MACHINE LEARNING
Science and Technology

PAPER

On the expressivity of embedding quantum kernels

Elies Gil-Fuster[1,2,*] (ID), Jens Eisert[1,2,3] (ID) and Vedran Dunjko[4,5] (ID)

[1] Dahlem Center for Complex Quantum Systems, Freie Universität Berlin, 14195 Berlin, Germany
[2] Fraunhofer Heinrich Hertz Institute, 10587 Berlin, Germany
[3] Helmholtz-Zentrum Berlin für Materialien und Energie, 14109 Berlin, Germany
[4] $\langle aQa^L \rangle$ Applied Quantum Algorithms, Universiteit Leiden, Leiden, The Netherlands
[5] LIACS, Universiteit Leiden, Niels Bohrweg 1, 2333 CA Leiden, The Netherlands
[*] Author to whom any correspondence should be addressed.

E-mail: emgilfuster@gmail.com

Abstract

One of the most natural connections between quantum and classical machine learning has been established in the context of kernel methods. Kernel methods rely on kernels, which are inner products of feature vectors living in large feature spaces. Quantum kernels are typically evaluated by explicitly constructing quantum feature states and then taking their inner product, here called embedding quantum kernels. Since classical kernels are usually evaluated without using the feature vectors explicitly, we wonder how expressive embedding quantum kernels are. In this work, we raise the fundamental question: can all quantum kernels be expressed as the inner product of quantum feature states? Our first result is positive: Invoking computational universality, we find that for any kernel function there always exists a corresponding quantum feature map and an embedding quantum kernel. The more operational reading of the question is concerned with efficient constructions, however. In a second part, we formalize the question of universality of efficient embedding quantum kernels. For shift-invariant kernels, we use the technique of random Fourier features to show that they are universal within the broad class of all kernels which allow a variant of efficient Fourier sampling. We then extend this result to a new class of so-called composition kernels, which we show also contains projected quantum kernels introduced in recent works. After proving the universality of embedding quantum kernels for both shift-invariant and composition kernels, we identify the directions towards new, more exotic, and unexplored quantum kernel families, for which it still remains open whether they correspond to efficient embedding quantum kernels.

1. Introduction

Quantum devices carry the promise of surpassing classical computers in certain computational tasks [1–6]. With machine learning playing a crucial role in predictive tasks based on training data, the question arises naturally to investigate to what extent quantum computers may assist in tackling *machine learning* (ML) tasks. Indeed, such tasks are among the potential applications foreseen for near-term and intermediate-term quantum devices [7–12].

In the evolving field of *quantum machine learning* (QML), researchers have explored the integration of quantum devices to enhance learning algorithms [13–19]. The most-studied approach to QML relies on learning models based on *parametrized quantum circuits* (PQCs) [20], sometimes referred to as quantum neural networks. When considering learning tasks with classical input data, PQCs must embed data into quantum states. This way, PQCs are built from encoding and trainable parts, and real-valued outputs are extracted from measuring certain observables. Since the inception of the field, a strong parallelism has been drawn between PQC-based QML models and *kernel methods* [14, 15, 21].
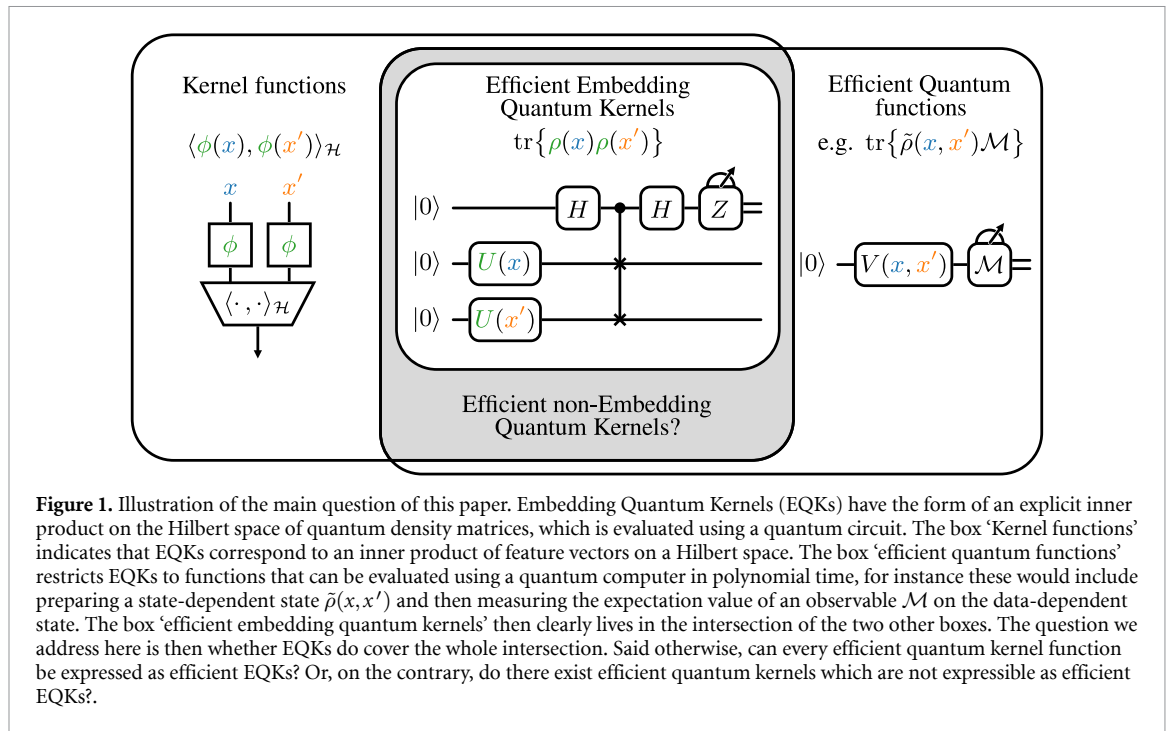
Figure 1. Illustration of the main question of this paper. Embedding Quantum Kernels (EQKs) have the form of an explicit inner product on the Hilbert space of quantum density matrices, which is evaluated using a quantum circuit. The box 'Kernel functions' indicates that EQKs correspond to an inner product of feature vectors on a Hilbert space. The box 'efficient quantum functions' restricts EQKs to functions that can be evaluated using a quantum computer in polynomial time, for instance these would include preparing a state-dependent state $\tilde{\rho}(x, x')$ and then measuring the expectation value of an observable $\mathcal{M}$ on the data-dependent state. The box 'efficient embedding quantum kernels' then clearly lives in the intersection of the two other boxes. The question we address here is then whether EQKs do cover the whole intersection. Said otherwise, can every efficient quantum kernel function be expressed as efficient EQKs? Or, on the contrary, do there exist efficient quantum kernels which are not expressible as efficient EQKs?.

Kernel methods, like neural networks, have been used in ML for solving complex learning tasks. Yet, unlike neural networks, kernel methods reach the solution by solving a linear optimization task on a larger feature space, onto which input data is mapped. Consequently, the kernel approach is very well suited by our ability to map classical data onto the Hilbert space of quantum states. These maps are called *quantum feature maps*, and they lead to *quantum kernel methods*. Although kernel methods are more costly to implement than neural networks, they are guaranteed to produce optimal solutions. Much the same way, with quantum kernel methods, we are guaranteed to find better solutions than with other PQC-based models (where 'better' means solutions which perform better on the training set; see [22] for a discussion on when this guarantee is not enough to ensure a learning advantage).

For quantum kernel methods, plenty of knowledge is inherited from classical ML—including kernel selection tools [19, 23], optimal solution guarantees [21, 22], generalization bounds [24], and approximation protocols [25–27]. Nevertheless, there is one large difference between quantum and classical kernel methods, and namely one that affects the cornerstone of these techniques: the *kernel function* (or just *kernel*). Formally, all kernels correspond to the inner product of a pair of feature maps. Yet, first constructing the feature vector and second evaluating the inner product is often inefficient. Fortunately, many cases are known in which the inner product can be evaluated efficiently, by means other than constructing the feature map explicitly. For instance the Gaussian kernel is a prominent instance of this case. This is sometimes called 'the kernel trick', and as a result it is often the case that practitioners do not even specify the feature vectors when using kernel methods. In contrast to this, it is fair to say that quantum kernels hardly ever use this trick, with some exceptions [28, 29].

Almost all quantum kernels conceived in the literature are constructed explicitly from a quantum feature map, or *quantum embedding* [21, 30], as discussed below. Specifically, one considers as quantum kernel κ the inner product $\kappa(x, x') := \langle \psi(x), \psi(x') \rangle$, where $x \mapsto \psi(x)$ is a representation of a quantum state, either a state vector or a density operator, and $\langle \cdot, \cdot \rangle$ is the appropriate inner product. In particular, quantum embeddings map classical data onto the Hilbert space of quantum states, or said otherwise, on the Hilbert space of quantum computations. We call *embedding quantum kernels* (EQKs) the kernels which come from quantum embeddings.

This difference between quantum and classical kernels raises some interesting questions, as for example: *Are EQKs the whole story for quantum kernel methods? Can all quantum kernels be expressed as EQKs?*

In this manuscript we analyze what families of kernels are already covered by EQKs, see figure 1. Our contributions are the following:

1. We show that all kernels can be seen as EQKs, thus proving their universality, when no considerations on efficiency are made.

2. We formalize the question of expressivity of *efficient* EQKs. We immediately provide a partial answer restricted to *shift-invariant kernels*. We show that efficient EQKs are universal within the class of shift-invariant kernels, and we provide sufficient conditions for an EQK approximation to be produced efficiently in time.

3. We introduce a new class of kernels, called composition kernels, containing also non-shift-invariant kernels. We prove that efficient EQKs are universal in the class of efficient composition kernels, from where we can show that the *projected quantum kernel* from [28] can in fact also be realized as an EQK efficiently.

In all, we unveil the universality of EQKs in two important function domains.

