significantly more efficiently than via the naive method of explicitly constructing the feature vector and taking the inner product.

B. Entangled tensor kernels

In this section, we generalize the simple product kernel construction to obtain ETKs. However, for ease of presentation, from this point on we will consider explicitly only the second perspective above—i.e., constituent kernels that all share the same domain. It should, however, be clear from the discussion in the previous section how this can be easily generalized to the setting of constituent kernels with different domains. Additionally, in this section we will also only consider feature maps $F : \mathcal{X} \mapsto \mathbb{R}^d$, for some finite d. However, as we have discussed in Sec. II C, for any kernel K_F defined via a feature map $F : \mathcal{X} \mapsto \mathcal{F}$, where $\mathcal{F}$ is any finite

d-dimensional real Hilbert space, one can always find an equivalent feature map $\tilde{F} : \mathcal{X} \mapsto \mathbb{R}^d$, for which $K_F = K_{\tilde{F}}$.

With this in mind, let us again consider two kernels $K_1 : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ and $K_2 : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, with corresponding feature maps $F^{(1)} : \mathcal{X} \to \mathbb{R}^{d_1}$ and $F^{(2)} : \mathcal{X} \to \mathbb{R}^{d_2}$. Given a $d_1 d_2 \times d_1 d_2$ PSD matrix C, we now define the entangled tensor kernel $K_C : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ via

$$K_C(\mathbf{x}, \mathbf{x}') = \langle F^{(1)}(\mathbf{x})| \langle F^{(2)}(\mathbf{x})|C|F^{(1)}(\mathbf{x}')\rangle |F^{(2)}(\mathbf{x}')\rangle. \quad (25)$$

To see that this is indeed a valid kernel, first note that we can always write $C = B^\top B$, where B is an $m \times d_1 d_2$ matrix and $m = \mathrm{rank}(C)$. With this, we can see that

$$K_C(\mathbf{x}, \mathbf{x}') = \langle F_C(\mathbf{x})|F_C(\mathbf{x}')\rangle, \quad (26)$$

where $F_C : \mathcal{X} \mapsto \mathbb{R}^m$ via

$$|F_C(\mathbf{x})\rangle = B|F^{(1)}(\mathbf{x})\rangle |F^{(2)}(\mathbf{x})\rangle. \quad (27)$$

As K_C is calculated via an inner product in a feature space, it is indeed symmetric and positive semidefinite, and therefore a valid kernel. Moreover, from Eq. (27) we can see that for an ETK K_C, the feature vector $|F_C(\mathbf{x})\rangle$ is no longer necessarily a tensor product of local feature vectors—i.e., it may not be possible to write $|F_C(\mathbf{x})\rangle = |\tilde{F}^{(1)}(\mathbf{x})\rangle |\tilde{F}^{(2)}(\mathbf{x})\rangle$ for all $\mathbf{x} \in \mathcal{X}$, for two local feature maps $\tilde{F}^{(1)} : \mathcal{X} \to \mathbb{R}^{d_1}$ and $\tilde{F}^{(2)} : \mathcal{X} \to \mathbb{R}^{d_2}$. By analogy with the definition of entanglement for a many-body quantum state, we *define* the feature map F_C as being entangled whenever it cannot be written as the tensor product of local feature maps. Whenever F_C is an entangled feature map according to this definition, then we call the kernel K_C defined via Eq. (26) an *entangled tensor kernel*. In particular, we stress that the word "entangled" is used in this article simply to represent the impossibility of writing a feature map as a tensor product of local feature maps.

Additionally, note that when C is full rank, then K_C is calculated via the inner product in $\mathbb{R}^{d_1 d_2}$ defined by C. Moreover, when $C = I$ this is the standard dot product, and one recovers the product kernel. To aid in developing intuition, we have provided in Appendix A a variety of examples, illustrating how various classical kernels can be understood as ETKs.

We stress, however, that unlike the case of product kernels, the Mercer decomposition of an ETK does not follow immediately from the Mercer decomposition of the constituent kernels. We address this issue in Appendix B, where we provide an algorithm for obtaining the Mercer decomposition of an ETK.

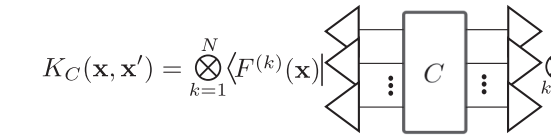
Finally, while the above example only considered two constituent kernels, it should be clear that the construction can easily be generalized to multiple constituent kernels. Specifically, given N feature maps $F^{(k)} : \mathcal{X} \to \mathbb{R}^{d_k}$ (some of which could be the same), we can create a feature map by entangling the feature vectors of $\{F^{(i)}\}$ via (the square root of) any $\prod_k d_k \times \prod_k d_k$ PSD matrix C. More specifically, in this case one has

$$K_C(\mathbf{x}, \mathbf{x}') = \langle F(\mathbf{x})|C|F(\mathbf{x}')\rangle, \quad (28)$$

where we have defined

$$|F(\mathbf{x})\rangle := \bigotimes_{k=1}^{N} |F^{(k)}(\mathbf{x})\rangle. \quad (29)$$

In the following sections, it will be extremely convenient to use tensor network notation [25,26] for the representation of such kernels. To this end, we note that one can represent the ETK K_C using tensor network notation via

$K_C(\mathbf{x}, \mathbf{x}') =$ 

(30)

where each small (left-) right-hand side looking triangle represents the local feature vector $(\langle F^{(k)}(\mathbf{x})|)\ |F^{(k)}(\mathbf{x})\rangle$ and the line represents the d_k-dimensional index. Alternatively, one could also write

$$K_C(\mathbf{x}, \mathbf{x}') = \langle F_C(\mathbf{x})|F_C(\mathbf{x}')\rangle, \quad (31)$$

with

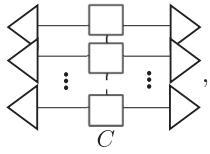
$$|F_C(\mathbf{x})\rangle = B|F(\mathbf{x})\rangle, \quad (32)$$

where $C = B^\top B$. Analogously to the two-feature-map case, we call the feature map *entangled* if $B|F(\mathbf{x})\rangle$ does not admit a tensor product decomposition, or equivalently, $B \neq \bigotimes_{k=1}^{N} B_k$ for any choice of B_ks.

C. Computational complexity of evaluating entangled tensor kernels

We have seen in Sec. III A 1 that whenever the constituent kernels of a product kernel admit a kernel trick, then the product kernel itself admits a kernel trick and can be evaluated efficiently. In this section, we examine the extent to which a kernel trick is possible for entangled tensor kernels. To this end, similarly to Sec. III A 1, let us consider an ETK $K_C : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, as in Eqs. (28) and (29), built from N local feature maps $\{F^{(i)} : \mathcal{X} \to \mathbb{R}^d \mid i \in [N]\}$ and some arbitrary PSD core matrix C. Unfortunately, in this case it is not immediately clear whether one can evaluate $K_C(\mathbf{x}, \mathbf{x}')$ more efficiently than by naively constructing $|F(\mathbf{x})\rangle, |F(\mathbf{x}')\rangle \in \mathbb{R}^{d^N}$ and then evaluating $\langle F(\mathbf{x})|C|F(\mathbf{x}')\rangle$, which requires $O(d^{2N})$ multiplications. However, by exploiting the tensor product structure of the feature space, we can indeed sometimes evaluate K_C more efficiently than via this naive method.

To be more specific, as the feature space of the ETK has the tensor product structure $\mathcal{F} = (\mathbb{R}^d)^{\otimes N}$, one can represent the core tensor as an N-site MPO. In this case, we have that

$K_C(\mathbf{x}, \mathbf{x}') =$ 

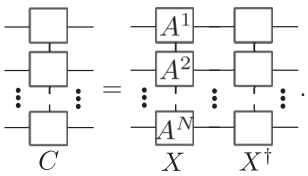
(33)

where, for convenience, we have omitted labeling the small triangles (local feature vectors) in the above diagram. Moreover, the tensor network on the right-hand side of Eq. (33) can be contracted using $O(d^2 \chi^2 N)$ multiplications, where χ is the *bond dimension* of the MPO representation of C [25,26].

In particular, we see that if $\chi = \text{poly}(N)$, then $K_C(\mathbf{x}, \mathbf{x}')$ can be evaluated efficiently with respect to N. To summarize, if C is specified via a polynomial bond-dimension MPO, then there does indeed exist a kernel trick, via tensor network contractions, which allows one to evaluate $K_C(\mathbf{x}, \mathbf{x}')$ significantly more efficiently than via the naive approach. Said another way, given an ETK specified by a core tensor C, the existence of a polynomial bond-dimension MPO representation of the core tensor is a *sufficient condition* for the efficient evaluation of the ETK, assuming that one can efficiently evaluate all local feature maps. Additionally, with this perspective we can give an alternative equivalent definition of an *entangled* feature map, which clarifies the sense in which we use the word entangled in this article. More specifically, the kernel defined via Eq. (33) has an *entangled* feature map whenever the minimum bond dimension of the MPO representation of C is strictly greater than 1. Conversely, if the maximum bond dimension of this MPO is equal to 1, then K_C has a product feature map and is therefore a product kernel. Note that such product kernels are still ETKs in our terminology; they correspond to ETKs whose core tensor$C$ is a product MPO. Therefore, with slight abuse of notation, we also refer to product kernels as ETKs, with a *product* feature map.

Given this, if one wants to specify an ETK, with the guarantee that it can be evaluated efficiently, it seems natural to simply specify a polynomial bond-dimension MPO core tensor. Unfortunately, however, the situation is not as simple as it might seem. In particular, the matrix defined by the MPO (after appropriate grouping of indices) is required to be positive semidefinite, and unfortunately, it is not immediately clear how one can ensure positive semidefiniteness of the global core tensor through constraints on the local tensors, which specify the MPO representation. In fact, if one attempts to define an ETK by specifying an MPO core tensor, it is provably NP-hard to check that the given MPO indeed defines a positive semidefinite matrix, and therefore a valid ETK core tensor [32].

Luckily, however, this is exactly the same problem encountered when trying to use matrix product operators to parametrize density matrices of many-body quantum systems, which are also required to be positive semidefinite (and Hermitian) [32]. As such, we can adopt the solution from that context, which is to use *locally purified matrix product operators* (LP MPOs), which are positive semidefinite and Hermitian by construction [33,34]. More specifically, we can specify the core tensor via an MPO representation $\{A^1, \ldots, A^{(N)}\}$ for an operator X, from which the core tensor is defined via $C = XX^\dagger$, which is guaranteed to be positive semidefinite (and Hermitian). Diagrammatically, such core tensors are represented as follows:



(34)

We note that each tensor $A^{(i)}$ in the MPO representation of X has dimensions $d \times \chi^2 \times p$, where χ is the bond dimension and p is the so-called purification dimension. When the core tensor C is represented by a tensor network in this way, we will refer to it as a locally purified core tensor. From the above discussion, it should now be clear that entangled tensor kernels, specified via locally purified core tensors with polynomial (in N) bond dimension and purification dimension, can be both evaluated efficiently, and efficiently verified as valid kernels.

IV. QUANTUM KERNELS ARE ENTANGLED TENSOR KERNELS

In this section, we provide evidence for the following statement:

All embedding quantum kernels are entangled tensor kernels

In particular, we show that *any* embedding quantum kernel K_U can be written as an ETK, whose local feature maps depend only on the data-dependent gates of the data-dependent quantum circuit U. As we will discuss at length in Sec. V, this perspective on quantum kernels is interesting, because the ETK form of a quantum kernel allows insights into both its inductive bias and classical simulability, which are not necessarily clear from the circuit description of the quantum kernel.

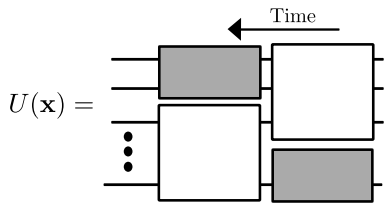
To show this, we start in Sec. IV A by introducing a *standard form* for the data-dependent quantum circuits, which define quantum kernels. This standard form then allows us, in Sec. IV B, to use tensor network techniques to express any embedding quantum kernel as an ETK.

A. A standard form for data-dependent quantum circuits

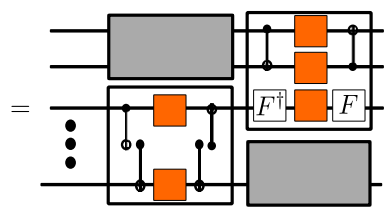
We start by noting that any data-dependent quantum circuit $U(\mathbf{x})$ can be written as a quantum circuit containing only nonparametrized gates, and data-dependent single-qubit gates of the form

$$U_\phi(\mathbf{x}) = e^{-iZ\phi(\mathbf{x})/2}, \tag{35}$$

where $\phi : \mathbb{R}^d \to \mathbb{R}$ is some preprocessing function and Z is the single-qubit Pauli-Z operator. This is achieved via successive Givens rotations [35], as described in detail in Appendix C. Diagrammatically, this decomposition yields



(36)

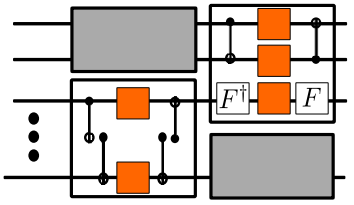


(37)

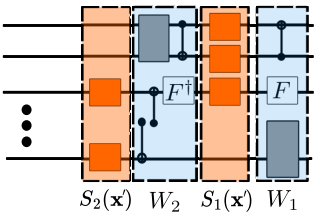
where gray boxes consist of nonparametrized unitaries, white boxes are arbitrary data-dependent unitaries, orange boxes are

data-dependent Pauli-Z rotations, and the F gates are defined in Appendix C. With this in hand, we will from now on assume that any data-dependent quantum circuit is in this form.