The rest of this work is organized as follows. The mathematical background and relevant definitions appear in section 2. Related and prior work is elucidated in section 3. Next, we prove the universality of embedding quantum kernels and formally state question 1 in section 4. Our results on shift-invariant kernels are in section 5, and the extension to composition kernels and the projected quantum kernel in section 6. Finally, section 7 contains a collection of questions left open. A summary of the manuscript and closing remarks constitute section 8.

2. Preliminaries

In this section we fix notation and introduce the necessary bits of mathematics on quantum kernel methods.

2.1. Notation

For a vector $v = (v_i)_i \in \mathbb{R}^m$, we call the 1-norm (or ℓ_1-norm) $\|v\|_1 = \sum_{i=1}^m |v_i|$ and the 2-norm $\|v\|_2 = \sqrt{\sum_{i=1}^m v_i^2}$. We denote as ℓ_1^m the set of m-dimensional unit vectors with respect to the 1-norm, and similarly ℓ_2^m the set of vectors that are normalized to have a unit 2-norm.

For a Hilbert space $\mathcal{H}$, we use $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ to denote the inner product on that space. In the case of Euclidean spaces, we drop the subscript. Let $A, B \in \mathbb{C}^{m \times m}$ be square complex matrices. Then, we call the *Hilbert–Schmidt inner product* of A and B

$$\langle A, B \rangle_{\mathrm{HS}} = \mathrm{tr}\left\{ A^\dagger B \right\} = \sum_{i,j=1}^m A_{i,j}^* B_{i,j}. \tag{1}$$

For Hermitian matrices, we have $A = A^\dagger$, and so the HS inner product becomes just

$$\langle A, B \rangle_{\mathrm{HS}} = \mathrm{tr}\left\{ AB \right\} = \sum_{i,j=1}^m A_{j,i} B_{i,j}. \tag{2}$$

The *Frobenius norm* $\|\cdot\|_F$ of a matrix A is the square root of the sum of the magnitude square of all its entries, which is also equal to the root of the Hilbert–Schmidt inner product of the matrix with itself, as

$$\|A\|_F^2 = \langle A, A \rangle_{\mathrm{HS}} = \sum_{i,j} |A_{i,j}|^2. \tag{3}$$

In what follows, we call $\mathcal{X} \subseteq \mathbb{R}^d$ a d-dimensional compact subset of the reals. We reserve n to denote qubit numbers. As we explain below, we use k to refer to arbitrary kernel functions, while κ is used exclusively for embedding quantum kernels.

When talking about efficient and inefficient approximations, we consider sequences of functions $\{k_s\}_{s \in \mathbb{N}}$. We refer to s as the *scale parameter*. Scale parameters can correspond to different qualities of the sequence, as for example the dimension of the input data, or the number of qubits involved in evaluating a function. Efficiency then means at most polynomial scaling in s, and inefficiency means at least exponential scaling in s, hence the name scale parameter.

2.2. Kernel methods

Kernel methods solve ML tasks as linear optimization problems on large feature spaces, sometimes implicitly. The connection between the input data and the feature space comes from the use of a kernel function. In this work we do not busy ourselves with how the solution is found, but rather we focus on our ability to evaluate kernel functions, which are defined as follows.

Definition 1 (Kernel function). A kernel function $k\colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a map from pairs of inputs on the reals fulfilling two properties:

1. Symmetry under exchange: $k(x,x') = k(x',x)$ for every $x,x' \in \mathcal{X}$.
2. Positive semi-definite (PSD): for any integer $m \in \mathbb{N}$, for any sets $\{x_1,\ldots,x_m\} \subseteq \mathcal{X}$ and $\{a_1,\ldots,a_m\} \subseteq \mathbb{R}$ of size m, it holds

$$\sum_{i,j=1}^{m} a_i a_j k\left(x_i, x_j\right) \geqslant 0. \tag{4}$$

This is equivalent to saying that the *Gram matrix* $K := [k(x_i,x_j)]_{i,j=1}^{m}$ is positive semi-definite for any m and any $\{x_i\}_{i=1}^{m}$.

Other standard definitions exclude the PSD property. In this work we do not study indefinite, non-PSD kernels, although the topic is certainly of interest. The common optimization algorithms used in kernel methods (SVM, KRR) require kernels to be PSD. Even though we said we do not deal with the optimization part in this manuscript, we study only the kernels that would be used in a (Q)ML context.

Symmetry and PSD are properties usually linked to inner products, which partly justifies our definition of a Gram matrix as the one built from evaluating the kernel function on pairs of inputs. Indeed, by Mercer's Theorem (detailed in appendix A) for any kernel function k, there exists a Hilbert space $\mathcal{H}$ and a *feature map* $\phi\colon \mathcal{X} \to \mathcal{H}$ such that, for every pair of inputs $x,x' \in \mathcal{X}$, it holds that evaluating the kernel is equivalent to computing the $\mathcal{H}$-inner product of pairs of *feature vectors* $k(x,x') = \langle \phi(x), \phi(x')\rangle_{\mathcal{H}}$. We say every kernel k has an associated feature map ϕ, which turns each datum x into a feature vector $\phi(x)$, living in a *feature space* $\mathcal{H}$.

One notable remark is that different kernel functions have different feature maps and feature spaces associated to them. Each learning task comes with a specific data distribution. The ultimate goal is to, given the data distribution, find a map onto a feature space where the problem becomes solvable by a linear model. The *model selection* challenge for kernel methods is to find a kernel function whose associated feature map and feature space fulfill this linear separation condition. Intuitively, in a classification task, we would want data from the same class to be mapped to the same corner of Hilbert space, and data from different classes to be mapped far away from one another. Then our project can be framed within the *quantum kernel selection* problem, we want to identify previously unexplored classes of quantum kernels.

2.3. Embedding quantum kernels

In the following, we introduce the relevant concepts in quantum kernel methods. Our presentation largely follows the lines of [21]. We use the name 'embedding quantum kernels' following [19]; many other works use simply 'quantum kernels' when referring to the same concept. Hopefully our motives are clear by now.

Learning models based on PQCs are the workhorses in much of today's approaches to QML. With some notable exceptions, these PQC-based models comprise a two-step process:

1. Prepare a data-dependent quantum state. For every $x \in \mathcal{X}$, produce $|\phi(x)\rangle = U(x)|0\rangle$ with some data-embedding unitary $U(x)$.
2. Measure the expected value of a variationally-tunable observable on the data-dependent state. Given a parametrized observable $\mathcal{M}(\vartheta)$, evaluate
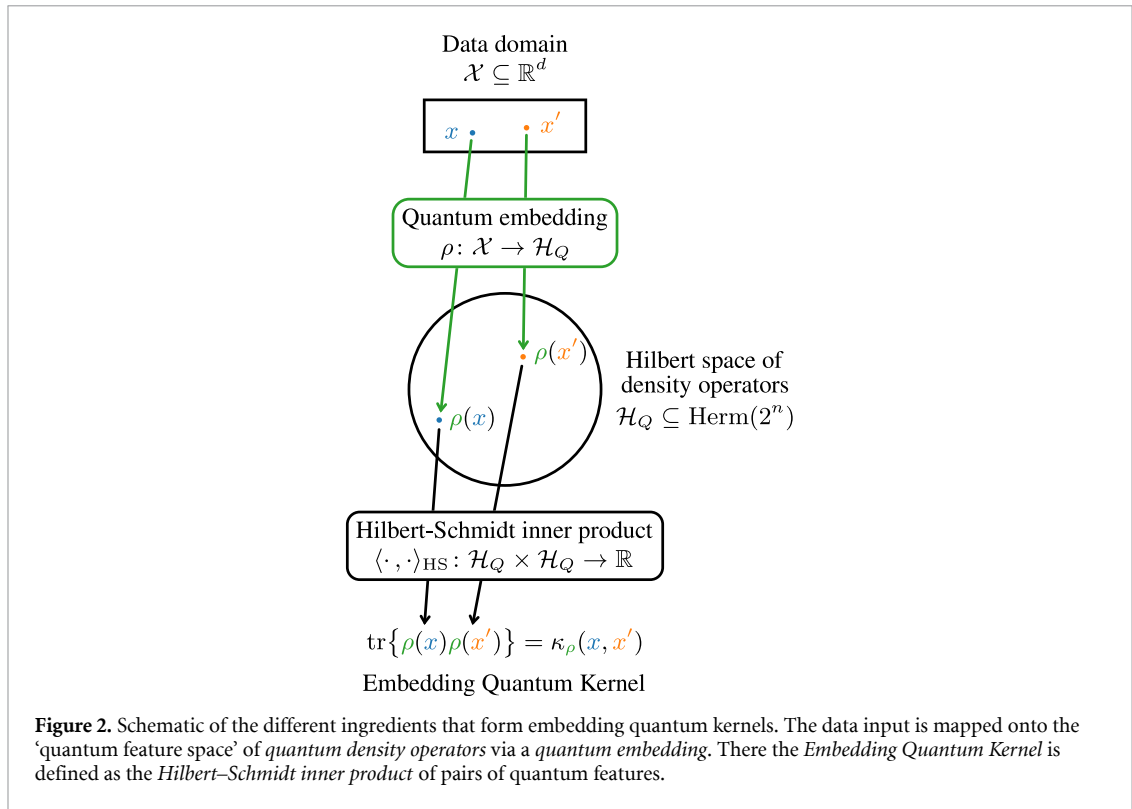
$$\langle \mathcal{M}(\vartheta)\rangle_{\phi(x)} = \operatorname{tr}\left\{|\phi(x)\rangle\langle\phi(x)|\mathcal{M}(\vartheta)\right\}. \tag{5}$$

For instance, for binary classification, one would then consider labeling functions like $h(x) = \operatorname{sign}\left(\langle\mathcal{M}(\vartheta)\rangle_{\phi(x)} + b\right)$ and optimize the variational parameters ϑ and b.