In addition, it will be convenient to further rewrite the circuit $U(\mathbf{x})$ as alternating layers of data-independent and data-dependent gates. Diagrammatically, we have

$U(\mathbf{x}) =$ 

(38)

$=$ 

(39)

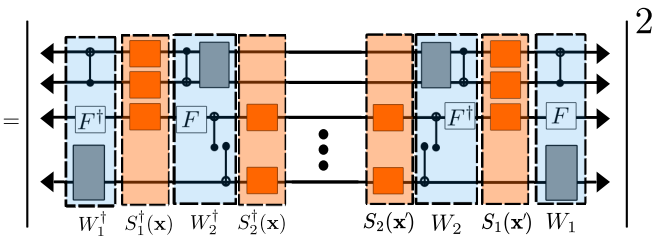
$$= \prod_{j=1}^{L} S_j(\mathbf{x}) W_j, \tag{40}$$

where $\{W_j\}$ are nonparametrized (data-independent) unitaries, and we define $S_j(\mathbf{x}) := (\bigotimes_{k=1}^{n} e^{-i\phi_{jk}(\mathbf{x})Z_k/2})$, with Z_k denoting the Pauli-Z operator on the kth qubit. Note that if there is no data-dependent unitary on qubit k in the jth data-dependent layer, then we simply set $\phi_{jk}(\mathbf{x}) = 0$ for all $\mathbf{x}$. We will consider circuits in the form of Eq. (40) to be in *standard form*.

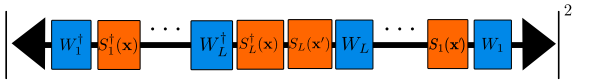
B. Quantum kernels are entangled tensor kernels

We now show how, using the standard form for data-dependent quantum circuits discussed in the previous section, we can write quantum kernels in a tensor network form, which allows us to easily identify them as entangled tensor kernels. To this end, we use tensor network manipulations to rewrite evaluation of the kernel as follows:

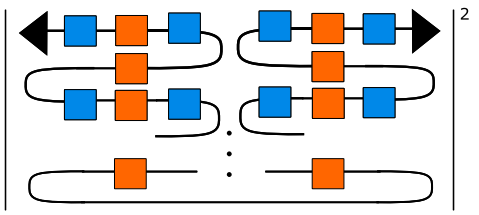
$$K_U(\mathbf{x}, \mathbf{x}') = \left|\langle 0|^{\otimes n} U^\dagger(\mathbf{x}) U(\mathbf{x}')|0\rangle^{\otimes n}\right|^2 \tag{41}$$

$=$ 

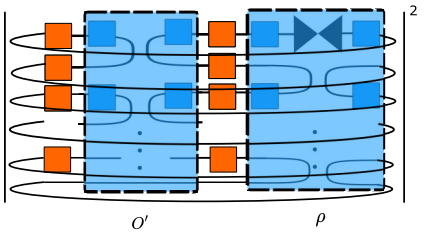
(42)

$=$ 

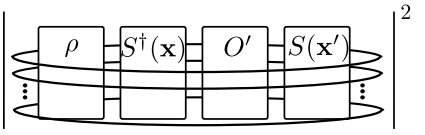
(43)

$=$ 

(44)

$=$ 

(45)

$=$ 

(46)

$$= \left|\mathrm{Tr}\rho S^\dagger(\mathbf{x}) O' S(\mathbf{x}')\right|^2, \tag{47}$$

where from Eq. (43) onward each line in the diagram represents an n-qubit wire, and we have defined

$$O' = \begin{cases} \bigotimes_{j=1}^{(L-1)/2} (W_{2j}^\dagger \otimes I)|\Phi\rangle\langle\Phi|(W_{2j} \otimes I) \otimes I, & \text{if } L \text{ is odd,} \\ \bigotimes_{j=1}^{L/2} (W_{2j}^\dagger \otimes I)|\Phi\rangle\langle\Phi|(W_{2j} \otimes I) \otimes I, & \text{if } L \text{ is even,} \end{cases} \tag{48}$$

$$\rho = \begin{cases} W_1|0\rangle\langle 0|W_1^\dagger \otimes \bigotimes_{j=1}^{(L-1)/2} (I \otimes W_{2j+1})|\Phi\rangle\langle\Phi|(I \otimes W_{2j+1}^\dagger), & \text{if } L \text{ is odd,} \\ W_1|0\rangle\langle 0|W_1^\dagger \otimes \bigotimes_{j=1}^{(L/2-1)} (I \otimes W_{2j})|\Phi\rangle\langle\Phi|(I \otimes W_{2j+1}^\dagger) \otimes |\Phi\rangle\langle\Phi|, & \text{if } L \text{ is even,} \end{cases} \tag{49}$$

$$S(\mathbf{x}) = \bigotimes_{j=1}^{L} S_j(\mathbf{x}), \tag{50}$$

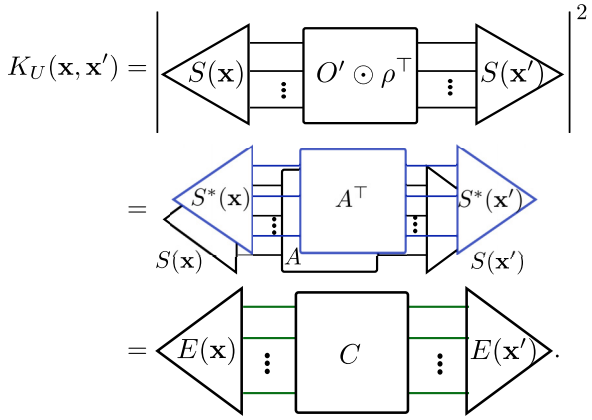
$$|\Phi\rangle\langle\Phi| = \sum_{i,j=1}^{2^n} |ii\rangle\langle jj|. \tag{51}$$

We then proceed by using the equality

$$\mathrm{Tr} D^\dagger A D B = \langle D|(A \odot B^\top)|D\rangle, \tag{52}$$

where $\odot$ represents the Hadamard (elementwise) product, D is a diagonal matrix, and $|D\rangle$ denotes the vector whose elements

are entries of diagonal matrix D. Using this equality and the fact that $S(\mathbf{x})$ is a tensor product of diagonal matrices, the quantum kernel becomes



(53)

In the above, we have defined

$$A := (O' \odot \rho^\top), \tag{54}$$

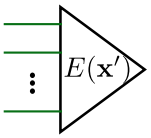
$$C := A^\top \otimes_v A, \tag{55}$$

$$E(\mathbf{x}) := S^*(\mathbf{x}) \otimes_v S(\mathbf{x}), \tag{56}$$

where $\otimes_v$ is a "vertical nearest-neighbor" tensor product, which juxtaposes two equal-dimension tensors "vertically" such that each index of the first tensor becomes the nearest neighbor of the corresponding index of the second tensor. Said another way, as illustrated above, one performs $\otimes_v$ by writing one tensor on top, and slightly shifted vertically of the other tensor, so that their legs are interlaced. To make things explicit, for C we have that

$$\begin{aligned} C_{i_1 i_2 \ldots i_{2nL}; j_1 j_2 \ldots j_{2nL}} =& A^\top_{i_1 i_3 \ldots i_{2nL-1}; j_1 j_3 \ldots j_{2nL-1}} \\ & \times A_{i_2 i_4 \ldots i_{2nL}; j_2 j_4 \ldots j_{2nL}}, \end{aligned} \tag{57}$$

where one should keep in mind that each black and blue wire in the diagram above in fact represents n two-dimensional wires. Additionally, recalling the definition of $S(\mathbf{x})$, and the ordering implied by $\otimes_v$, we see that $E(\mathbf{x})$ is now an nL product of four-dimensional vectors, which is given by



$$= \bigotimes_{j=1}^{L} \left(\bigotimes_{k=1}^{n} \begin{pmatrix} 1 \\ e^{i\phi_{jk}(\mathbf{x}')} \\ e^{-i\phi_{jk}(\mathbf{x}')} \\ 1 \end{pmatrix} \right) = \bigotimes_{j=1}^{L} \left(\bigotimes_{k=1}^{n} |E^{(jk)}(\mathbf{x}')\rangle \right). \tag{58}$$

Moreover, we observe that the core matrix C is complex valued, and that the complex vector $|E(\mathbf{x})\rangle$ possesses many redundant 1's in known indices. We can remove this redundancy by introducing an isometry

$$P := \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & i \\ 0 & 1 & -i \\ 1 & 0 & 0 \end{pmatrix} = \frac{1}{\sqrt{2}} (|I\rangle\rangle\langle 0| + |X\rangle\rangle\langle 1| + |Y\rangle\rangle\langle 2|), \tag{59}$$

where for some matrix M, we have defined $|M\rangle\rangle$ as its column-major vectorization. From now on, for notational simplicity, we will set $N := nl$ and combine "layer, qubit" index jk into one index $k \in \{1, \ldots, N = nl\}$. This gives us the relation

$$|E^{(k)}(\mathbf{x})\rangle = \begin{pmatrix} 1 \\ e^{i\phi_k(\mathbf{x})} \\ e^{-i\phi_k(\mathbf{x})} \\ 1 \end{pmatrix} \tag{60}$$

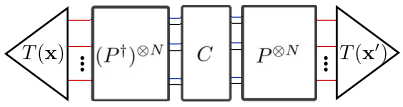
$$= \sqrt{2} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & i \\ 0 & 1 & -i \\ 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \cos\phi_k(\mathbf{x}) \\ \frac{1}{\sqrt{2}} \sin\phi_k(\mathbf{x}) \end{pmatrix} \tag{61}$$

$$:= 2P|T^{(k)}(\mathbf{x})\rangle. \tag{62}$$

We have set $\langle T^{(k)}(\mathbf{x})|T^{(k)}(\mathbf{x})\rangle = 1$, for normalization. Additionally, when tensors lose their superscripts, one should understand them as the N-fold tensor product of the tensors with the same symbol—for example, $|T(\mathbf{x})\rangle = \bigotimes_{k=1}^{N} |T^{(k)}(\mathbf{x})\rangle$. Now, by exploiting Eq. (60), we can obtain

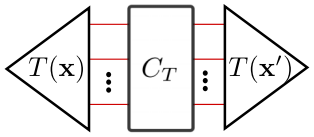
$$K_U(\mathbf{x},\mathbf{x}') = \langle E(\mathbf{x})|C|E(\mathbf{x}')\rangle \tag{63}$$

$= 4^N \times$



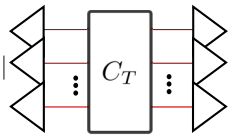
(64)

$:=$



(65)

$= \bigotimes_{k=1}^{N} \langle T^{(k)}(\mathbf{x})|$



$\bigotimes_{k=1}^{N} |T^{(k)}(\mathbf{x})\rangle,$

(66)

where $C_T := 4^N (\bigotimes_k P^\dagger) C (\bigotimes_k P)$ and now red lines indicate three-dimensional wires. We added the last equality to clearly show the tensor product structure of $|T(\mathbf{x})\rangle$. Given this, we see by comparing Eq. (66) with Eq. (30) that K_U is indeed an entangled tensor kernel, with local feature maps

$$|T^{(k)}(\mathbf{x})\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ \cos\phi_k(\mathbf{x}) \\ \sin\phi_k(\mathbf{x}) \end{pmatrix}, \tag{67}$$

and a quantum-circuit-generated symmetric PSD core tensor C_T.