Examples of how $U(x)$ could look like in practice include *amplitude encoding* [21], different *data re-uploading* encoding strategies [17, 31, 32], or the *IQP ansatz* [14]. In turn, variationally-tunable observables are usually realized as a fixed 'easy' observable (like a single Pauli operator), preceded by a few layers of brickwork-like 2-local trainable gates. In our current presentation, we do not restrict a particular form for $U(x)$ or $\mathcal{M}(\vartheta)$, but rather allow for any general form possible, as long as it can be implemented on a quantum computer.

This approach can also be understood through the lens of kernel methods. In this case, the feature space is explicitly chosen to be the Hilbert space of Hermitian matrices (of which density matrices are a subset, so also quantum states). Given a data-dependent state preparation $x \mapsto |\phi(x)\rangle$, one could choose to call it a *quantum feature map* ρ and promote it to quantum density operators, or quantum states, as

$$\rho\colon \mathcal{X} \to \operatorname{Herm}, \tag{6}$$

Figure 2. Schematic of the different ingredients that form embedding quantum kernels. The data input is mapped onto the 'quantum feature space' of *quantum density operators* via a *quantum embedding*. There the *Embedding Quantum Kernel* is defined as the *Hilbert–Schmidt inner product* of pairs of quantum features.

$$x \mapsto \rho(x) = |\phi(x)\rangle\langle\phi(x)|. \tag{7}$$

Another fitting name would be *quantum feature state*. In a slightly more general view, we call *data embedding* any map from classical data onto quantum density matrices of fixed dimension $\rho\colon \mathcal{X} \to \mathrm{Herm}(2^n)$ for n-qubit systems. With this we abandon the need for a unitary gate applied to the $|0\rangle$ state vector.

Together with the Hilbert–Schmidt inner product, quantum feature maps give rise to an important family of kernel functions:

Definition 2 (Embedding quantum kernel (EQK)). Given a data embedding $x \mapsto \rho(x)$ used to encode classical data x in a quantum density operator $\rho(x)$, we call *embedding quantum kernel (EQK)* κ_ρ the Hilbert–Schmidt inner product of pairs of quantum feature vectors:

$$\kappa_\rho\colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}, \tag{8}$$

$$(x, x') \mapsto \kappa_\rho(x, x') := \mathrm{tr}\{\rho(x)\rho(x')\}. \tag{9}$$

Figure 2 illustrates this construction.

Since κ_ρ is defined explicitly as an inner product for any ρ, it follows that it is a PSD, symmetric function. For ease of notation, we will write just κ whenever ρ is unimportant or clear from context. There are, from the outset, a few good reasons to consider EQKs as QML models:

1. It is possible to construct EQK families that are not classically estimable unless $\mathsf{BQP} = \mathsf{BPP}$, thus opening the door to quantum advantages [33].
2. A core necessity for successful kernel methods is to map complex data to high-dimensional spaces where feature vectors become linearly separable. The Hilbert space of quantum states thus becomes a prime candidate due to its exponential dimension, together with our ability to estimate inner products efficiently[6].
3. Since EQKs are explicitly defined from the data embeddings, we are free to design embeddings with specific desired properties.

---

[6] However, high dimensions are related to overfitting, so as usual there is a trade-off.

And, *a priori*, other than the shot noise[7], EQKs do not add drawbacks to the list of general issues for kernel methods. So, EQKs are a well-founded family of quantum kernel functions. Nevertheless, their reliance on a specific data embedding could be a limiting factor *a priori*.

In the case of classical computations we are not accustomed to think in terms of the Hilbert space of the computation (even though it does exist). There are numerous examples of kernels which have been designed without focusing on the feature map. The most prominent example of a kernel function used in ML is the *radial basis function (RBF) Gaussian kernel* (or just Gaussian kernel, for short), given by

$$k_\sigma (x, x') = \exp \left(-\frac{\|x - x'\|^2}{2\sigma^2} \right). \tag{10}$$

On the one hand, we know the Gaussian kernel is a PSD function by virtue of Bochner's theorem (stated in section 5 as theorem 4), which explains it via the *Fourier transform* (FT). On the other hand, though, one can prove that the Gaussian kernel corresponds to an inner product in the Hilbert space of all monomials of the components of *x*, from where it follows that the Gaussian kernel can learn any continuous function on a compact domain [34], provided enough data is given. The feature map corresponding to the Gaussian kernel is infinite-dimensional. This fact alone makes the Gaussian kernel not immediately identifiable as a reasonable EQK, which motivates the question: *can all quantum kernels be expressed as embedding quantum kernels?*

In this section, we have introduced kernel functions, embedding quantum kernels, and the question whether EQKs are all there is to quantum kernels. In the following sections, we first see that EQKs are universal, then we formalize a question about the expressivity of efficient EQKs, and finally we answer the question by proving universality of EQKs in two restricted kernel families.

3. Related work

An introduction to quantum kernel methods can be found in the note [21] and in the review [30]. We refrain from including here a full compendium of all quantum kernel works to date, as good references can already be found in [12]. Instead, we make an informed selection of papers that bear relation with the object of our study: which quantum kernels are *embedding quantum kernels* (EQKs).

Kernel methods use different optimization algorithms depending on each task, the most prominent examples being *support vector machine* and *kernel ridge regression*. The early years of QML owed much of their activity to the HHL algorithm [35]. The quantum speed-up for linear algebra tasks was leveraged to propose a *quantum support vector machine* (qSVM) algorithm [36]. The qSVM algorithm is listed among the first steps of QML historically. Since qSVM is a quantum application for kernel methods, it is no wonder that the term quantum kernel methods was introduced for concepts around it. This, however, is not what we mean by quantum kernel methods in this work. Instead, we occupy ourselves with kernel methods where the kernel function itself requires a quantum computer to be evaluated, independently of the nature of the optimization algorithm. The optimization step comes only after the kernel function has been evaluated on the training data, so the object of study of qSVM is not the same as in this manuscript. We study the expressivity of a known kind of kernel functions, and not how to speed up the optimization of otherwise classical algorithms.

Among the first references mentioning the evaluation of kernel functions with quantum computers is [13], where the differences between quantum kernels and qSVM have been made explicit. [14, 15] have showcased the link between quantum kernels and PQCs and have demonstrated their implementation experimentally. In them, the authors have mentioned the parallelisms between quantum feature maps and the kernel trick. In particular, [15] has coined the distinction between an *implicit quantum model* (quantum kernel method), and an *explicit quantum model* (PQC model with gradient-based optimization). Implicit models are in this way an analogous name for EQKs, where emphasis is made on the distinction from other PQC-based models.

Quantum kernels have enjoyed increasing attention since they have been used to prove an advantage of QML over classical ML in [33]. The authors have morphed the discrete logarithm problem into a learning task and then used a quantum kernel to solve it efficiently, which cannot be done classically according to well-established cryptographic assumptions. In that, the approach taken is similar to that of [37] (showing a quantum advantage in distribution learning). As such, this has been among the first demonstrations of quantum advantage in ML, albeit in an artificially constructed learning task. Importantly, the quantum kernel used in this work has explicitly been constructed from a quantum embedding, so it is an EQK in the sense of this work.

One important difference between EQKs and other PQC-based approaches is that, for EQKs, the only design choice is the data embedding itself. Lloyd *et al* [18] has wondered how to construct optimal quantum

---

[7] It is known that the finite approximation to the expectation values can break down PSD.

feature maps using measurement theory. The work has proposed constructing embeddings specific to learning tasks, which resonates with the idea that some feature maps are better than others for practically relevant problems. Hubregtsen *et al* [19], where the term EQKs has been introduced for the first time, has presented the possibility of optimizing the feature map variationally, drawing bridges to ideas from *data re-uploading* [17, 31]. Other than the trainable kernels of [19], the data re-uploading framework did not fit with the established quantum kernel picture up until this point in time.

In [22], the differences between explicit models, implicit models, and re-uploading models have been analyzed. On the one hand, the authors have found that the optimality of kernel methods might not be optimal enough, since they show a learning task in which kernels perform much worse than explicit models when evaluated on data outside the training set. On the other hand, a rewriting algorithm has been devised to convert re-uploading circuits into equivalent encoding-first circuits, so re-uploading models are explicit models. By construction, each explicit model has a quantum kernel associated to it. So, for the first time, using the rewriting via explicit models, [22] made data re-uploading models fit in the quantum kernel framework. This reference takes the explicit versus implicit distinction from [15], which means that it also only considers EQKs when it comes to kernels.

Some examples of different kinds of quantum kernels can be found in [28, 29], which upon first glance do not resemble the previous proposals. In [28] a type of kernel functions based on the classical shadows protocol has been proposed, under the name of *projected quantum kernel*. These functions have been analyzed in the context of a quantum–classical learning separation, now considering real-world data, as opposed to the results of [33]. Even though the projected quantum kernel uses an explicit feature map which requires a quantum computer, the feature vectors are only polynomial in size, and they are stored in classical memory. Next, the kernel function is the Gaussian kernel evaluated on pairs of feature vectors, and not their Euclidean or Hilbert–Schmidt inner product. These two differences set the projected quantum kernel apart from the rest. In turn, the authors of [29] set out to address a looming issue for EQKs called *exponential kernel concentration* [38], or *vanishing similarity*, which is tantamount to the *barren plateau problem* [39] that could arise also for kernels. They show that the new construction, called 'anti-symmetric logarithmic derivative quantum Fisher kernel', does not suffer from the vanishing similarity problem. Here, the classical input data is mapped onto an exponentially large feature space. Given a PQC, classical inputs are mapped to a long array, with as many entries as trainable parameters in the PQC. Each entry in this array is the product of the unitary matrix implemented by the PQC and its derivatives with respect to each variational parameter. Interestingly, this kernel can be seen as the Euclidean inner product of a flattened vector of unitaries, with a metric induced by the initial quantum state. In this way, classical data is not mapped onto the Hilbert space of quantum states, but the inner product used is the same as in regular EQKs. This realization paves the way to expressing these kernels as EQKs.

Recent efforts in de-quantizing PQC-based models via *classical surrogates* [40] have touched upon quantum kernels. In [25–27], the authors propose using a classical kernel-approximation protocol based on *random Fourier features* (RFF) to furnish classical learning models capable of approximating the performance of PQC-based architectures. The techniques used in these works are not only very promising for de-quantization of PQC-based learning models, but also they are relevant to our discussion. Below, we also use the RFF approach, albeit in quite a different way, as it is not our goal to find classical approximations of quantum functions, but rather quantum approximations of other quantum functions. The goal of the study in [25–27] is to, given a PQC-based model, construct a classical kernel model with guarantees that they are similarly powerful. In this way, the input to the algorithms is a PQC (either encoding-first, or data re-uploading), and the output is a classical kernel. Conversely, in our algorithms, the input is a kernel function, and the output is an EQK-based approximation of the same function.

In [41], we find an earlier study of RFFs in a QML scenario. The authors focus on a more advanced version of RFFs, called *optimized Fourier Features*, which involves sampling from a data-dependent distribution. In the classical literature, it was found that sampling from this distribution could be 'hard in practice' [42], so the authors of [41] set out to propose an efficient quantum algorithm for sampling. This way, a quantum algorithm is proposed to speed up a training algorithm for a classical learning architecture, similar to the case of qSVM [36].

Similarly, [43] considered combining notions from EQKs, the projected quantum kernel, and their respective approximations with the RFF algorithm. The authors explored the options of combining distance-based and inner-product-based kernels in an attempt to address the vanishing similarity problem. In turn, this paper tackles the suboptimal scaling of kernel methods by making use of RFFs. Some of the techniques introduced in this reference are very much in line with the composition kernels we introduce in section 6. The main similarity between our work and [43] is the identification of feature maps for different quantum kernels, including the use of RFFs and the study of the projected quantum kernel. The main difference is the perspective in which these objects are studied. In [43] the authors look for kernel

constructions that have advantageous properties when it comes to solving learning tasks. Conversely, in this work we want to establish what are the ultimate limits in expressivity of using kernel functions based on quantum feature maps.

4. The universality of quantum feature maps

In the previous section we saw how we can define quantum kernel functions explicitly from a given data embedding. This section, together with the following two sections, contains the main results of this manuscript. The statements shift around different notions of efficiency and different classes of kernel functions. In figure 3 we provide a small sketch of how each of our results relates to one another from a zoomed-out perspective. The leitmotiv is that we can always restrict further what restrictions we are satisfied with when concocting EQK-based approximations of kernels. We find all kernels can be approximated as EQKs if our only restriction is to use finitely many quantum resources. But then, we specialize the search for kernels which can be approximated efficiently as EQKs, with a distinction between space-efficiency (number of qubits required) and total run-time efficiency. It should also be said that when we talk about time efficiency we always consider 'quantum time', so we assume we are always allowed access to a quantum computer. This way, the analysis from now on departs from the usual quantum-classic separation mindset. In figure 4 we show how all our results follow the same recipe, in that our constructions always involve constructing a feature map and then prepare the corresponding quantum state.

From the outset, two basic results combined, namely Mercer's feature space construction (elaborated in appendix A), together with the universality of quantum circuits, already certify the possibility of all kernel functions to be realized as EQKs. If we demanded mathematical equality, Mercer's construction could require infinite-dimensional Hilbert spaces, and thus quantum computers with *infinitely many qubits*. Instead, for practical purposes, from now on we do not talk about 'evaluating' functions in one or another way, but rather about 'approximating' them to some precision. With this, theorem 1 confirms that we can always approximate any kernel function as an EQK to arbitrary precision *with finitely many resources*.

For this universality statement, we allow for extra multiplicative and additive factors. Instead of talking about ε-approximation as $|k(x,x') - \mathrm{tr}\{\rho_n(x)\rho_n(x')\}| < \varepsilon$, we consider $|k(x,x') - 2^n \mathrm{tr}\{\rho_n(x)\rho_n(x')\} + 1| < \varepsilon$. These extra factors come from lemma 3, introduced right below, and they do not represent an obstacle against universality.

Theorem 1 (Approximate universality of finite-dimensional quantum feature maps). *Let $k\colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a kernel function. Then, for any $\varepsilon > 0$ there exists $n \in \mathbb{N}$ and a data embedding ρ_n onto the Hilbert space of quantum states of n qubits such that*

$$|k(x,x') - 2^n \mathrm{tr}\{\rho_n(x)\rho_n(x')\} + 1| < \varepsilon \tag{11}$$

for almost all $x, x' \in \mathcal{X}$.

The statement that equation (11) holds for *almost all $x, x' \in \mathcal{X}$* comes from measure theory, and it is a synonym of 'except in sets of measure 0', or equivalently 'with probability 1'. That means that although there might exist individual adversarial instances of $x, x' \in \mathcal{X}$ for which the inequality does not hold, these 'bad' instances are sparse enough that the event of drawing them from the relevant probability distribution has associated probability 0.

Theorem 1 says every kernel function can be approximated as an EQK up to a multiplicative and an additive factor using finitely-many qubits. Before we give the theorem proof, we introduce the useful algorithm 1 to map classical vectors to quantum states that we can then use to evaluate Euclidean inner products as EQKs. Then, lemma 2 contains the correctness statement and runtime complexity of algorithm 1, and lemma 3 shows the relation between the Euclidean inner product of the encoded real vectors and the Hilbert–Schmidt inner product of the encoding quantum states.

Algorithm 1. Classical to quantum embedding, `C2QE`.

Input: A 1-norm unit vector $r \in \ell_1^d$.
Output: A quantum state $\rho_r \propto \mathbb{I} + \sum_{i=1}^d r_i P_i$.   ▷ See lemma 2.
  1: Set $n = \lceil \log_4(d+1) \rceil$.
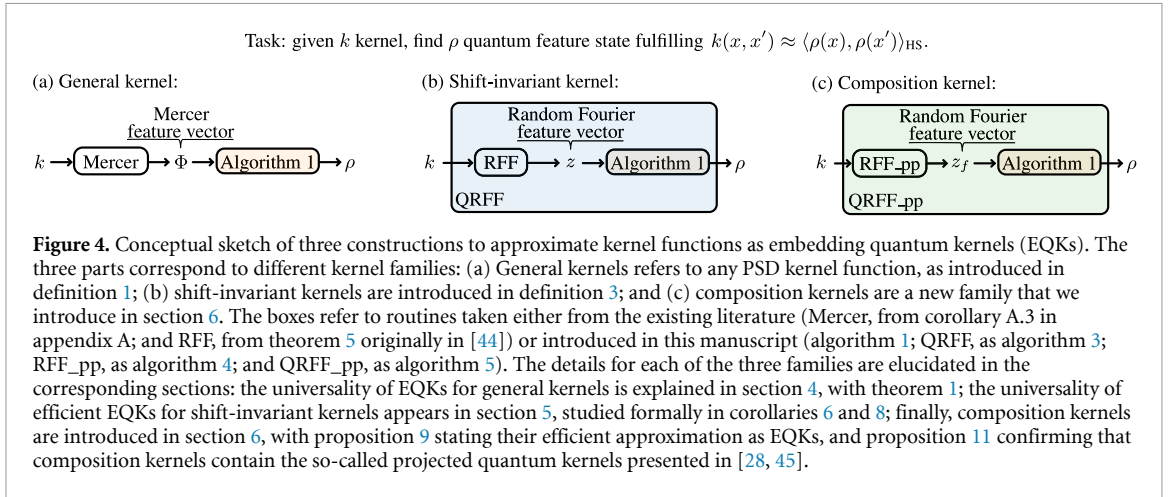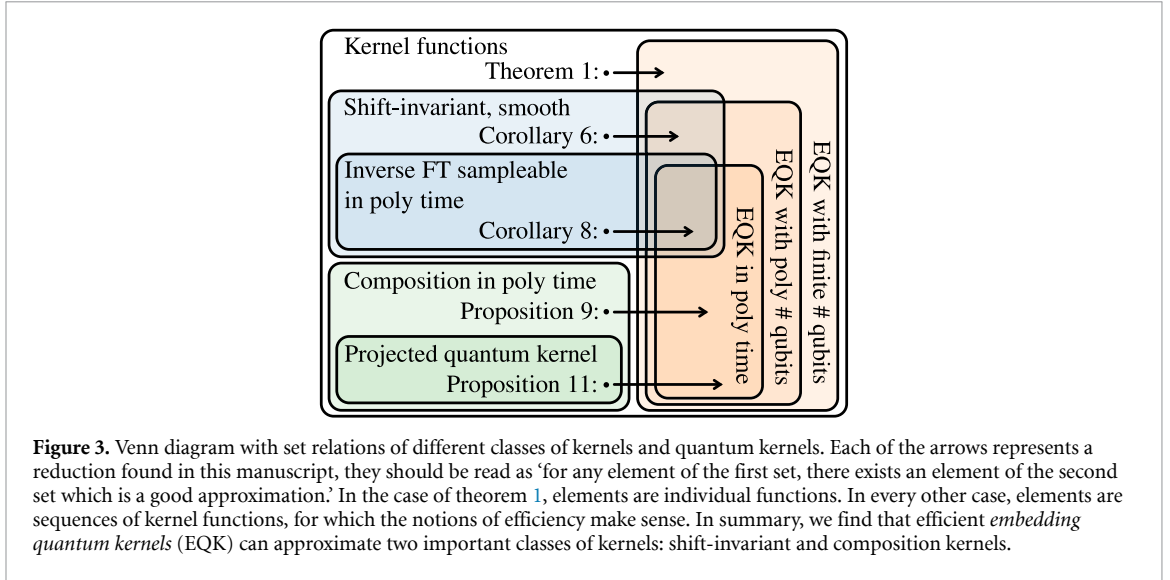 2: Pad r with 0s until it has length $4^n - 1$.
  3: Draw $i \in \{1, \ldots, 4^n - 1\}$ with probability $|r_i|$.
  4: Prepare $\rho_i = \frac{1}{2^n}(\mathbb{I} + \mathrm{sign}(r_i)P_i)$.
  5: **return** ρ_i.