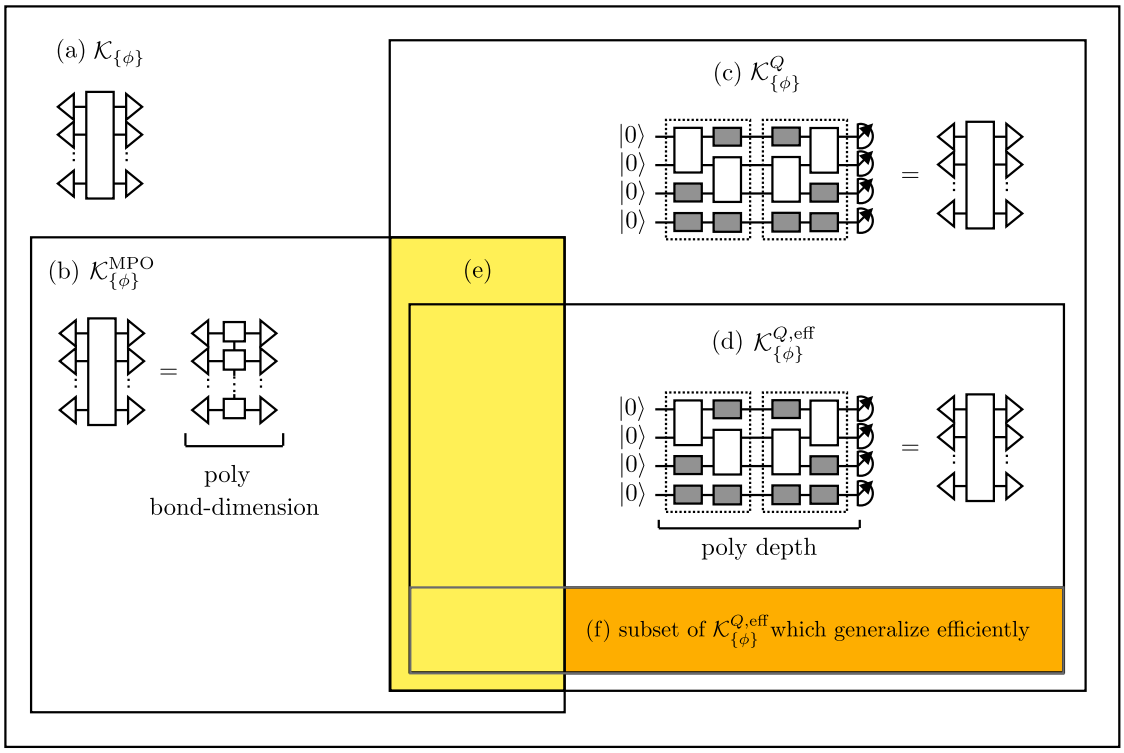


FIG. 2. An overview of the conjectured landscape of ETKs, for some fixed set of local feature maps derived from data preprocessing functions $\{\phi\}$. Region (a) is the set of *all* such ETKs, and region (b) is the subset of those ETKs whose core tensor admits a decomposition into a polynomial bond-dimension MPO and can therefore be evaluated efficiently classically (when given this MPO). Region (c) contains all those ETKs that can be evaluated using an embedding quantum kernel, and region (d) is the subset of those ETKs that can be evaluated using a polynomial depth quantum circuit—i.e., those that can be evaluated efficiently quantumly. In Sec. II C 1, we discuss the *inductive bias*, which distinguishes quantum kernels that are in region (d), but not in region (b). Region (f) is the subset of efficient quantum kernels that also generalize efficiently (for some problem of interest). We identify these quantum kernels as naturally interesting quantum kernels for further exploration and discuss candidates in Sec. V B. Region (e) is the subset of quantum kernels that can in principle be evaluated efficiently classically. The existence of this set suggests that ETKs with poly-bond-dimension MPOs can sometimes be used to *dequantize* quantum kernels, which is discussed in Sec. V C. Whether or not region (b) is a proper subset of region (c) is an open question.

Finally, we note that we can also identify

$$(C_T)_{\mathbf{ij}} = 4^N \times \tfrac{1}{2^N}\mathrm{Tr}\mathcal{P}_{\mathbf{i}}A\mathcal{P}_{\mathbf{j}}A, \quad \mathbf{i},\mathbf{j} \in \{0,1,2\}^N, \tag{68}$$

where $\mathcal{P}_{\mathbf{i}}$ is the Pauli string associated with index string $\mathbf{i}$. In light of this, one could say that C_T is a (truncated) rescaled Pauli transfer matrix of the linear map $\mathcal{A} : \mathbf{M} \mapsto A\mathbf{M}A$. Additionally, we note that the matrix C_T is also symmetric and now has real-valued elements.

V. THE LANDSCAPE OF ENTANGLED TENSOR KERNELS

In light of the results of the previous section, we proceed here to further examine the landscape of ETKs—illustrated in Fig. 2—with the goal of better understanding the relationship between quantum kernels and classical ETKs. In particular, we will also highlight in Secs. V A– V C the implications this relationship has for the design of both potentially interesting quantum kernels and "classical surrogate kernels" for dequantizing quantum kernel methods.

To this end, we start by noting that the local feature maps $\{|T^{(k)}(\mathbf{x})\rangle\}$ of the ETK representation of a quantum kernel are uniquely determined by a set of data preprocessing functions $\{\phi\} := \{\phi_k \,|\, k \in [N]\}$. With this in mind, we assume some fixed set of preprocessing functions $\{\phi\}$, and start by defining $\mathcal{K}_{\{\phi\}}$, the set of *all* ETKs using the associated local feature maps $\{|T^{(k)}(\mathbf{x})\rangle\}$, via

$$\mathcal{K}_{\{\phi\}} := \{K_C \,:\, K_C(\mathbf{x},\mathbf{x}') = \langle T(\mathbf{x})|C|T(\mathbf{x})\rangle\}, \tag{69}$$

where C is positive semidefinite. As we have discussed in Sec. III C, not all ETKs in $\mathcal{K}_{\{\phi\}}$ can be evaluated efficiently. However, we have seen that whenever the core tensor C can be represented by an MPO with polynomial bond dimension, then the associated ETK *can* be efficiently evaluated classically. In light of this, we define the subset of $\mathcal{K}_{\{\phi\}}$ whose core tensor C admits a decomposition into an MPO with polynomial bond dimensions as $\mathcal{K}^{\mathrm{MPO}}_{\{\phi\}}$. By construction, all ETKs in $\mathcal{K}^{\mathrm{MPO}}_{\{\phi\}}$ can be evaluated efficiently classically, when given the MPO representation of the core tensor C.

We would now like to understand how quantum kernels fit into the picture. To this end, let us denote by $C_T(\{W\},\{\phi\})$ the core tensor C_T that one obtains, via the derivation in Sec. IV B, when starting from a quantum kernel K_U with some data-dependent quantum circuit $U(\mathbf{x}) = \prod_{j=1}^{L} S_j(\mathbf{x})W_j$, which uses the preprocessing functions $\{\phi\}$. This allows us to

define the set of "quantum ETKs" $\mathcal{K}^{Q}_{\{\phi\}}$ via

$$\mathcal{K}^{Q}_{\{\phi\}} := \{K_C \,:\, K_C(\mathbf{x},\mathbf{x}') = \langle T(\mathbf{x})|C_T(\{W\},\{\phi\})|T(\mathbf{x}')\rangle\}. \tag{70}$$

In words, $\mathcal{K}^{Q}_{\{\phi\}}$ is nothing but the set of entangled tensor kernels one can realize by using quantum kernels K_U with valid data-dependent quantum circuits $U(\mathbf{x}) = \prod_{j=1}^{L} S_j(\mathbf{x})W_j$, using preprocessing functions $\{\phi\}$. As any "quantum-generated" core tensor $C_T(\{W\},\{\phi\})$ is PSD, we immediately have that

$$K^{Q}_{\{\phi\}} \subseteq K_{\{\phi\}}, \tag{71}$$

i.e., as illustrated in Fig. 2, "quantum ETKs" are a subset of all possible ETKs using the same local feature maps.

Consider now some ETK $K_C \in \mathcal{K}^{Q}_{\{\phi\}}$. By construction, we know that there exists a data-dependent unitary $U(\mathbf{x})$, such that $K_C = K_U$. As such, one could evaluate K_C by executing the quantum circuit shown in Fig. 1. When implementing K via this route, the efficiency of the implementation of the kernel depends on the depth of the quantum circuit $U(\mathbf{x})$—when this circuit can be efficiently implemented, then the kernel can be evaluated efficiently. Motivated by this observation, we call the subset of quantum ETKs $\mathcal{K}^{Q}_{\{\phi\}}$ that can be efficiently evaluated on a quantum computer $\mathcal{K}^{Q,\mathrm{eff}}_{\{\phi\}}$.

To summarize, at this stage we have established the following relationships, also illustrated in Fig. 2:

$$\mathcal{K}^{Q,\mathrm{eff}}_{\{\phi\}} \subset \mathcal{K}^{Q}_{\{\phi\}} \subseteq \mathcal{K}_{\{\phi\}} \cup \mathcal{K}^{\mathrm{MPO}}_{\{\phi\}}. \tag{72}$$

With this in hand, we would now ideally like to understand the relationships between the set of quantum kernels, which can be evaluated efficiently with a quantum computer $\mathcal{K}^{Q,\mathrm{eff}}_{\{\phi\}}$, and the set of ETKs, which can be evaluated efficiently classically via tensor network contractions $\mathcal{K}^{\mathrm{MPO}}_{\{\phi\}}$. We conjecture that the relationships between these sets are as illustrated in Fig. 2. With this in mind, the following questions are of particular interest to us:

(1) Do there exist kernels in $\mathcal{K}^{Q,\mathrm{eff}}_{\{\phi\}}$ that are *not* in $\mathcal{K}^{\mathrm{MPO}}_{\{\phi\}}$? Said another way: Are there quantum kernels that can be evaluated efficiently with a quantum computer, but do not admit efficient evaluation via the ETK perspective? If the answer to this question is "yes," then what properties do these kernels have? Are they potentially useful? We explore this question and provide some (partial) answers in Secs. V A and V B.

(2) Similarly, are there kernels in $\mathcal{K}^{\mathrm{MPO}}_{\{\phi\}}$ that are *not* in $\mathcal{K}^{\mathrm{Q,eff}}_{\{\phi\}}$? We conjecture that the answer to this question is "yes," but leave a resolution of this to future work.

(3) Finally, given the existence of kernels in the intersection $\mathcal{K}^{Q,\mathrm{eff}}_{\{\phi\}} \cap \mathcal{K}^{\mathrm{MPO}}_{\{\phi\}}$, to what extent can we use this to *dequantize* quantum kernel methods. We discuss the potential and limitations in this regard in Sec. V C.

Before proceeding, we comment on projected quantum kernels. Projected quantum kernels introduced in Ref. [15] use classical functions of quantities derived from reduced states, so they do not directly fit our ETK construction in full generality. However, when a projected kernel can be written as an embedding kernel on an enlarged feature map (e.g., by explicitly including the measured local features), the ETK lens may still apply to that embedding representation. Moreover, recent works have provided unifying perspectives [11,12]; in particular, it has been shown that *any* reasonable quantum kernel can be written as an embedding quantum kernel [11]. We leave a systematic treatment of projected kernels in the ETK language to future work.

A. Quantum kernels can potentially realize superpolynomial bond-dimension ETKs

As illustrated in Fig. 2, there is the possibility of quantum kernels that

(1) can be evaluated efficiently via the execution of a polynomial depth quantum circuit;

(2) do not admit a polynomial bond-dimension MPO decomposition for the core tensor of the ETK representation.

We refer to such quantum kernels as having "superpolynomial bond dimension." As per the discussion in Sec. III C, such kernels cannot be evaluated efficiently classically by simply contracting the tensor network defining the ETK representation. As such, one can think of "superpolynomial bond dimension" as an *inductive bias* that can be possessed by efficient quantum kernels, but not by ETKs that are classically efficient to evaluate via the tensor network contraction method discussed in Sec. III C.

We stress, however, that having superpolynomial bond dimension is a *necessary but not sufficient* condition for a quantum kernel to be hard to evaluate classically. We cannot rule out the existence of ETKs with superpolynomial bond dimension, which can nevertheless be evaluated efficiently classically via some other method than contracting the tensor network representation of the ETK. In this sense, we can view "superpolynomial bond dimension" as an inductive bias which is *easily* accessible to efficient quantum kernels, but not so obviously accessible to kernels that are efficient to evaluate classically. Given this, we leave to future work the following questions:

(1) How can we design quantum kernels to ensure that they have superpolynomial bond dimension?

(2) Can one construct and analyze examples of superpolynomial bond-dimension ETKs, which can nevertheless be evaluated efficiently classically?

B. Do useful quantum kernels exist?

In the previous section, we discussed how having superpolynomial bond dimension is a necessary but not sufficient condition for a quantum kernel to be hard to evaluate classically. However, for a quantum kernel to be potentially *useful* for a specific problem, one should both (1) provide evidence that the quantum kernel cannot be evaluated efficiently classically and (2) provide evidence that the kernel generalizes efficiently for some interesting problem—i.e., that it can generalize well on a polynomially sized dataset of interest. As having superpolynomial bond dimension provides some evidence for being hard to evaluate efficiently classically, we ask here whether there are quantum kernels that have a superpolynomial bond dimension *and* generalize efficiently.

As discussed in Refs. [14,18,28], one way to show that a kernel generalizes efficiently (for some set of problems) is to show that the spectrum of the kernel is concentrated on

a polynomially sized subset of eigenvalues. At a high level, such kernels will generalize efficiently for target functions that are supported on the corresponding eigenfunctions. At first glance, one might think that this immediately precludes the existence of quantum kernels that both have superpolynomial bond dimension and generalize efficiently! Indeed, one might think that when the spectrum of the kernel is concentrated on polynomially many eigenvalues, one could simply truncate the size of the feature space to some polynomial in the dimension of the data (namely, the feature space spanned by the top eigenfunctions) and therefore evaluate the kernel efficiently classically (by exploiting the truncated core tensor that would admit polynomial bond dimension).