Figure 3. Venn diagram with set relations of different classes of kernels and quantum kernels. Each of the arrows represents a reduction found in this manuscript, they should be read as 'for any element of the first set, there exists an element of the second set which is a good approximation.' In the case of theorem 1, elements are individual functions. In every other case, elements are sequences of kernel functions, for which the notions of efficiency make sense. In summary, we find that efficient *embedding quantum kernels* (EQK) can approximate two important classes of kernels: shift-invariant and composition kernels.

Figure 4. Conceptual sketch of three constructions to approximate kernel functions as embedding quantum kernels (EQKs). The three parts correspond to different kernel families: (a) General kernels refers to any PSD kernel function, as introduced in definition 1; (b) shift-invariant kernels are introduced in definition 3; and (c) composition kernels are a new family that we introduce in section 6. The boxes refer to routines taken either from the existing literature (Mercer, from corollary A.3 in appendix A; and RFF, from theorem 5 originally in [44]) or introduced in this manuscript (algorithm 1; QRFF, as algorithm 3; RFF_pp, as algorithm 4; and QRFF_pp, as algorithm 5). The details for each of the three families are elucidated in the corresponding sections: the universality of EQKs for general kernels is explained in section 4, with theorem 1; the universality of efficient EQKs for shift-invariant kernels appears in section 5, studied formally in corollaries 6 and 8; finally, composition kernels are introduced in section 6, with proposition 9 stating their efficient approximation as EQKs, and proposition 11 confirming that composition kernels contain the so-called projected quantum kernels presented in [28, 45].

Notice that the output of algorithm 1, $\frac{1}{2^n}(\mathbb{I} \pm P)$, is a single (pure) eigenstate of a Pauli operator P with eigenvalue ± 1. Nevertheless, as line 3 involves drawing an individual index $i \in \{1, \ldots, 4^n - 1\}$, we see algorithm 1 as a random algorithm, which prepares a mixed state as a classical mixture of pure states.

Lemma 2 (Correctness and runtime of algorithm 1). *Let $r \in \ell_1^d \subseteq \mathbb{R}^d$ be a unit vector with respect to the 1-norm, $\|r\|_1 = 1$. Take $n = \lceil \log_4(d+1) \rceil$ and pad r with 0s until it has length $4^n - 1$. Let $(P_i)_{i=1}^{4^n-1}$ be the set of all Pauli matrices on n qubits without the identity. Then algorithm 1 prepares the following state as a classical mixture*

$$\rho_{(\cdot)} \colon \ell_1^d \to \mathrm{Herm}\,(2^n), \tag{12}$$

$$r \mapsto \rho_r = \frac{\mathbb{I} + \sum_{i=1}^{4^n-1} r_i P_i}{2^n}. \tag{13}$$

The total runtime complexity t of algorithm 1 fulfills $t \in \mathcal{O}(\mathrm{poly}(d))$.

Lemma 3 (Euclidean inner products). *Let $r, r' \in \mathbb{R}^d$ be unit vectors with respect to the 1-norm $\|r^{(\prime)}\|_1 = 1$. Then, for $\rho_r, \rho_{r'}$ as produced in algorithm 1, the following identity holds*

$$\langle r, r' \rangle = 2^n \,\mathrm{tr}\,\{\rho_r \rho_{r'}\} - 1. \tag{14}$$

Proof of lemmas 2 and 3. The proofs are presented in appendix B.   □

Remark (Prefactors). The 2^n multiplicative factor is of no concern, since $n \in \mathcal{O}(\log(d))$ and we are interested in methods that are allowed to scale polynomially in d. In general ρ_r will be a mixed quantum state which can be efficiently prepared. Also, the map is injective, but not surjective.

We are now in the position to prove theorem 1. We note that the first step in the proof is to furbish a 'classical' feature map, and in a second step we turn the resulting feature vector into a quantum state. This way, theorem 1 makes no pretense at quantum advantage, but rather establishes the ultimate expressivity of embedding quantum kernels. We also point out that not imposing space efficiency breaks the division between quantum and classical computing. Every quantum circuit running in finite time can be fully simulated by a classical circuit also running in finite time. This way, the universality statement coming from theorem 1 is not unique to quantum kernels, since formally at this point there is no distinction between classical and quantum functions.

Proof of theorem 1. We prove this statement directly using a corollary of Mercer's theorem and the universality of quantum computing. First, we invoke corollary A.3 in appendix A, which ensures the existence of a finite-dimensional feature map $\Phi_m : \mathcal{X} \to \mathbb{R}^m$ for which it holds that

$$|k(x,x') - \langle \Phi_m(x), \Phi_m(x') \rangle| < \varepsilon. \tag{15}$$

Without loss of generality, we assume $\|\Phi_m(x)\|_1 = 1$ for all $x \in \mathcal{X}$. Now, we can prepare the quantum state $\rho_{\Phi_m(x)}$, which will use $\lceil \log_4(m+1) \rceil$ many qubits. By preparing two such states, one for $\Phi_m(x)$ and one for $\Phi_m(x')$, we can compute their inner product as the Hilbert–Schmidt inner product of the quantum states as in lemma 3, to get

$$\langle \Phi_m(x), \Phi_m(x') \rangle = 2^n \operatorname{tr} \left\{ \rho_{\Phi_m(x)} \rho_{\Phi_m(x')} \right\} - 1. \tag{16}$$

For reference, notice $\operatorname{tr}\{\rho_{\Phi_m(x)} \rho_{\Phi_m(x')}\}$ could be computed using the SWAP test, to additive precision in the number of shots. With this, we can ensure good approximation to polynomial additive precision efficiently

$$|k(x,x') - 2^n \operatorname{tr} \left\{ \rho_{\Phi_m(x)} \rho_{\Phi_m(x')} \right\} + 1| < \varepsilon, \tag{17}$$

for almost every $x, x' \in \mathcal{X}$. This completes the proof. $\square$

Notice this is an conceptually motivated existence result only. Noteworthy is also that no claims are made about practicality neither in theorem 1 nor in algorithm 1. These results only aim at establishing the ultimate existence of EQKs for any kernel, and they are not claims that computing these kernels as EQKs should lead to any form of advantage over the inner product of the classical feature vectors.

The only statement is that there exists an EQK using a finite number of qubits, but it does not get into how quickly the number of qubits will grow for increasingly computational complicated kernel function k, or increased required precision $\varepsilon > 0$. The number of qubits n will depend on some properties of the kernel k and the approximation error ε, and if for example we had exponential scaling of the required number of qubits on some of these quantities, then theorem 1 would bring no practical application. Similarly, one should also consider the time it would take to find such an EQK approximation, independently of the memory and run-time requirements of preparing the feature vectors and computing their inner product.

Let us take the anti-symmetric logarithmic derivative quantum Fisher kernel of [29] as an example. Upon first inspection, evaluating that kernel does not seem to rely in the usual quantum feature map and Hilbert–Schmidt inner product combination. Yet, a feature map can be identified by rewriting some of the variables involved as a vector of exponential length. That means the scaling in the number of qubits required to encode the feature vector is at worst polynomial, according to lemma 3. The normalization requirement from lemma 3 could still prevent the feature map to be encoded using the construction we propose, but for the sake of illustration let us assume normalization is not a problem. Then, we have found a way of realizing the same kernel as an EQK. In this case, even though the scaling of the qubit number is no more than polynomial, the scaling in total run-time of the EQK approximation would still be at worst exponential.

The message remains that, although all kernel functions can be realized as EQKs, there could still exist kernel functions which cannot be realized as EQKs *efficiently*. Pointing back to figure 1, this explains why we added the word 'efficient' to the sets of quantum functions and EQKs. In order to talk about efficiency we need to replace individual functions k by function sequences $\{k_s\}_{s \in \mathbb{N}}$, where s is the scale parameter. When we refer to an efficient ε-approximation, we refer to an algorithm making use of up to $\operatorname{poly}(s, 1/\varepsilon)$ resources, for each k_s in the sequence. We now formally present the question we aim at answering.

Question 1 (Expressivity of efficient EQKs). *Let $\{k_s\}_{s \in \mathbb{N}}$ be a sequence of kernel functions, let $\varepsilon > 0$ be a precision parameter, and consider the properties:*

1. *Quantum efficiency: There is an algorithm that takes a specification of k_s as input and produces an ε-approximation of k_s with a quantum computer efficiently in s and $1/\varepsilon$.*

2. *Embedding-quantum efficiency: There is an algorithm that takes a specification of k_s as input and produces an ε-approximation of k_s as an EQK efficiently in s and $1/\varepsilon$.*
3. *Classical inefficiency: Any algorithm that takes a specification of k_s as input and produces an ε-approximation of k_s with a classical computer must be inefficient in either s or $1/\varepsilon$.*

Then, assuming $\{k_s\}_s$ fulfills classical inefficiency, does quantum efficiency imply embedding-quantum efficiency?

The question above contains a few moving pieces which still need to be made fully precise, as for instance: the meaning of the scaling parameter s, the sequence of domains $\mathcal{X}_s$ from where each k_s takes its input, any restrictions on the functions k_s, in what form must the functions k_s be specified, the choice of notion of ε-approximation, and the choice between space and time efficiency. These are left open on purpose to admit for diverse approaches to studying the question.

We could have required a stronger sense of inefficiency, namely that there cannot exist an efficient and uniform construction for ε-approximating k_s with classical computers. Instead, we judge it enough to require that, even if such an efficient approximation could exist, it would be impossible to find efficiently, conditional on $\mathsf{BQP} \nsubseteq \mathsf{P/poly}$. We added classical inefficiency because otherwise, in principle, one could have all: quantum efficiency, embedding-quantum efficiency, and classical efficiency, in which case the result would not be interesting for QML. An interesting question is then to imagine how classical kernel functions could also be embedding classical kernels, in the sense of data being mapped onto the Hilbert space of classical computation.

In later sections, we fix all of these in our answers to question 1. For instance, s for us is the dimension of the domains $\mathcal{X}_s$ from where data is taken, the kernel functions can be specified either as black-boxes or as the description of circuits, we will take infinity-norm approximation almost everywhere, and we alternate between qubit number efficiency and run-time efficiency.

As an aside, question 1 also invites research in the existence of quantum kernels beyond EQKs, so efficient quantum kernels which do not admit efficient EQK-based approximations. At the moment of writing, the authors are not aware of any concrete example of a quantum efficient kernel function which is provably not embedding-quantum efficient. Moreover, we do not know any quantum efficient kernel for which we do not have an explicit way of constructing the efficient quantum embedding. The outlook in section 7, and Appendices A and E contain several directions in which we expect candidate kernels beyond efficient EQKs to appear. It should be said that searching for quantum kernels beyond EQKs is slightly contradictory to the foundational philosophy of QML in the beginnings, which actively sought to express everything in terms of inner products in the Hilbert space of density operators. The foundational works [13–15] have motivated the use of embedding quantum kernels precisely because the inner product could be taken directly efficiently. Nevertheless, we point out the possibility of alternative constructions, keeping focus on methods which could still harbour quantum advantages.

5. The universality of efficient shift-invariant embedding quantum kernels

In this section we present our second result: All shift-invariant kernels admit space-efficient EQK approximation provided they are smooth enough. We also give sufficient conditions for a constructive time-efficient EQK approximation. We arrive at these results in two steps: We first prove an upper bound on the Hilbert space dimension required for an approximation as an explicit inner product, classically. We next construct and EQK based on this classical approximation.

Shift-invariant kernels have enjoyed significant attention in the ML literature. On the one hand, the Gaussian RBF kernel (arguably the most well-known shift-invariant kernel) has been found useful in a range of data-driven tasks. On the other hand, as we see in this section, shift-invariant kernels are more amenable to analytical study than other classes of kernels. The property of shift-invariance, combined with exchange symmetry and PSD, allows for deep mathematical characterization of functions.

Let us first introduce the class of functions of interest:

Definition 3 (Shift-invariant kernel function). A kernel function $k: \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is called *shift-invariant* if, for every $x, x', \xi \in \mathcal{X}$, it holds that

$$k(x + \xi, x' + \xi) = k(x, x'). \tag{18}$$

As is standard, we then write shift-invariant kernels as a function of a single argument $k(x - x')$ (which amounts to taking $\xi = -x'$.) We define $\Delta := x - x'$ and then talk about $k(\Delta)$.

One motivation for using shift-invariant kernels, as a more restricted function family, is that they have a few useful properties that ease their analysis. Also, it is difficult to decide whether an arbitrary function is

PSD, so characterizing general kernel functions is difficult. Conversely, for shift-invariant functions, Bochner's theorem gives a condition that is equivalent with being PSD.

Theorem 4 (Bochner [46]). *Let k be a continuous, even, shift-invariant function on $\mathbb{R}^d$. Then k is PSD if and only if k is the Fourier transform (FT) of a non-negative measure p,*

$$k(\Delta) = \int_{\mathbb{R}^d} p(\omega) e^{i\langle \omega, \Delta \rangle} \mathrm{d}\omega. \tag{19}$$

Furthermore, if $k(0) = 1$, then p is a probability distribution $\int_{\mathbb{R}^d} p(\omega) \mathrm{d}\omega = 1$.

Bochner's theorem is also a central ingredient in a powerful kernel-approximation algorithm known as *random Fourier features* (RFF) [44] presented here as algorithm 2.

Algorithm 2. Random Fourier features [44], `RFF`.

Input: A PSD, continuous, shift-invariant kernel $k(x - x')$.
Output: A feature map z so that $\langle z(x), z(x') \rangle \approx k(x - x')$.

1: $p(\omega) \leftarrow \frac{1}{2\pi} \int_{\mathbb{R}^d} e^{-i\langle \omega, \Delta \rangle} k(\Delta) \mathrm{d}\Delta$ ▷ Inverse FT of k.

2: $\{\omega_1, \dots, \omega_{D/2}\} \sim p^{D/2}$ ▷ Draw $D/2$ i.i.d. samples from p.

3: $z(\cdot) \leftarrow \sqrt{\frac{2}{D}} \begin{pmatrix} \cos\langle \omega_1, \cdot \rangle \\ \sin\langle \omega_1, \cdot \rangle \\ \vdots \\ \cos\langle \omega_{D/2}, \cdot \rangle \\ \sin\langle \omega_{D/2}, \cdot \rangle \end{pmatrix}.$

4: **return** z

The input to algorithm 2 is any shift-invariant kernel function k, and the output is a feature map z such that the same kernel can be ε-approximated as an explicit inner product. In Step 1, the inverse Fourier transform p of the kernel k is produced. In Step 2, $D/2$ many samples ω_i are drawn i.i.d. according to p. In Step 3, the feature map z is constructed using the samples ω_i and the sine and cosine functions. That the inner product of z gives a good approximation to k is further elucidated in appendix A.

With RFF, given a kernel function, algorithm 2 produces a randomized feature map such that the kernel corresponding to the inner product of pairs of such maps is an unbiased estimator of the initial kernel. The vital question is how large the dimension D has to be in order to ensure ε approximation error, which is the object of study of theorem 5.

Theorem 5 (Random Fourier features, claim 1 in [44]). *Let $\mathcal{X} \subseteq \mathbb{R}^d$ be a compact data domain. Let k be a continuous shift-invariant kernel acting on $\mathcal{X}$, fulfilling $k(0) = 1$. Then, for the probabilistic feature map $z(\cdot) \colon \mathcal{X} \to \mathbb{R}^D$ produced by algorithm 2,*

$$\mathbb{P}\left[\sup_{x,x' \in \mathcal{X}} |\langle z(x), z(x') \rangle - k(x, x')| \geqslant \varepsilon \right] \leqslant \tag{20}$$

$$\leqslant 2^8 \left(\frac{\sigma_p \operatorname{diam}(\mathcal{X})}{\varepsilon} \right)^2 \exp\left(-\frac{D\varepsilon^2}{8(d+2)} \right) \tag{21}$$

holds for almost every $x, x' \in \mathcal{X}$. Here $\operatorname{diam}(\mathcal{X}) = \sup_{x,x' \in \mathcal{X}} \{\|x - x'\|\}$ is the diameter of $\mathcal{X}$, and σ_p is the variance of the inverse FT of k interpreted as a probability distribution

$$p(\omega) = \mathrm{FT}^{-1}[k](\omega), \tag{22}$$

$$\sigma_p^2 = \mathbb{E}_p \left[\|\omega\|^2 \right]. \tag{23}$$

In particular, it follows that for any constant success probability, there exists an ε-approximation of k as an Euclidean inner product where the feature space dimension D satisfies

$$D \in \mathcal{O}\left(\frac{d}{\varepsilon^2} \log \frac{\sigma_p \operatorname{diam}(\mathcal{X})}{\varepsilon} \right). \tag{24}$$

The RFF construction can fail to produce a good approximation with a certain probability, but the failure probability can be pushed down arbitrarily close to 0 efficiently.

This theorem can be understood as a probabilistic existence result for efficient embedding-quantum approximations of kernel functions, which we present next as our second main contribution. We consider the input dimension d as our scaling parameter, which plays the role of s. We further take ε-approximation to be the supremum of the pointwise difference almost everywhere, which we inherit from theorem 5. In this result we do not need to fix how the input kernel sequence $\{k_s\}_{s\in\mathbb{N}}$ is specified, but we assume it is specified in a way that allows us to approximate it using a quantum computer. When it comes to what definition of efficiency we need to use, we consider first space efficiency, since we talk about the required number of dimensions to approximate the kernel as an explicit inner product. Later on we pin down the time complexity as well.

In the following, for clarity of presentation, we decouple the construction of EQKs via RFFs as a two-step process: first produce the (classical) feature map z, and second realize the inner product as an EQK as in lemma 3. As of now, algorithm 2 produces a classical approximation via Random Fourier Features, not an EQK yet. Next we add a smoothness assumption to theorem 5 to produce corollary 6. Further below we introduce algorithm 3, which takes the feature map z from algorithm 2 and produces an EQK from it using algorithm 1.

Corollary 6 (Smooth shift-invariant kernels). *Let $\mathcal{X}_d \subseteq [-R,R]^d$ be a compact domain. For $\varepsilon > 0$, let k_d be a continuous shift-invariant kernel function. Assume the kernel fulfills $k_d(0) = 1$ and it has bounded second derivatives at the origin $|\partial_i^2 k_d(0)| \leqslant B$. Then algorithm 2 produces an ε-approximation of k_d as an explicit inner product. In particular, the scaling of the required dimension D of the (probabilistic) feature map is*

$$D \in \mathcal{O}\left(\frac{d}{\varepsilon^2} \log \frac{Rd\sqrt{B}}{\varepsilon}\right). \tag{25}$$

Proof. From Bochner's theorem (theorem 4), we know that k_d is the FT of a probability distribution p_d. Next, a standard Fourier identity allows us to relate the variance σ_d^2 of p_d and the trace of the Hessian $H(k_d)$ at the origin $\Delta = 0$ (see, e.g. [44, 47]):

$$\sigma_d^2 = -\operatorname{tr}\left\{ H(k_d)|_{\Delta=0} \right\} = -\sum_{i=1}^{d} \left.\frac{\partial^2 k_d}{\partial \Delta_i^2}\right|_{\Delta=0}. \tag{26}$$

Using the assumption that $|\partial_i^2 k(0)| \leqslant B$ for all $i \in [d]$, this results in $\sigma_d^2 \leqslant dB$. In parallel, we can upper bound the diameter of $\mathcal{X}_d$ by the diameter of $[-R,R]^d$ to obtain $\operatorname{diam}(\mathcal{X}_d) \leqslant 2R\sqrt{d}$. By plugging the bounds on σ_d^2 and $\operatorname{diam}(\mathcal{X}_d)$ into the bound of theorem 5 we obtain the claimed result

$$D \in \mathcal{O}\left(\frac{d}{\varepsilon^2} \log \frac{\sigma_d \operatorname{diam}(\mathcal{X}_d)}{\varepsilon}\right) \in \mathcal{O}\left(\frac{d}{\varepsilon^2} \log \frac{Rd\sqrt{B}}{\varepsilon}\right). \tag{27}$$

$\square$

In appendix C, we provide corollary C.1, which fixes the number of bits of precision required to achieve a close approximation to the second derivative using finite difference methods.