However, here we show that this is *not* the case, and that there are indeed kernels that both have superpolynomial bond dimension and generalize efficiently. We stop short of the true goal of showing the existence of *quantum* kernels with these properties, but discuss why one might expect them to exist.

To begin, consider the kernel

$$K_G(\mathbf{x}, \mathbf{x}') = \langle T(\mathbf{x})|U^\dagger D U|T(\mathbf{x}')\rangle, \tag{73}$$

where D has only $O(\text{poly}(d))$ nonzero diagonal elements, or its values are concentrated on $O(\text{poly}(d))$ elements, where d is the dimension of the local feature maps. The unitary U, however, has $O(\exp(d))$ bond dimensions when represented as an MPO. Using the tools and techniques of Refs. [14,18,28], one can see that K_G will indeed generalize efficiently (for some set of problems). However, the contracted core tensor $C_G := U^\dagger D U$ could have $O(\exp(d))$ bond dimension, placing it outside $\mathcal{K}^{\text{MPO}}_{\{\phi\}}$. As such, this kernel has both a superpolynomial bond dimension *and* generalizes efficiently for some class of problems. Whether $\mathcal{K}^{\text{Q,eff}}_{\{\phi\}}$—the set of *quantum* kernels that can be evaluated efficiently—indeed includes kernels like K_G remains open. However, we conjecture it does, due to the ability of such kernels to generate superpolynomial bond-dimension ETKs.

We note that one can interpret K_G as a model utilizing a (relatively) small function space spanned by eigenfunctions that are highly entangled (in the sense discussed in Sec. III), so that they are classically hard to compute (at least via tensor network contractions). These classically hard eigenfunctions appear as exponentially large sums of trigonometric functions, $\hat{e}_i(\mathbf{x}) = \sum_j U_{ij} T_j(\mathbf{x})$. Even if the form of these eigenfunctions is explicitly known (i.e., one knows all U_{ij}'s), evaluating them is nontrivial because the summation involves contracting high-bond-dimensional tensor networks, which lack concise representations. Thus, we can characterize candidates for *useful* quantum kernels as those with the following properties:

(1) Generalize efficiently: The spectrum of the kernel is concentrated on polynomially many eigenvalues, and therefore the kernel method is biased toward functions spanned by a polynomial number of top eigenfunctions.

(2) Classically hard eigenfunctions: The top eigenfunctions that span the (polynomially large) effective function space cannot be efficiently evaluated using only the given description of the data-dependent circuit $U(\mathbf{x})$.

One final important remark is that the above conditions are only *necessary* (but not sufficient) for useful quantum kernels. More specifically, we cannot rule out the possibility that a clever classical algorithm (different from just contracting the given tensors) might exist to evaluate superpolynomial bond dimension ETKs efficiently.

C. ETKs as classical surrogates for quantum kernels

A standard approach to assessing whether or not one can gain a quantum advantage from a specific quantum algorithm is to ask whether one can use insights into the nature and structure of the quantum algorithm to construct a purely classical algorithm that is both efficient and guaranteed to perform as well as the quantum algorithm. In other words, to ask whether one can *dequantize* the quantum algorithm [36]. For those quantum machine learning algorithms, including quantum kernel methods, which work by optimizing over a model class defined by a parametrized quantum circuit, a natural approach to dequantization is to look for a classical model class whose *inductive bias* is similar to that of the quantum model class, and can be evaluated efficiently classically. Such a model class has come to be known as a *classical surrogate* for the quantum model class, and surrogates for variational quantum machine learning (QML) models for supervised learning have recently begun to be proposed and studied [24,37,38].

From this perspective, the fact that all quantum kernels are ETKs immediately suggests ETKs as a natural classical surrogate for quantum kernels. More specifically, as we have seen in Fig. 2, there exist quantum kernels K_U, which are also in $K^{\text{MPO}}_{\{\phi\}}$, and can therefore be evaluated efficiently classically, when given the MPO description of the ETK core tensor. This seems to immediately suggest that kernel methods using such quantum kernels can be immediately dequantized, and cannot be used to obtain any quantum advantage—even if the circuit $U(x)$ is classically hard to simulate! Specifically, instead of using the quantum circuit to evaluate the kernel, one could just efficiently evaluate the corresponding ETK classically via tensor network contractions. Unfortunately, however, there is the following subtlety:

Given a description of a data-dependent quantum circuit U, we cannot always efficiently obtain the optimal MPO description of the core tensor $C_T(\{W\}, \{\phi\})$.

As such, even if we are promised that a quantum kernel has a polynomial bond-dimension MPO core tensor, we might not be able to efficiently obtain this MPO from the circuit description that we are given.

In light of this, there are three natural strategies to dequantization of quantum kernel methods via ETKs as classical surrogates:

(1) Assess whether the structure of the quantum circuit U allows for efficient extraction of an MPO representation of the corresponding ETK core tensor.

(2) *Learn* a polynomial bond-dimension positive semidefinite MPO core tensor (via optimization of kernel-target alignment, for example, Ref. [10]) and use the corresponding ETK.

(3) Use the structure of the quantum circuit U to make informed guess, or random selection, of a polynomial bond-dimension positive semidefinite MPO core tensor, and use the corresponding ETK (in some ways, analogously to the use

of random Fourier features for dequantizing variational QML [24,38]).

We leave further study of these approaches to dequantization of classical kernel methods to future work; however, we make a few remarks. First, as per Sec. III B, in order to be a valid kernel, we know that the core tensor of an ETK needs to be positive semidefinite. As such, as we have discussed in Sec. III C, a natural approach when trying to guess or learn a suitable ETK core MPO is to use LP MPOs, which are positive semidefinite by construction. Additionally, one might think that only those quantum kernels that are also in $\mathcal{K}^{\mathrm{MPO}}_{\{\phi\}}$ are susceptible to dequantization via the approaches described above. However, this is not necessarily the case. Indeed, even if the optimal MPO representation of the ETK core tensor for a quantum kernel requires superpolynomial bond dimension, it may be the case that a polynomial bond-dimension MPO provides a suitably accurate approximation. Moreover, and counterintuitively, the fact that ETKs in $\mathcal{K}^{\mathrm{MPO}}_{\{\phi\}}$ are more constrained in certain ways than quantum kernels may indeed give them advantages in generalization over many quantum kernels (as discussed in Ref. [24]).

VI. THE POWER OF THE ETK LENS FOR ANALYSIS OF QUANTUM KERNELS

As we have discussed briefly in Sec. II C 1, obtaining the Mercer decomposition of a kernel allows one to extract a variety of quantitative insights into the inductive bias, sample complexity, and generalization capacity of the kernel. Indeed, Refs. [14,17,18] have all obtained a variety of quantitative insights into properties of quantum kernels, via Mercer decompositions of simple examples. These insights have then facilitated concrete recommendations for both the design of quantum kernels and suitable applications.

A. Analysis on single-layer model with arbitrary n-qubit unitary

In this section, in order to showcase the power of the ETK lens for the analysis of quantum kernels, we show how to use the ETK picture to obtain the Mercer decomposition of a family of specific quantum kernels, which goes beyond those previously studied in Refs. [14,17,18]. In particular, previous works have analyzed what we refer to in this work as *product kernels*. In this section, we will show how to use the insights from Sec. IV, together with techniques from Appendix B, to obtain the Mercer decomposition of a specific family of quantum kernels.

More specifically, we study quantum kernels K_U on n qubits, satisfying the following two assumptions.

Assumption 1: Single-layer of data-dependent gates. We assume that the data-dependent unitary specifying the quantum kernel is of the form

$$U(\mathbf{x}) = S(\mathbf{x})W \tag{74}$$

$$= \left(\bigotimes_{k=1}^{n} e^{-i\phi_k(\mathbf{x})Z_k/2}\right)W. \tag{75}$$

This is a special case of the general setting studied in Sec. IV, with $L = 1$.

Assumption 2: Simple preprocessing. We assume that $\mathbf{x} \in [-\pi, \pi]^n$, and that $\phi_k(\mathbf{x}) = \mathbf{x}_k$.

With these assumptions, we start by deriving the ETK representation of the kernel, as per the techniques of Sec. IV. To this end, we see from Eqs. (48) and (49) that, when $L = 1$, one has $O' = I$ and $\rho = W|0\rangle\langle 0|^{\otimes n}W^\dagger := |\psi\rangle\langle\psi|$. We then have from Eq. (53) that

$$K_U(\mathbf{x}, \mathbf{x}') = \langle \mathrm{diag}(S(\mathbf{x}))|(I \odot |\psi\rangle\langle\psi|)|\mathrm{diag}(S(\mathbf{x}'))\rangle \times (\text{c.c.}). \tag{76}$$

In this case, when we apply the Hadamard product with the identity matrix I, it selects the diagonal elements, leading to $(I \odot |\psi\rangle\langle\psi|) = \mathrm{diag}(|\psi\rangle\langle\psi|) := D(\psi^2)$, which is a diagonal matrix whose elements are the squared magnitudes of the components of $|\psi\rangle$. Using this in Eq. (63), we obtain

$$K_U(\mathbf{x}, \mathbf{x}') = \langle E(\mathbf{x})|C|E(\mathbf{x}')\rangle \tag{77}$$

$$= \left(\begin{pmatrix} 1 \\ e^{-i\phi_k(\mathbf{x})} \\ e^{i\phi_k(\mathbf{x})} \\ 1 \end{pmatrix}^{\otimes n}\right)^\top (D(\psi^2) \otimes_v D(\psi^2)) \times \begin{pmatrix} 1 \\ e^{i\phi_k(\mathbf{x}')} \\ e^{-i\phi_k(\mathbf{x}')} \\ 1 \end{pmatrix}^{\otimes n}. \tag{78}$$

We have now arrived at an ETK representation of the quantum kernel. In Sec. IV, we proceeded to use an isometry P to remove redundant components of $|E(\mathbf{x})\rangle$ and obtain a simplified ETK. However, as obtaining the Mercer decomposition requires the orthonormalization of components $|E(\mathbf{x})\rangle$, and the removal of redundant components, we can simply proceed with the Mercer decomposition from here. To this end, we recall from Appendix B that the first step is to obtain an orthonormal basis for the component functions of $|E(\mathbf{x})\rangle$. However, we note that under Assumption 2, we have that

$$|E(\mathbf{x})\rangle = \begin{pmatrix} 1 \\ e^{-i\mathbf{x}_k} \\ e^{i\mathbf{x}_k} \\ 1, \end{pmatrix}^{\otimes n}, \tag{79}$$

and as such all the component functions of $|E(\mathbf{x})\rangle$ are already orthonormal with respect to the inner product $\langle f|g\rangle = \prod_{k=1}^{N}(\frac{1}{2\pi}\int_{-\pi}^{\pi} f^*(\mathbf{x}_k)g(\mathbf{x}_k)d\mathbf{x}_k)$. Indeed, this was the motivation for Assumption 2. In light of this, we can proceed with Step 2 from Appendix B. For orthonormalizing tensor L, we choose

$$L = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -1 & 0 & 0 & 1 \end{pmatrix}^{\otimes n}, \quad L^{-1} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}^{\otimes n}, \tag{80}$$

which satisfies

$$L|E(\mathbf{x})\rangle = \begin{pmatrix} 1 \\ e^{-i\mathbf{x}_k} \\ e^{i\mathbf{x}_k} \\ 0 \end{pmatrix}^{\otimes n}. \tag{81}$$

With this, we then have that

$$K_U(\mathbf{x},\mathbf{x}') = \left(\begin{pmatrix} 1 \\ e^{i\mathbf{x}_k} \\ e^{-i\mathbf{x}_k} \\ 0 \end{pmatrix}^{\otimes n}\right)^{\top} (L^{-1})^{\top} C L^{-1} \begin{pmatrix} 1 \\ e^{-i\mathbf{x}'_k} \\ e^{i\mathbf{x}'_k} \\ 0 \end{pmatrix}^{\otimes n}$$
$$:= \langle \bar{E}(\mathbf{x})|(L^{-1})^{\top} C L^{-1}|\bar{E}(\mathbf{x}')\rangle. \tag{82}$$

Here, C is diagonal and elements are

$$C_{\mathbf{i};\mathbf{i}} := C_{i_1 i_2,\ldots,i_{2N};i_1 i_2,\ldots,i_{2N}} = \psi^2_{i_1 i_3,\ldots,i_{2N-1}} \psi^2_{i_2 i_4,\ldots,i_{2N}}, \tag{83}$$

with $\mathbf{i} \in \{00, 01, 10, 11\}^n$.

Now, we will see how conjugating L affects the elements of C. Let us denote $\hat{C} := (L^{-1})^{\top} C L^{-1}$.

(1) For elements of $\hat{C}_{\mathbf{h};\mathbf{h}}$, where index $\mathbf{h}$ contains the sequence $h_{2k-1}h_{2k} = 00$ for any $k \in [N]$, these elements are the sum of elements $C_{\mathbf{i};\mathbf{i}}$, where the index $\mathbf{i}$ contains $i_{2k-1}i_{2k} = 11$ or $i_{2k-1}i_{2k} = 00$ for the same k such that $h_{2k-1}h_{2k} = 00$.