Indeed, corollary 6 ensures that the Hilbert-space dimension required to ε-approximate any shift-invariant kernel fulfilling mild conditions of smoothness scales at most polynomially in the relevant scale parameters, if R and B are considered to be constant. This does not provide an answer to question 1 yet, as the result only talks about existence of a space-efficient approximation, and not about the complexity of finding such an approximation only from the specification of each kernel in the sequence $\{k_d\}_d$. Noteworthy is that so far we have not made any assumptions on the complexity of finding the inverse FT of k_d, nor the complexity of sampling from it.

Before going forward, one could enquire whether the upper bound from corollary 6 could be forced to scale exponentially in d. In this direction, one would e.g. consider cases where R and B are not fixed, but rather also depend on d. But, since both R and B appear inside a logarithm, in order to achieve a scaling exponential in d overall, we would need to let either R or B to grow doubly-exponentially. While that remains a possibility, we point out that in such a case we would easily lose the ability to ε-approximate each of the kernels k_d quantum-efficiently to begin with, so we judge these scenarios as less relevant. And even then, it should be noted that theorem 5 offers only an upper bound to the required dimension D. In order to discuss whether D can be forced to scale exponentially, we would also need a lower bound. The same reasoning applies for ε.

Notice in corollary 6 we talk about the required feature dimension, not the number of qubits. Indeed, we can encode the feature vectors (which are nicely normalized) onto quantum states, as presented in

algorithm 3, which we call *quantum random Fourier features* (QRFF). What QRFF does is first obtain the probablistic map z from RFF, and then encode it into quantum states and take their inner product as in lemma 3. By construction, the feature maps produced by RFF are unit vectors with respect to the 2-norm, $\|z(x)\|_2 = 1$ for any $x \in \mathcal{X}$. For algorithm 1 we require unit vectors with respect to the 1-norm, so we need to renormalize the vectors and this introduces another multiplicative factor, which now depends on the input vectors:

Lemma 7 (Inner product normalization). *Let $r, r' \in \ell_2^d$ be 2-norm unit vectors $\|r\|_2 = \|r'\|_2 = 1$. Then, the identity*

$$\langle r, r' \rangle = \|r\|_1 \|r'\|_1 \left(2^n \operatorname{tr} \{ \rho_{\tilde{r}} \rho_{\tilde{r}'} \} - 1 \right) \tag{28}$$

holds, where $\tilde{r}^{(\prime)} = r^{(\prime)} / \|r^{(\prime)}\|_1 \in \ell_1^d$ corresponds to renormalizing with respect to the 1-norm. Here $\rho_{\tilde{r}}$ refers to encoding $\tilde{r}$ onto a quantum state using algorithm 1.

Proof. The proof is given in appendix B. □

Remark (Bounded pre-factors). The re-normalization is of no concern, since the fact r, r' are 2-norm unit vector implies that their 1-norm is bounded $\|r\|_1 \in [1, \sqrt{d}]$.

This explains the extra factor $g(x)g(x')$ appearing in algorithm 3, where we have

$$
\begin{aligned}
g(x) &= \|z(x)\|_1 \\
&= \sqrt{\frac{2}{D}} \sum_{i=1}^{D/2} |\cos\langle \omega_i, x \rangle| + |\sin\langle \omega_i, x \rangle|,
\end{aligned} \tag{29}
$$

and $g(x) \in [1, \sqrt{D}]$, so $g(x)g(x') \in [1, D]$ for all $x, x' \in \mathcal{X}$.

Algorithm 3. Quantum random Fourier features, `QRFF`.

Input: A PSD, continuous, shift-invariant kernel $k(x - x')$.
Output: A quantum feature map ρ so that
    $k(x - x') \approx g(x)g(x')(2^n \operatorname{tr}\{\rho(x), \rho(x')\} - 1)$.   ▷ $g(x)$ defined in equation (29).
  1: $z \leftarrow \texttt{RFF}(k)$ ▷ Apply algorithm 2.
  2: $\rho(\cdot) \leftarrow \texttt{C2QE}(z(\cdot))$ ▷ Apply algorithm 1.
  3: **return** ρ

The number of qubits required n scales logarithmically in the dimension of the feature vector $n \in \mathcal{O}(\log(D))$. So the number of qubits n necessary for ε-approximating shift-invariant kernel functions as EQKs has scaling $n \in \tilde{\mathcal{O}}\left(\log d/\varepsilon^2\right)$, where the tilde hides doubly-logarithmic contributions.

With these, we can almost conclude that corollary 6 results in a positive answer to question 1 for shift-invariant kernels provided they are smooth, with smoothness quantified as the magnitude of the second derivative at the origin. We are still missing the complexity of producing samples from the inverse FT of each k_d, as in Step 2 of algorithm 2. The efficiency criterion taken here would be the required number of qubits. Nevertheless, it is true that for any smooth shift-invariant kernel, there exists an ε-approximation as an EQK using at most logarithmically many qubits following algorithm 3.

The complexity of algorithm 2 could become arbitrarily large based on the difficulty of sampling from the distribution p_d corresponding to the inverse FT of k_d. In order to ensure embedding-quantum efficiency, we need to add the requirement of efficient sampling from the inverse FT of the kernel. With this, we proceed to state our main result: efficient EQKs are universal within the class of smooth shift-invariant kernels whose inverse FT is efficiently sampleable.

Corollary 8 (Polynomial run-time). *Under the assumptions of corollary 6, let p_d be the inverse FT of the kernel k_d. If obtaining samples from p_d can be done in time $t \in \mathcal{O}(\operatorname{poly}(d))$, then both algorithms 2 and 3 have total run-time complexity polynomial in d.*

Proof. For algorithm 2, Step 1 is only a mathematical definition, Step 2 runs in time linear in D and polynomial in d by assumption, finally Step 3 also runs in time linear in D and polynomial in d. Corollary 6 further states that D is at most essentially linear in d. In total, algorithm 2 has run-time complexity at most polynomial in d.

In turn, the first step of algorithm 3 is calling algorithm 2, which we just saw takes time polynomial in d. Step 2 uses algorithm 1, which also runs in time polynomial in d. With this it is also clear that the total run-time complexity of algorithm 3 is at most polynomial in d. □

With the added condition of producing samples in polynomial time, we obtain a positive answer to question 1. Namely, we show the existence of time-efficient EQK approximations for any kernel fulfilling the smoothness and efficient sampling conditions from corollaries 6 and 8. Notice our result does not even require quantum efficiency of k_d, so in fact we have proved something stronger.

One point to address is whether assuming quantum efficiency for evaluating k_d directly implies the ability to sample efficiently from the inverse FT of k_d. In the cases where this is true, adding quantum-efficiency to the assumptions of corollary 6 suffices to prove that efficient EQKs are universal within the class of efficient shift-invariant kernels. At the face of it, it is unclear whether for all reasonable ways of specifying k_d the capacity to efficiently evaluate it using a quantum computer implies an efficient algorithm of sampling from the distribution obtained by the inverse FT. If the task were to sample from k_d itself (in case k_d were a probability distribution), then we know that being able to evaluate k_d does not imply the capacity to efficiently sample from k_d[8]. Then, it is unclear why sampling from the inverse FT of k_d would be easy in every case, especially, e.g. in the black mox model. Resolving this question is left as an open problem, and we note it is not an entirely new one [49, 50].

Finally, it should be noted that, although the feature map produced by algorithm 2 can be stored classically (assuming $D \in \mathcal{O}(d/\varepsilon^2)$), this is not a de-quantization algorithm. Algorithm 2 requires sampling from the inverse FT of the input kernels k_d. Reason dictates that, if the kernel is quantum efficient and classically inefficient to ε-approximate, in general sampling from its inverse FT should also be at least classically inefficient. This is what we meant earlier when we said we consider quantum time: even though the intermediate variable $z(x)$ can be stored classically, producing it requires usage of a quantum computer.

In this section, we show that smooth shift-invariant kernels admit space-efficient EQK approximations. We also give sufficient conditions for the same result to hold for embedding-quantum efficiency in total runtime. In the next section, we see how the same ideas extend to another class of kernel functions, beyond shift-invariant ones.

6. Composition kernel and projected quantum kernel

In this section we introduce a new class of quantum kernels which can still be turned into EQKs using another variant of algorithm 2. We show that the new class also admits time-efficient approximations as EQK, and that the projected quantum kernel from [28] belongs to this class. During the publication phase of this manuscript we were made aware of the similarities between the composition kernels we introduce here and the 'distance kernel with RDM' proposed and studied in [43].

Again, we separate the EQK-based approximation into two steps: first we propose a variant of RFF producing a classical feature map, and second we construct an EQK that evaluates the inner product of pairs of features.