(2) For indices $\mathbf{i}$ not containing sequences of 00, $\hat{C}_{\mathbf{i},\mathbf{i}} = C_{\mathbf{i},\mathbf{i}}$.

(3) Nondiagonal elements are created, and the associated indices contain 11 sequences.

The next thing we need to do is a truncation where $\bar{E}_{\mathbf{i}}(\mathbf{x}) = 0$. The zero elements occur whenever there is an 11 sequence in the index $\mathbf{i}$, so we remove rows and columns where indices have 11 sequences, and this removes all the nondiagonal elements of $\hat{C}$, giving us

$$K_U(\mathbf{x},\mathbf{x}') = \left(\begin{pmatrix} 1 \\ e^{i\mathbf{x}_k} \\ e^{-i\mathbf{x}_k} \end{pmatrix}^{\top}\right)^{\otimes n} \tilde{C} \begin{pmatrix} 1 \\ e^{-i\mathbf{x}'_k} \\ e^{i\mathbf{x}'_k} \end{pmatrix}^{\otimes n}$$
$$= \sum_{\alpha \in \{00,01,10\}^n} \tilde{C}_{\alpha,\alpha} e^{-i\omega^{\alpha}\cdot(\mathbf{x}-\mathbf{x}')}, \tag{84}$$

where frequency vectors $\omega^{\alpha} \in \{-1, 0, 1\}^n$ are defined as

$$(\omega^{\alpha})_k = \begin{cases} 0, & \text{if } \alpha_{2k-1}\alpha_{2k} = 00, \\ 1, & \text{if } \alpha_{2k-1}\alpha_{2k} = 01, \\ -1, & \text{if } \alpha_{2k-1}\alpha_{2k} = 10, \end{cases} \tag{85}$$

$$\tilde{C}_{\alpha,\alpha} = \sum_{\alpha \in \mathcal{I}_{\alpha}} \psi^2_{\alpha_1\alpha_3,\ldots,\alpha_{2n-1}} \psi^2_{\alpha_2\alpha_4,\ldots,\alpha_{2n}}, \tag{86}$$

and $\mathcal{I}_{\alpha}$ is the set of all indices containing 00 and 11 sequences where $\alpha_{2k-1}\alpha_{2k} = 00$. For example, $\mathcal{I}_{0000} = \{0000, 0011, 1100, 1111\}$, $\mathcal{I}_{0001} = \{0001, 1101\}$, $\mathcal{I}_{0101} = \{0101\}$, and so on. Therefore, the cardinality of the index set becomes $|\mathcal{I}_{\alpha}| = 2^{|\alpha|_{00}}$, where $|\alpha|_{00}$ denotes the number of 00 sequences in α.

We have now finally, in Eq. (84), arrived at the Mercer decomposition of K_U. Indeed, the diagonal elements of $\tilde{C}$ now contain the eigenvalues of the kernel, from which we can extract insights regarding generalization capacity and sample complexity of learning.

To this end, it is useful to first make several remarks on the matrix $\tilde{C}$, and the eigenvalues that it contains. First of all, we note that $\tilde{C}$ has a "Hermiticity" property. Let $\bar{\alpha}$ be the index with 01 and 10 sequences interchanged. Then, we have $\tilde{C}_{\alpha,\alpha} = \tilde{C}_{\bar{\alpha},\bar{\alpha}}$ and $\omega^{\alpha} = \omega^{\bar{\alpha}}$. Therefore, K_U is indeed real and can be written in the form of

$$K_U(\mathbf{x},\mathbf{x}') = \tilde{C}_{\mathbf{0},\mathbf{0}} + \sum_{\alpha \in \Omega^+} 2\tilde{C}_{\alpha,\alpha} \cos(\omega^{\alpha}\cdot(\mathbf{x}-\mathbf{x}')), \tag{87}$$

where Ω^+ is the set of α indices, obtained by retaining only one index between α and $\bar{\alpha}$ in $\{00, 01, 10\}^n$. Second, the redundancy of the frequency ω^{α} is given by $2^{|\alpha|_{00}}$, where $|\alpha|_{00}$ is the number of 00 sequences in α. This results in eigenvalues that tend to concentrate more on indices with many 00 sequences as they have more terms to be added. In other words, low-degree eigenfunctions (cosine functions associated with the low-Hamming-weight frequency ω^{α}) get larger eigenvalues. With these insights, we can now examine the following two cases.

Case A: Approximately uniform pre-encoding state $|\psi\rangle$. In this case, we assume that the pre-encoding state $|\psi\rangle = W|\mathbf{0}\rangle$ is approximately uniform—i.e., all components of ψ^2 are on the order of $1/2^n$. We note that this will typically be the case when W is chosen randomly. One quick observation in this case is that the largest eigenvalue of the kernel $\tilde{C}_{\mathbf{0},\mathbf{0}}$, associated with the function $\mathbf{1}$, will scale as $2^n(1/4^n) = 1/2^n$. The next largest eigenvalues scale as $2^{n-1}(1/4)^n = 1/2^{n+1}$, and they are associated with the n degree-one frequencies like $\omega = \{(1, 0, \ldots, 0), \ldots, (0, 0, \ldots, 1)\}$ and so on. Essentially, we see that the largest eigenvalue decays exponentially. Using the insights from Refs. [14,18,28], we can then immediately conclude that such kernels are not able to generalize well, from only a polynomial number of data samples (we refer to the abovementioned references for precise statements).

Case B: (s,ϵ)-concentrated pre-encoding state $|\psi_{(s,\epsilon)}\rangle$. We say a state $|\psi\rangle$ is (s,ϵ) concentrated, when $\sum_{\alpha\in S}|\psi_{\alpha}|^2 \geqslant (1-\epsilon)$ for some index set with $|S| = s$. We denote an (s,ϵ) concentrated state as $|\psi_{(s,\epsilon)}\rangle$, and use $W_{(s,\epsilon)}$ for unitaries that produce (s,ϵ) concentrated states.

Physically, sparse—$\epsilon = 0$ concentrated—unitaries can be realized by evolution under block-diagonal Hamiltonians, which yield block-diagonal unitaries. Accordingly, one can construct quantum circuits with appropriate symmetries to generate sparse unitaries. Moreover, if the locations of the nonzero matrix elements can be computed efficiently, these unitaries can be implemented efficiently [39]. As an alternative route to concentration, one might use peaked quantum circuits [40], which generate states whose maximum amplitude is larger than $1/\mathrm{poly}(n)$.

For this Case B, let the amplitude squares of $|\psi_{(s,\epsilon)}\rangle$ in the set S be $L \sim \frac{1-\epsilon}{s}$. Then, according to Eq. (86), the largest eigenvalue is $\Omega(L^2) = \Omega(1/s^2)$. This is because there exists at least one index set $\mathcal{I}_{\alpha}$ that contains $|\psi^2_{\alpha_1\alpha_3,\ldots}\psi^2_{\alpha_2\alpha_4,\ldots}| = \Omega(L^2)$. Moreover, note that the number of α's that make $|\psi^2_{\alpha_1\alpha_3,\ldots}\psi^2_{\alpha_2\alpha_4,\ldots}| \sim L^2$ is at most s^2; therefore, the number of leading $\tilde{C}_{\alpha,\alpha}$ terms can be at most s^2, which means that the eigenspectrum from the (s,ϵ) concentrated state is at most concentrated as $(s^2, 1-(1-\epsilon)^2)$.

In light of this, we can conclude that in order to be useful—in the sense of being able to generalize well from polynomially many samples, as discussed in Sec. V B—the unitary W that specifies the kernel needs to be such that the pre-encoded state $|\psi\rangle = W|\mathbf{0}\rangle$ has its amplitudes concentrated on only $O(\text{poly}(n))$ basis states. However, the core tensor C_T constructed from such states can (in principle) be approximated by an MPO with $O(\text{poly}(n))$ bond dimension by truncating exponentially small values $\tilde{C}_{\alpha,\alpha}$ and retaining only $O(\text{poly}(n))$ significant indices α. Furthermore, in this scenario, all eigenfunctions of the kernel can be efficiently computed classically; therefore, any generalizable quantum kernel satisfying Assumptions 1 and 2 admits a description from which they can be evaluated efficiently classically!

However, similarly to what we have discussed previously in Sec. V C, this efficient classical evaluation relies on knowing the polysized set of dominant eigenvalues. As such, if one can efficiently derive an explicit expression for $|\psi\rangle$ [or more precisely (ψ^2)] that allows for the efficient truncation described above, then the kernel can be efficiently dequantized. As we have discussed in Sec. V C, though, in general the problem of constructing an efficient representation of $|\psi\rangle$ is hard, but for many circuits for W this may indeed be possible.

In summary, we see that our analysis in this section has allowed us to clearly link properties of the quantum kernel K_U (satisfying Assumptions 1 and 2) to properties of the data-independent unitary matrix W and corresponding pre-encoded state $W|0\rangle$. Eigenvalues of kernel integral operators are determined by the squared amplitudes of pre-encoded states. We have derived an explicit formula to connect them and this is an observation that could be made through the lens of ETK. These insights can hopefully be used to guide the design of quantum kernels.

B. Numerical experiments

To complement our analysis and discussions above, here we present numerical experiments comparing single-layer quantum kernels from Haar random unitaries W_H (i.e., Case A) and (s,ϵ)-concentrated unitaries $W_{(s,\epsilon)}$ (i.e., Case B). In all simulations, we set $\epsilon = 0.001$, and to avoid notational clutter, we will omit the explicit dependence on ϵ when referring to (s,ϵ)-concentrated models. Also, we call s-concentrated models s-sparse models even though they are only approximately sparse.

1. Eigenvalue scaling

Here, we study the scaling of eigenvalues of the kernel integral operators that come from given unitaries. Note that these are different from the eigenspectra of unitaries or quantum states themselves. To study, we construct 30 random kernel instances and observe the magnitudes of their largest eigenvalues as the number of qubits increases. For runs with the same sparsity s, the positions of the concentrated amplitudes are chosen randomly.

Figure 3 clearly displays the decay of the largest eigenvalues when the pre-encoding state comes from a Haar random unitary. Meanwhile, the largest eigenvalues from the sparse pre-encoding state do not scale with the system size, but are

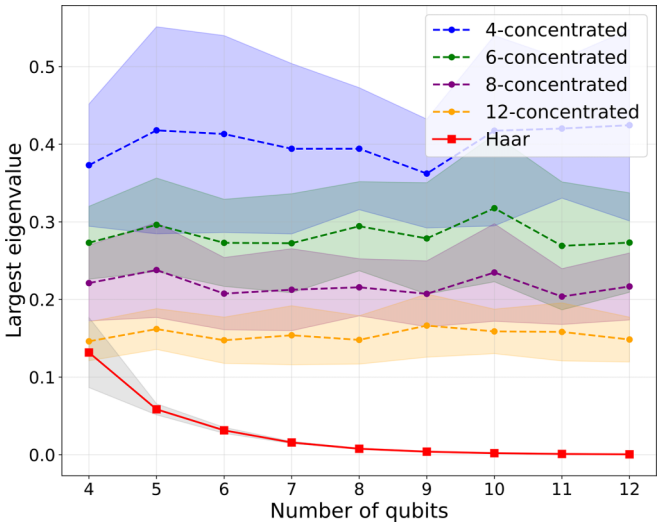


FIG. 3. The scaling of the largest eigenvalues of quantum kernel integral operators with respect to the number of qubits. All data points are averaged over 30 different instances, and shaded regions represent standard deviation. Concentrated positions of each concentrated unitary are chosen randomly for all instances.

only affected by the sparsity. In Fig. 4, we plot all eigenvalues with a log scale. One can clearly see rather nonflat eigenspectra of the s-concentrated models, and their concentration to large values. The large steplike structure of spectra from concentrated models arises because only s components contribute significantly to $|\psi_\alpha|^2$, and these contributions are typically much larger than those from the remaining $2^n - s$ components. As discussed earlier in the theoretical analysis, in principle these kernels with concentrated spectra can be approximated classically by truncating small eigenvalues; however, this is only possible when the entire spectral decomposition is known, which is not typically the case.

These numerical results confirm the conclusions of our theoretical analysis that using sparse (s-concentrated) unitaries provides an alternative way to obtain rapid decay of quantum kernels, distinct from bandwidth-tuning strategies explored in the literature [18,28]. Moreover, our analysis applies to more general models beyond the product-bandwidth kernels of the form $K(x.x') = \prod_{i=1}^{n} K_i(\lambda x_i, \lambda x_i')$ that were previously analyzed.

2. Generalizability comparison on synthetic dataset

Given the "no free lunch" theorem of machine learning (ML), we cannot expect a single kernel to perform well on all learning problems. In particular, all kernels have different strengths, and a kernel can only perform well when the target function has a large overlap with its dominant eigenspaces. In other words, even if the largest eigenvalue is relatively large, the kernel can still perform poorly on some datasets if the target function is not aligned with those eigenvectors. To isolate the effect of largest-eigenvalue scaling, we construct datasets *tailored* to each s-concentrated model and to the Haar model. First, let $P := 1 + n + n(n-1)/2$ (a number of eigenvalues that $\gtrsim 1/2^{n+2}$ for typical Haar random models). For a given unitary W_H, and W_s, we select the P eigenfunctions

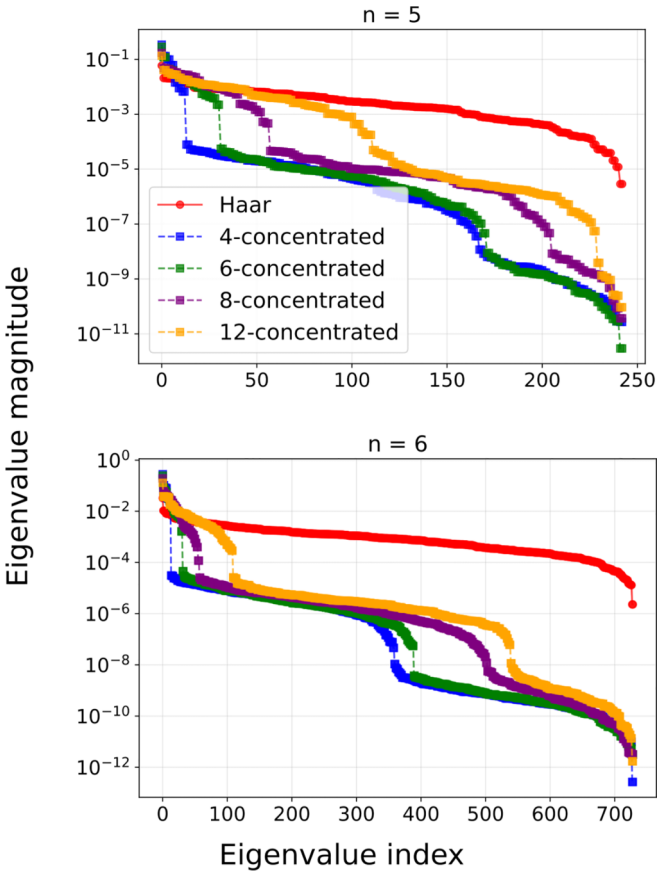


FIG. 4. Eigenvalue spectra of quantum kernels satisfying two assumptions in Sec. VI. These plots are from one instance among 30 experiments.

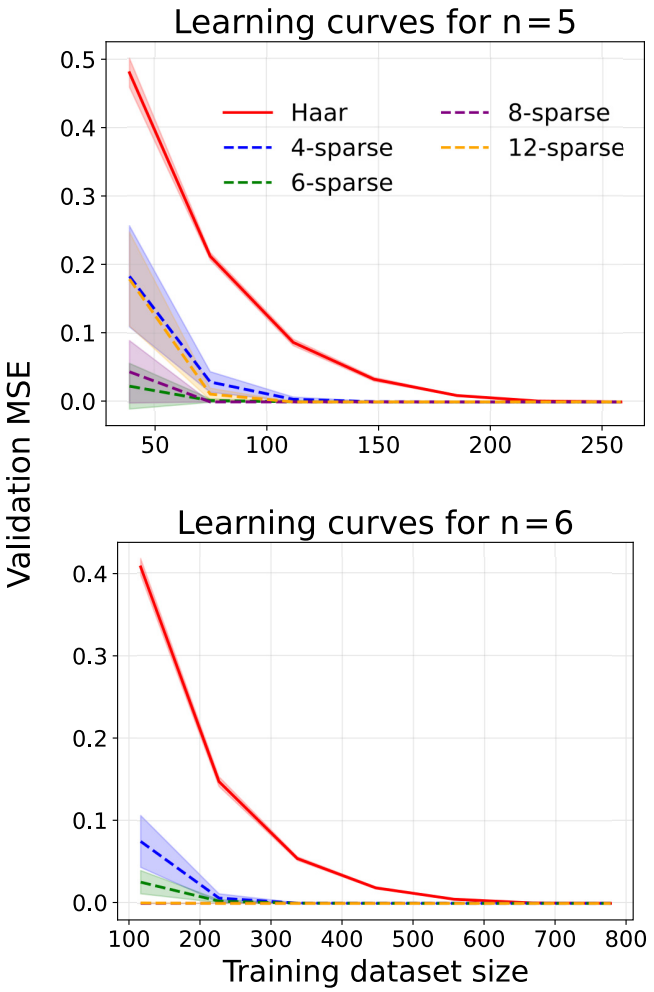


FIG. 5. Learning curves for kernel ridge regression tasks on the model-tailored datasets. Data points are averaged over 30 random models, and shaded regions stand for the standard deviation. Total number of training samples is 2×3^n, and the test set ratio was set to 0.2. Training was performed with the KernelRidgeRegression implementation from the scikit-learn Python package.

$\{\cos(\omega^\alpha \cdot x)\}_\alpha$ associated with the P largest eigenvalues of the kernel defined by the W operators. We then draw a random P-dimensional coefficient vector $c_{\alpha,W}$ whose entries have the same magnitude order as the model's, and construct the target function $f_{t,W}(x) = \sum_\alpha (\sqrt{c_{t,W}})_\alpha \cos(\omega^\alpha \cdot x)$. By using these model-specific target functions, the target-model alignment across experiments is comparable, so each kernel is effectively evaluated under near best-case alignment.

Figure 5 shows the average validation mean squared error (MSE) during training. The results show that the concentrated models reach a lower validation error more quickly than the Haar model, as predicted by our prior theoretical analysis. Here, we once more emphasize that each model—constructed from a different unitary—works on target functions that were built to lie in that model's top P-dimensional eigenspace. Therefore, these experiments do not imply the general claim that concentrated models are always superior to Haar models. For an extreme example, when the target function lies in the subspace spanned by several of the lowest eigenvectors of a concentrated model, the Haar model could trivially outperform the concentrated model. However, what it does confirm is that when the necessary criteria of kernel-target alignment are satisfied, kernels with more concentrated spectra generalize more rapidly than kernels with uniform spectra. We also discussed potential sources for notably bad performance of four-sparse model at early stage in Appendix E.

Taken together, these numerical experiments confirm the design principles suggested by our theoretical analysis. Specifically, when using kernels satisfying Assumptions 1 and 2 in the previous section, in order to generalize well, it is necessary (but not sufficient) to use kernels with a concentrated pre-encoding state.

VII. SUMMARY AND DISCUSSIONS

In this work, we have introduced the concept of an ETK, a generalization of product kernels. These ETKs allow one to construct kernels from a set of existing feature maps by using a *core matrix* to define an inner product on the tensor product of feature spaces of the original feature maps. This core matrix can be understood as creating an entangled feature map. Moreover, we have discussed the complexity of evaluating ETKs, showing that in some cases they can be efficiently evaluated via tensor networks.

Having defined ETKs, we then showed that all quantum kernels are ETKs, whose local feature maps are defined by the data-encoding strategy of the quantum circuit. This fact then allowed us to obtain a variety of insights into quantum kernels by examining them through the lens of their ETK representations.

Our first insight in this regard is that quantum kernels can provide a mechanism for the efficient evaluation of "superpolynomial bond-dimension" ETKs. These are ETKs whose core tensor does not admit a representation as an MPO with polynomial bond dimension and, therefore, cannot be straightforwardly evaluated classically in an efficient way. In this sense, one can think of quantum kernels that generate superpolynomial bond-dimension ETKs as possessing an *inductive bias* that is not straightforwardly accessible to efficient classical kernels with the same feature maps. This notion, therefore, provides a different lens through which to understand the potential advantages and applications of quantum kernels.

Complementing the above insights into the potential advantages of quantum kernels that generate superpolynomial bond-dimension ETKs, we have also provided some analysis of whether there exist quantum kernels that are both superpolynomial bond-dimension ETKs, and have the capacity to generalize well for specific problems from polynomially sized datasets. Indeed, in order to generalize efficiently one requires that the Reproducing Kernel Hilbert Space (RKHS) of the quantum kernel be effectively of only polynomially large dimension [18,28], which seems potentially incompatible with inefficiency of classical evaluation. We have shown that while a polynomially large effective RKHS is required, *obtaining* the description of the relevant subspace of the RKHS can be computationally hard, and even if one knows the description, evaluating the eigenfunctions of the kernel can also be computationally hard. Thus, we conjecture that there exist quantum kernels that both lead to superpolynomial bond-dimension ETKs *and* generalize efficiently for some class of problems.

Additionally, the fact that all quantum kernels admit an ETK representation naturally suggests the use of ETKs as *classical surrogates* for the dequantization of quantum kernel methods, and we discuss the potential and limitations of this approach. In particular, we provide a variety of concrete directions for future work in this direction.

Finally, to further showcase the power of the ETK lens for the analysis of quantum kernels, we have used the ETK framework for the analysis of a class of concrete quantum kernels, which goes beyond those analyzed in previous works [14,17,18]. In particular, we have used the ETK framework to obtain the Mercer decomposition for the relevant class of quantum kernels, from which one can gain insights into the spectrum of the kernel. This then itself informs design choices that effect the ability of the kernel to generalize well from only polynomially sized datasets.

Taken together, we see that ETKs provide a lens for the analysis of quantum kernels, which helps us to understand both the potential advantages and limitations of quantum kernel methods. It is our hope that the ETK framework can be of use in the further analysis and design of both classical and quantum kernels.

ACKNOWLEDGMENTS

The authors would like to thank Elies Gil-Fuster and Jens Eisert for helpful conversations and constructive feedback. R.S. is grateful for support from the Alexander von Humboldt Foundation, under the German Research Chair program. This research was supported by the education and training program of the Quantum Information Research Support Center, funded through the National Research Foundation of Korea (NRF) by the Ministry of Science and ICT (MSIT) of the Korean government (Grants No. RS-2023-NR057243, No. RS-2024-00413957, No. RS-2024-00438415, No. RS-2023-NR076733, No. RS-2025-25464492, and No. RS-2024-00437191), and by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (RS-2025-25464252, IITP-2025-RS-2020-II201606 and IITP-2025-RS-2024-00437191).

DATA AVAILABILITY

The data that support the findings of this article are not publicly available upon publication because it is not technically feasible and/or the cost of preparing, depositing, and hosting the data would be prohibitive within the terms of this research project. The data are available from the authors upon reasonable request.

APPENDIX A: EXAMPLES OF ENTANGLED TENSOR KERNELS

We supplement the definition of entangled tensor kernels in Sec. III B with a variety of concrete examples.

1. Polynomial kernel

The polynomial kernel $K_{(N,c)}$ is defined via

$$K_{(N,c)}(\mathbf{x}, \mathbf{x}') = (c + \mathbf{x}^\top \mathbf{x}')^N. \tag{A1}$$

One can immediately see that

$$K_{(N,c)}(\mathbf{x}, \mathbf{x}') = \prod^N K_c(\mathbf{x}, \mathbf{x}'), \tag{A2}$$

where

$$K_c(\mathbf{x}, \mathbf{x}') = c + \mathbf{x}^\top \mathbf{x}'. \tag{A3}$$

As such, the polynomial kernel $K_{(N,c)}$ is a product kernel. Note that from the feature map perspective, we can write

$$K_{(N,c)}(\mathbf{x}, \mathbf{x}') = \langle F(\mathbf{x}) | F(\mathbf{x}') \rangle, \tag{A4}$$

where

$$|F(\mathbf{x})\rangle = \bigotimes^N |F_c(\mathbf{x})\rangle, \tag{A5}$$

and

$$|F_c(\mathbf{x})\rangle = [\sqrt{c},\ \mathbf{x}_1,\ \mathbf{x}_2,\ \ldots,\ \mathbf{x}_d]. \tag{A6}$$

2. Linear sum kernels

It is a well-known fact that any linear sum of kernels with positive coefficients is also a valid kernel; i.e.,

$$K_{(N,\mathbf{a})}(\mathbf{x},\mathbf{x}') := \sum_{i=1}^{N} \mathbf{a}_i K_i(\mathbf{x},\mathbf{x}') \tag{A7}$$

is a valid kernel.

To represent this kernel as an ETK, we construct a product feature map

$$|F(\mathbf{x})\rangle = \bigotimes_{i=1}^{N} \begin{pmatrix} 1 \\ F_i(\mathbf{x}) \end{pmatrix}. \tag{A8}$$

Additionally, let us denote the dimensions of local feature maps as d_i. For notational simplicity, we define the multidimensional index $h := h_1 h_2, \ldots, h_N$, where $h_i \in \{0, 1, \ldots, d_i - 1\}$. Then,

$$K_{(N,\mathbf{a})}(\mathbf{x},\mathbf{x}') = \sum_{i=1}^{N} \mathbf{a}_i \langle F_i(\mathbf{x}) | F_i(\mathbf{x}')\rangle \tag{A9}$$

$$= \langle F(\mathbf{x}) | C | F(\mathbf{x}')\rangle, \tag{A10}$$

where the core tensor is diagonal with

$$C_{h,h} = \begin{cases} a_i, & \text{if} \quad h_i \neq 0 \quad \text{and} \quad h_{k\neq i} = 0, \\ 0, & \text{else.} \end{cases} \tag{A11}$$

One can note that linear sum kernels are a subclass of ETK where we only use first-order feature maps among all possible correlated feature maps such as $|F_i(\mathbf{x})\rangle|F_j(\mathbf{x})\rangle$, $|F_i(\mathbf{x})\rangle|F_j(\mathbf{x})\rangle|F_k(\mathbf{x})\rangle$, and so on.

3. Shift-invariant kernels

A shift-invariant kernel is one satisfying

$$K(\mathbf{x},\mathbf{x}') = K(\mathbf{x} - \mathbf{x}') \tag{A12}$$

for some positive semidefinite function $K : \mathcal{X} \mapsto \mathbb{R}$. If we consider $\mathcal{X} = [-\pi, \pi]$, then any such shift-invariant kernel can be written as

$$K(x - x') = \sum_{j=0}^{\infty} \gamma_j \cos(j(x - x')), \tag{A13}$$

with $\gamma_j \geqslant 0$ [31]. Here, we show that all shift invariant kernels on $\mathcal{X}$ with a *finite frequency cutoff*—i.e., with $\gamma_j = 0$ for all $j > N$—can be written as entangled tensor kernels, with only $\tilde{N} := \lceil \log_3(2N + 1) \rceil$ local feature maps. To this end, we start by defining the local feature maps,

$$|F^{(k)}(x)\rangle = \begin{pmatrix} e^{-i3^{k-1}x} \\ 1 \\ e^{i3^{k-1}x} \end{pmatrix} \tag{A14}$$

for $k \in [\tilde{N}]$. As per Sec. III B, we then define

$$|F(x)\rangle = \bigotimes_{k} |F^{(k)}(x)\rangle \in \mathbb{C}^{3^{\tilde{N}}}. \tag{A15}$$

The components of $|F(x)\rangle$ can be indexed with trit strings $h \in \{0, 1, 2\}^{\tilde{N}}$, and one can verify that

$$F_h(x) = \exp\left(\left[\sum_{k}^{\tilde{N}} 3^{k-1}(h_k - 1)\right] ix\right) \tag{A16}$$

$$= \exp(i\alpha_h x), \tag{A17}$$

where we have defined

$$\alpha_h = \sum_{k}^{\tilde{N}} 3^{k-1}(h_k - 1). \tag{A18}$$

We note that

$$\{\alpha_h \,|\, h \in \{0, 1, 2\}^{\tilde{N}}\} = \left\{ -\left(\frac{3^{\tilde{N}} - 1}{2}\right), \ldots, -1, 0, 1, \ldots, \left(\frac{3^{\tilde{N}} - 1}{2}\right)\right\}, \tag{A19}$$

and that $\alpha_{\overline{h}} = -\alpha_h$, where $\overline{h}$ is defined to be the trit string obtained by exchanging all occurrences of 0 with 2 and vice versa, while leaving 1s unchanged. Using this, let C be the $3^{\tilde{N}} \times 3^{\tilde{N}}$ diagonal matrix defined via

$$C_{h,h} = \begin{cases} 0, & \text{if} \quad |\alpha_h| > N, \\ \gamma_j, & \text{if} \quad |\alpha_h| = j \leqslant N. \end{cases} \tag{A20}$$

We then have

$$\langle F(x) | C | F(x')\rangle = \sum_{h\in\{0,1,2,3\}^{\tilde{N}}} C_{h,h} e^{i\alpha_h(x-x')} \tag{A21}$$

$$= \gamma_0 + \sum_{j=1}^{N} \gamma_j (e^{ij(x-x')} + e^{-ij(x-x')}) \tag{A22}$$

$$= \sum_{j=0}^{N} \gamma_j \cos(j(x - x')) \tag{A23}$$

$$= K(x - x'). \tag{A24}$$

As such, K is indeed an entangled tensor kernel, with local feature maps $\{F^{(k)} \,|\, k \in \tilde{N}\}$ defined via Eq. (A14), and core tensor C defined via Eq. (A20).

As per the discussion in Sec. III C, naively evaluating K requires N operations. However, if the core tensor C admits a representation as an MPO with $O(\log N)$ bond dimension, then the ETK representation can be used to evaluate $K(x, x')$ using only $O(\log N)$ operations. Additionally, we note that the ETK representation presented here is *not* unique. For instance, one could use $[e^{-ix}\ 1\ e^{ix}]^\top$ as local feature maps, but this choice would require N local feature maps. We use this observation to highlight the nonuniqueness of the ETK representation of a given kernel and the strong dependence of computational/memory complexity on the representation chosen.

APPENDIX B: OBTAINING MERCER DECOMPOSITIONS OF ENTANGLED TENSOR KERNELS

As discussed in Sec. II C 1, one can extract a variety of quantitative insights into the inductive bias of a kernel from its *Mercer decomposition*. In light of this, we provide here an algorithm for obtaining the Mercer decompositions of ETKs. We stress from the outset, however, that the complexity of this algorithm scales with the product of the sizes of the local feature spaces used in the ETK and is therefore often not efficient in practice.

To start, let us assume that we are given an ETK

$$K_C(\mathbf{x}, \mathbf{x}') = \langle F(\mathbf{x})|C|F(\mathbf{x}')\rangle, \tag{B1}$$

with

$$|F(\mathbf{x})\rangle = \bigotimes_k |F^{(k)}(\mathbf{x})\rangle. \tag{B2}$$

We assume $|F^{(k)}(\mathbf{x})\rangle \in \mathbb{C}^{d_k}$, so that $|F(\mathbf{x})\rangle \in \mathbb{C}^d$, where $d = \prod_k d_k$. As before, we denote the components of $F : \mathcal{X} \to \mathbb{C}^d$ with $F_i : \mathcal{X} \to \mathbb{C}$, for all $i \in [d]$. We note that all component functions are elements of the vector space of square integrable functions $L^2(\mathcal{X}, P_{\mathcal{X}})$.

A first simple case is the one in which C is diagonal and all components of F are orthonormal. By comparison with Eq. (9), we see that this kernel is already in Mercer form. However, in general the core tensor C will not be diagonal, and the component functions $\{F_i\}$ will not be orthonormal. In this case, we proceed via the following steps.

Step 1: Obtain an orthonormal basis from component functions. We start by defining

$$\tilde{d} = \dim[\mathrm{span}(\{F_i \,|\, i \in [d]\})]. \tag{B3}$$

Additionally, without loss of generality we assume an ordering such that $\{F_i \,|\, i \in [\tilde{d}]\}$ are linearly independent (one simply reorders rows of C appropriately). By using the Gram-Schmidt orthonormalization procedure, we can construct an orthonormal basis $\{e_i \,|\, i \in [\tilde{d}]\}$ for $\mathrm{span}(\{F_i \,|\, i \in [d]\})$ by first constructing

$$u_i = F_i - \sum_{j=1}^{i-1} \frac{\langle u_j, F_i\rangle}{\langle u_j, u_j\rangle} u_j, \tag{B4}$$

and then defining $e_i = u_i/\|u_i\|_2$, where $\|u_i\|_2 = \sqrt{\langle u_i, u_i\rangle}$. If we then define

$$|\tilde{e}(\mathbf{x})\rangle = (e_1(\mathbf{x}), \ldots, e_{\tilde{d}}(\mathbf{x}))^\top, \tag{B5}$$

one has that

$$|\tilde{e}(\mathbf{x})\rangle = \tilde{L}|F(\mathbf{x})\rangle, \tag{B6}$$

where $\tilde{L}$ is a $\tilde{d} \times d$ lower-triangular matrix with

$$\tilde{L}_{i,i} = \frac{1}{\|u_i\|_2},$$
$$\tilde{L}_{i,i-j} = -\sum_{l=0}^{j-1} \tilde{L}_{i,i-l} \frac{\langle u_{i-j}, F_{j-l}\rangle}{\langle u_{i-j}, u_{i-j}\rangle} \quad (1 \leqslant j \leqslant i-1). \tag{B7}$$

Step 2: Pad the orthonormal basis. Now, we note that by assumption on the dimension of $\mathrm{span}(\{F_i \,|\, i \in [d]\})$ and the ordering of $\{F_i\}$, for all $i > \tilde{d}$ one has that

$$F_i = \sum_{j=1}^{\tilde{d}} \alpha_{i,j} F_j, \tag{B8}$$

for some $\{\alpha_{i,j} \in \mathbb{C} \,|\, j \in [\tilde{d}]\}$. Using this, we can construct a $d \times d$ lower triangular matrix L by adding rows to $\tilde{L}$, via

$$L_{i,j} = \begin{cases} \tilde{L}_{i,j}, & \text{if} \quad i \leqslant \tilde{d}, \\ -\alpha_{i,j}, & \text{if} \quad i > \tilde{d},\ j \leqslant \tilde{d}, \\ 1, & \text{if} \quad i = j > \tilde{d}. \end{cases} \tag{B9}$$

This matrix is lower triangular, with nonzero entries on all diagonals, and is therefore invertible (which, as we will see in Step 3, was the motivation for the padding). Additionally, we have that

$$L|F(\mathbf{x})\rangle = \begin{pmatrix} e_1(\mathbf{x}) \\ \vdots \\ e_{\tilde{d}}(\mathbf{x}) \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \tag{B10}$$
$$:= |e(\mathbf{x})\rangle. \tag{B11}$$

Step 3: Transform and truncate C. Using the above, we have that

$$K_C(\mathbf{x}, \mathbf{x}) = \langle F(\mathbf{x})|C|F(\mathbf{x})\rangle \tag{B12}$$
$$= \langle F(\mathbf{x})|L^\top (L^\top)^{-1} C L^{-1} L|F(\mathbf{x})\rangle \tag{B13}$$
$$= \langle e(\mathbf{x})|(L^\top)^{-1} C L^{-1}|e(\mathbf{x})\rangle \tag{B14}$$
$$= \langle e(\mathbf{x})|\hat{C}|e(\mathbf{x})\rangle \tag{B15}$$
$$= \langle \tilde{e}(\mathbf{x})|\tilde{C}|\tilde{e}(\mathbf{x})\rangle, \tag{B16}$$

where $\hat{C} = (L^\top)^{-1} C L^{-1}$ and $\tilde{C}$ is the upper left $\tilde{d} \times \tilde{d}$ block of $\hat{C}$.

Step 4: Diagonalize $\tilde{C}$. The expression in Eq. (B16) is almost in Mercer form. The final step is to diagonalize $\tilde{C}$. More specifically, to find U such that $\tilde{C} = U^\dagger D U$. Given this, we then have that

$$K_C(\mathbf{x}, \mathbf{x}) = \langle \tilde{e}(\mathbf{x})|\tilde{C}|\tilde{e}(\mathbf{x})\rangle \tag{B17}$$
$$= \langle \tilde{e}(\mathbf{x})|U^\dagger D U|\tilde{e}(\mathbf{x})\rangle \tag{B18}$$
$$= \langle \hat{e}(\mathbf{x})|D|\hat{e}(\mathbf{x})\rangle, \tag{B19}$$

where $|\hat{e}(\mathbf{x})\rangle := U|\tilde{e}(\mathbf{x})\rangle$. As the components of $|\tilde{e}(\mathbf{x})\rangle$ are orthonormal, so too are the components of $|\hat{e}(\mathbf{x})\rangle$. As a result, Eq. (B19) is finally the desired Mercer decomposition of K_C.

APPENDIX C: A STANDARD FORM FOR DATA-DEPENDENT QUANTUM CIRCUITS

In this Appendix, we show how any data-dependent unitary can be transformed into a data-dependent quantum circuit of a "standard form." To make this precise, we formalize a data-dependent unitary as a map $U : \mathcal{X} \to \mathrm{SU}(2^n)$. With this convention, $U(\mathbf{x}) \in \mathrm{SU}(2^n)$ represents the specific unitary corresponding to $\mathbf{x} \in \mathcal{X}$. At this stage, we make no assumptions on the form of this map. We now show that for any such

map U, there exists a parametrized quantum circuit $C : \mathcal{X} \to \mathrm{SU}(2^n)$, such that $U(\mathbf{x}) = C(\mathbf{x})$ for all $x \in \mathcal{X}$, and C is in the *standard form*

$$C(\cdot) = \prod_{j=1}^{L} S_j(\cdot) W_j, \tag{C1}$$

where for all $j \in L$, $W_j \in \mathrm{SU}(2^n)$ is a fixed unitary, and $S_j : \mathcal{X} \to \mathrm{SU}(2^n)$ is a data-dependent unitary of the form

$$S_j(\cdot) := \left(\bigotimes_{k=1}^{n} e^{-i\phi_{jk}(\cdot)Z_k/2} \right), \tag{C2}$$

$$:= \left(\bigotimes_{k=1}^{n} R_Z(\phi_{jk}(\cdot)) \right), \tag{C3}$$

where $\phi_{jk} : \mathcal{X} \to [0, 2\pi)$ are functions that depend on U.

To show this, we consider decomposing U into a sequence of Givens rotations, as per Ref. [35]. If one does this, in the case where U is data dependent, one will obtain

$$U(\cdot) = \prod_{j=1}^{\tilde{L}} G_j(\cdot), \tag{C4}$$

where for all j, the parametrized Givens rotation $G_j : \mathcal{X} \to \mathrm{SU}(2^n)$ is a parametrized rotation in a fixed (non-data-dependent) two-dimensional subspace of the 2^n dimensional Hilbert space. Given this, we then note the following:

(1) Any such rotation can be written as a multicontrolled parametrized single-qubit rotation, which in turn can be decomposed into a circuit consisting only of cnot gates and parametrized single-qubit gates $R : \mathcal{X} \to \mathrm{SU}(2)$. We stress that as the subspace in which the rotation occurs is fixed, the order and position of the cnot gates and single-qubit gates are also fixed—i.e., they depend only on j, and not on $\mathbf{x}$.

(2) Each parametrized single-qubit rotation can be written as

$$R(\cdot) = R_Z(\phi_1(\cdot)) R_Y(\phi_2(\cdot)) R_Z(\phi_3(\cdot)), \tag{C5}$$

where $\phi_i : \mathcal{X} \to [0, 2\pi)$ are functions that depend on U.

(3) Each R_Y gate can be decomposed into R_Z gates and basis rotations using the relation $Y = F^\dagger Z F$, where $F := \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ i & -i \end{pmatrix}$.

If one does all of the above, and then groups together all data-dependent gates and data-independent gates into separate layers, one finally arrives at a circuit in the form of Eq. (C1).

At this stage, we stress that in the worst case, the decomposition above will lead to circuits in which

(1) L is exponentially large in the number of qubits;

(2) the functions $\{\phi_{jk}\}$ are extremely complicated.

However, in practice, one normally starts not from some arbitrarily defined data-dependent unitary $U : \mathcal{X} \to \mathrm{SU}(2^n)$, but rather from some parametrized quantum circuit $C : \mathcal{X} \to \mathrm{SU}(2^n)$ with a polynomial number of gates, of which the data-dependent gates are all of some fixed simple form (e.g., gates that encode data by parametrizing the time evolution of a simple few-body Hamiltonian [9,41]). In many such cases, the decomposition of the circuit into the standard form of Eq. (C1) can be done much more efficiently than via the Givens rotation construction given above, which is simply used to demonstrate the universality of the standard form.

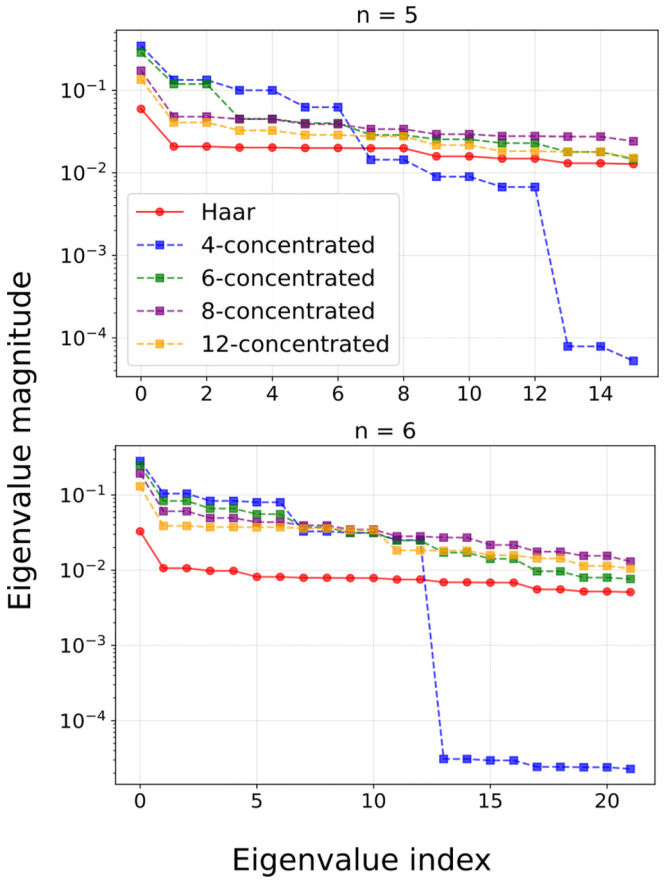


FIG. 6. Eigenspectra of the largest P eigenvalues of kernel models, where P is the number of support of target function.

APPENDIX D: POSITIVE SEMIDEFINITENESS AND HERMITICITY OF C_T

To ensure a tensor network representation of a quantum kernel is ETK, we need to check that the core matrix that comes from the quantum circuit contraction is positive semidefinite and Hermitian (symmetric).

First, we observe that the operator O' is a tensor product of identity and Choi matrices of unitaries that are applied in the circuit, so it is PSD and Hermitian. The operator $\rho^\top$ is also PSD as it is a tensor product of (un-normalized) density matrices. By the Schur product theorem, $O' \odot \rho^\top = A$ is also PSD. Transpose preserves positivity and eigenvalues of the vertical tensor product of two matrices are given by multiplication between the eigenvalues of two matrices. Also, Schur product of two Hermitian (symmetric) matrices is Hermitian (symmetric), and from Eq. (57), we see that the vertical tensor product of Hermitian matrix and its transpose preserves the Hermiticity. Therefore, C is PSD and Hermitian.

Next, we show that C_T is PSD and Hermitian as well. For all $|v\rangle \in \mathbb{C}^{3^N}$, and PSD C,

$$\begin{aligned}\tfrac{1}{4^N}\langle v|C_T|v\rangle &= 2^N\langle\tilde{v}|\Pi_3\Pi_3 P_U^\dagger C P_U \Pi_3\Pi_3|\tilde{v}\rangle \\ &= 2^N\langle\tilde{v}|\Pi_3 C'\Pi_3|\tilde{v}\rangle \geqslant 0, \end{aligned} \tag{D1}$$

where $\Pi_3 := \begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&0\end{pmatrix}^{\otimes N}$ is a projection and $P_U = \frac{1}{\sqrt{2}}\begin{pmatrix}1&0&0&1\\0&1&i&0\\0&1&-i&0\\1&0&0&-1\end{pmatrix}^{\otimes N}$, which is a unitary extension of P and $|\tilde{v}\rangle$ is an arbitrary extension of $|v\rangle$ to $\mathbb{C}^{4^n}$, which embeds arbitrary elements to the extended dimension. Conjugation with unitary preserves PSD property, and $\Pi_3|\tilde{v}\rangle \in \mathbb{C}^{4^n}$, the last inequality follows. If H is Hermitian, then both $U^\dagger H U$ and $\Pi H \Pi$ are Hermitian whenever U is unitary and Π is a projection operator.

APPENDIX E: POSSIBLE SOURCES FOR BAD PERFORMANCE OF FOUR-SPARSE MODEL IN OUR NUMERICS

The performance of quantum kernels depends not only on the scaling of the largest eigenvalue but also on kernel-target alignment, which quantifies how well the function space spanned by the eigenfunctions with the largest eigenvalues covers the target function. In Fig. 6, we plot the largest $P = 1 + n + n(n-1)/2$ eigenvalues for the models from one of our experiments, where P is the number of basis functions of the target function. One can see sharp drops in the eigenvalues for the four-sparse model due to its high sparsity. This possibly makes it difficult for the model to learn components associated with the low-eigenvalue basis functions.

[1] M. Schuld, *Supervised Learning with Quantum Computers* (Springer, Cham, 2018).

[2] V. Dunjko and H. J. Briegel, Machine learning & artificial intelligence in the quantum domain: A review of recent progress, Rep. Prog. Phys. **81**, 074001 (2018).

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

[4] M. Benedetti, E. Lloyd, S. Sack, and M. Fiorentini, Parameterized quantum circuits as machine learning models, Quantum Sci. Technol. **4**, 043001 (2019).

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

[6] I. Steinwart and A. Christmann, *Support Vector Machines* (Springer, New York, 2008).

[7] R. Mengoni and A. Di Pierro, Kernel methods in quantum machine learning, Quantum Mach. Intell. **1**, 65 (2019).

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

[9] 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 (London) **567**, 209 (2019).

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

[11] E. Gil-Fuster, J. Eisert, and V. Dunjko, On the expressivity of embedding quantum kernels, Mach. Learn.: Sci. Technol. **5**, 025003 (2024).

[12] B. Y. Gan, D. Leykam, and S. Thanasilp, A Unified framework for trace-induced quantum Kernels, arXiv:2311.13552.

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

[14] J. M. Kübler, S. Buchholz, and B. Schölkopf, The inductive bias of quantum kernels, arXiv:2106.03747.

[15] 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).

[16] S. Thanasilp, S. Wang, M. Cerezo, and Z. Holmes, Exponential concentration in quantum kernel methods, Nat. Commun. **15**, 5200 (2024).

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

[18] A. Canatar, E. Peters, C. Pehlevan, S. M. Wild, and R. Shaydulin, Trans. Mach. Learn. Res. (2023).

[19] P. Rodriguez-Grasa, Y. Ban, and M. Sanz, Neural quantum kernels: Training quantum kernels with quantum neural networks, Phys. Rev. Res. **7**, 023269 (2025).

[20] A. Cichocki, N. Lee, I. Oseledets, A.-H. Phan, Q. Zhao, and D. P. Mandic, Tensor networks for dimensionality reduction and large-scale optimization: Part 1 low-rank tensor decompositions, Found. Trends Mach. Learn. **9**, 249 (2016).

[21] R. Sengupta, S. Adhikary, I. Oseledets, and J. Biamonte, Tensor networks in machine learning, Eur. Math. Soc. Mag. **126**, 4 (2022).

[22] A. Anandkumar, R. Ge, D. Hsu, S. M. Kakade, and M. Telgarsky, Tensor decompositions for learning latent variable models, J. Mach. Learn. Res. **15**, 2773 (2014).

[23] I. Glasser, R. Sweke, N. Pancotti, J. Eisert, and J. I. Cirac, Expressive power of tensor-network factorizations for probabilistic modeling, with applications from hidden Markov models to quantum machine learning, Adv. Neural Info. Proc. Sys. **32**, 1498 (2019).

[24] R. Sweke, E. Recio, S. Jerbi, E. Gil-Fuster, B. Fuller, J. Eisert, and J. J. Meyer, Potential and limitations of random Fourier features for dequantizing quantum machine learning, Quantum **9**, 1640 (2023).

[25] U. Schollwöck, The density-matrix renormalization group in the age of matrix product states, Ann. Phys. **326**, 96 (2011).

[26] J. C. Bridgeman and C. T. Chubb, Hand-waving and interpretive dance: an introductory course on tensor networks, J. Phys. A: Math. Theor. **50**, 223001 (2017).

[27] N. Cristianini, J. Shawe-Taylor, A. Elisseeff, and J. S. Kandola, On kernel-target alignment, Adv. Neural Info. Proc. Sys. **14**, 367 (2001).

[28] A. Canatar, B. Bordelon, and C. Pehlevan, Spectral bias and task-model alignment explain generalization in kernel regression and infinitely wide neural networks, Nat. Commun. **12**, 2914 (2021).

[29] A. Shastry, A. Jayakumar, A. Patel, and C. Bhattacharyya, Shot-frugal and robust quantum kernel classifiers, arXiv:2210.06971.

[30] J. Schnabel and M. Roth, Quantum kernel methods under scrutiny: A benchmarking study, Quantum Mach. Intell. **7**, 58 (2025).

[31] M. J. Wainwright, Reproducing kernel Hilbert spaces, in *High-Dimensional Statistics: A Non-Asymptotic Viewpoint*, *Cambridge Series in Statistical and Probabilistic Mathematics* (Cambridge University Press, Cambridge, England, 2019), pp. 383–415.

[32] M. Kliesch, D. Gross, and J. Eisert, Matrix-product operators and states: NP-hardness and undecidability, Phys. Rev. Lett. **113**, 160503 (2014).

[33] A. Werner, D. Jaschke, P. Silvi, M. Kliesch, T. Calarco, J. Eisert, and S. Montangero, Positive tensor network approach for simulating open quantum many-body systems, Phys. Rev. Lett. **116**, 237201 (2016).

[34] G. D. l. Cuevas, N. Schuch, D. Pérez-García, and J. Ignacio Cirac, Purifications of multipartite states: Limitations and constructive methods, New J. Phys. **15**, 123021 (2013).

[35] J. J. Vartiainen, M. Möttönen, and M. M. Salomaa, Efficient decomposition of quantum gates, Phys. Rev. Lett. **92**, 177902 (2004).

[36] E. Tang, Dequantizing algorithms to understand quantum advantage in machine learning, Nat. Rev. Phys. **4**, 692 (2022).

[37] F. J. Schreiber, J. Eisert, and J. J. Meyer, Classical surrogates for quantum learning models, Phys. Rev. Lett. **131**, 100803 (2023).

[38] J. Landman, S. Thabet, C. Dalyac, H. Mhiri, and E. Kashefi, Classically approximating variational quantum machine learning with random Fourier features, *The Eleventh International Conference on Learning Representations* (ICLR, 2023).

[39] S. P. Jordan and P. Wocjan, Efficient quantum circuits for arbitrary sparse unitaries, Phys. Rev. A **80**, 062301 (2009).

[40] S. Aaronson and Y. Zhang, On verifiable quantum advantage with peaked circuit sampling, arXiv:2404.14493.

[41] 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).