First we introduce the new class. Consider the usual Gaussian kernel with parameter $\sigma > 0$ defined as

$$k(x, x') = e^{-\frac{\|x - x'\|^2}{2\sigma^2}}, \tag{30}$$

only now we allow for some pre-processing of the inputs $x \mapsto f(x)$, resulting in

$$k_f(x, x') = e^{-\frac{\|f(x) - f(x')\|^2}{2\sigma^2}}. \tag{31}$$

The introduction of f breaks shift invariance in general, hence the need to specify both arguments independently $x, x' \in \mathcal{X}$. Since k_f is a PSD kernel for any function f, we refer to such constructions as *composition kernel*. Next we propose a generalization of algorithm 2 that also works for composition kernels, as algorithm 4, which we call simply *random Fourier features with pre-processing* (RFF_pp).

---

[8] In the black box model, lower bounds on Grover's search algorithm imply an exponential cost of sampling. Also, NP-hardness of e.g. sampling from low-temperature Gibbs states of the Ising model [48] provides examples relevant for other input models.

Algorithm 4. RFF with pre-processing, `RFF_pp`.

Input: A composition kernel $k_f(x, x')$, see equation (31).
Output: A feature map z_f so that $k_f(x, x') \approx \langle z_f(x), z_f(x') \rangle$.
  1: $z \leftarrow \text{RFF}(k \colon f(\mathcal{X}) \to \mathbb{R})$ ▷ Apply algorithm 2.
  2: $z_f(\cdot) \leftarrow z(f(\cdot))$ ▷ Apply the pre-processing function.
  3: **return** z_f

Since the core feature of algorithm 4 is to call algorithm 2, we inherit the ε-approximation guarantee. Notice in line 1 of algorithm 4 we invoke algorithm 2, but taking as input domain the range of the pre-processing function, $f(\mathcal{X})$, instead of the original domain $\mathcal{X}$. If we assume f to be continuous, it follows that $f(\mathcal{X})$ is also compact, which we require for the application of theorem 5.

Proposition 9 (Performance guarantee of algorithm 4). *Let $f \colon \mathcal{X} \to [-B, B]^{g_1(d)}$ be a pre-processing function, and let k_f be the Gaussian kernel composed with f, as introduced in equation (31). Let finally the parameter of the Gaussian kernel be $\sigma = g_2(d)$. If $g_1(d) \in \mathcal{O}(\text{poly}(d))$ and $g_2(d) \in \Omega(\text{poly}(d)^{-1})$, then algorithm 4 produces an ε-approximation of $k_f(x, x')$ as an explicit inner product. In particular, the required dimension D of the randomized feature map is at most polynomial in the input dimension d and the inverse error $1/\varepsilon$.*

$$D \in \mathcal{O}\left(\frac{\text{poly}(d)}{\varepsilon^2} \log\left(\frac{dB}{\varepsilon} \right) \right) \in \tilde{\mathcal{O}}\left(\frac{\text{poly}(d)}{\varepsilon^2} \right). \tag{32}$$

Proof. The proof is provided in appendix D.   □

Notice proposition 9 shares a deep similarity with corollary 6. Both are direct applications of theorem 5 to kernels fulfilling different properties. They have nevertheless one stark difference. The assumptions in proposition 9 are sufficient to guarantee we can sample efficiently from the inverse FT of the kernels in the sequence k_d. This is because the probability distributions involved, p_d, are nothing but Gaussian distribution themselves. As shown in algorithm 4, the pre-processing function f shows up only after the sampling step. So, unlike corollary 6, proposition 9 already represents a direct positive answer to question 1, relying on of algorithm 5, which follows.

In algorithm 5, *quantum random Fourier features with pre-processing* (QRFF_pp), we take the output of algorithm 4 and convert it into a quantum feature map with algorithm 1, leading to the corresponding EQK approximation. Again now the multiplicative term $g(f(x))g(f(x'))$ appears from the re-normalization in lemma 7.

Algorithm 5. QRFF with pre-processing, `QRFF_pp`.

Input: A composition kernel $k_f(x, x')$, see equation (31).
Output: A quantum feature map ρ_f so that
    $k_f(x, x') \approx g(f(x))g(f(x'))(2^n \text{tr}\{\rho_f(x)\rho_f(x')\} - 1)$.   ▷ $g(x)$ defined in equation (29).
  1: $z_f \leftarrow \text{RFF\_pp}(k_f)$ ▷ Apply algorithm 4.
  2: $\rho_f(\cdot) \leftarrow \text{C2QE}\left(z_f(\cdot) \right)$ ▷ Apply algorithm 1.
  3: **return** ρ_f

The scaling in the number of qubits is once more logarithmic in the scaling of the number of required dimensions given in proposition 9. For the number of qubits n, the scaling is $n \in \tilde{\mathcal{O}}\left(\log(d/\varepsilon^2) \right)$. Moreover, the run-time complexity of algorithm 5 is polynomial in d and in the run-time complexity of evaluating f, since now the sampling step corresponds to sampling from a product Gaussian distribution.

For completeness sake, it would be good also to rule out the possibility of classically efficient approximation. Although it might be counter-intuitive, in principle there could exist pre-processing functions f that are hard to evaluate but which result in a composition kernel k_f that is not hard to evaluate. In the following we just confirm that there exist pre-processing functions which are hard to evaluate classically which result in composition kernels which are also hard to evaluate classically.

Proposition 10 (No efficient classical approximation). *There exists a function $f \colon \mathcal{X} \to [0, 1]^d$ which can be ε-approximated quantum efficiently in d for which the composition kernel k_f cannot be ε-approximated classically efficiently in d, with*

$$k_f(x, x') := \exp\left(-\|f(x) - f(x')\|^2 \right). \tag{33}$$

We take $\sigma^2 = 1/2$ for simplicity.

Proof. Here, the proof is presented in appendix D. $\qquad\square$

By selecting pre-processing functions that are quantum efficient but classically inefficient, we reach a class of quantum kernels that are not shift-invariant, but for which the RFF_pp construction still applies, from algorithm 4. Next, we show that this class of kernels contains the recently introduced *projected quantum kernel* [28].

Going back to the quantum feature map $x \mapsto \rho(x)$, the authors of [28] proposed a mapping from the exponentially-sized $\rho(x)$, to an array of reduced density matrices $(\rho_k(x))_{k=1}^N$, for an N-qubit quantum state[9]. According to our definitions, we would not call this a quantum feature map, since the data is not mapped onto the Hilbert space of quantum states. Rather, it is natural to think of this as a quantum pre-processing function. A nice property of this alternative mapping is that the feature vectors can be efficiently stored in classical memory, even though obtaining each of the $\rho_k(x)$ matrices can only be done efficiently using a quantum computer in general. Once these are stored, the projected quantum kernel k^{PQ} is the composition kernel as we introduced earlier, just setting f to be the function that computes the entries of all the reduced density matrices

$$k^{\mathrm{PQ}}(x,x') = \exp\left(-\gamma \sum_{k=1}^N \|\rho_k(x) - \rho_k(x')\|_F^{\,2}\right), \tag{34}$$

where $\gamma > 0$ can be safely taken as $\gamma = 1/(2\sigma^2)$, $\rho_k(x)$ is the reduced density matrix of the kth qubit of ρ, and recall $\|\cdot\|_F$ is the Frobenius norm. The number of qubits N is left as a degree of freedom, but for d-dimensional input data, reason would say the number of qubits used would be $N \propto d$, or at most $N \in \mathcal{O}(\mathrm{poly}(d))$.

The projected quantum kernel enjoys valuable features. On the one hand, it was used in an effort to prove quantum–classical learning separations in the context of data from quantum experiments [28]. On the other hand, the projected quantum kernel is less vulnerable to the exponential kernel concentration problem [38]. The projected quantum kernel is also deeply related to the shadow tomography formalism and guarantees can be placed on its performance for quantum phase recognition [45] among others.

Proposition 11 (Projected quantum kernel as an efficient EQK). *The projected quantum kernel k^{PQ} fulfills the assumptions of proposition 9, so it can be efficiently approximated as an EQK with the number of dimensions required D fulfilling*

$$D \in \tilde{\mathcal{O}}\left(\frac{\mathrm{poly}(d)}{\varepsilon^2}\right). \tag{35}$$

Proof. The proof of this statement is presented in appendix D. $\qquad\square$

Remark (General projected quantum kernels). In the proof, we have assumed that the projected quantum kernel is taken with respect to the subsystems being each individual qubit. A general definition of the projected quantum kernel would allow for other subsystems, but the requirement that the feature map must be efficient to store classically prevents that the number of subsystems scales more than polynomially, and that the local dimension of each subsystem scales more than logarithmically, in which case we still have $g_1(d) \in \mathcal{O}(\mathrm{poly}(d))$.

The projected quantum kernel poses a clear example of a quantum kernel that is not constructed from a feature map onto the Hilbert space of density operators. Its closeness to the classical shadow formalism has earned it attention from the moment it was proposed. Thus, proving that also the projected quantum kernel admits an efficient approximation as an EQK results in another important kernel family that is also covered by EQKs.

In order to identify scenarios in which the scaling D could become super-polynomial in the relevant parameters $d, 1/\varepsilon$, one would require for example $g_1(d) \propto \exp(d)$, or $g_2(d) \propto 1/\exp(\exp(d))$, using the definitions from proposition 9. Indeed, in those scenarios we would not be able to guarantee the space efficiency of algorithm 4 while keeping good ε-approximation. Nevertheless the conditions $g_1(d) \propto \exp(d)$ or $g_2(d) \propto 1/\exp(\exp(d))$ alone would not be sufficient to prove that algorithm 4 must fail. Said otherwise, this is again because from theorem 5 we only have an upper bound on the required space complexity, and not a lower bound. At the same time, though, both cases would prevent us from being able to quantum-efficiently ε-approximate the projected quantum kernel in the first place, so these scenarios must be ruled out as they break the hypotheses. And even if they were allowed, recall that we can store classical vectors in quantum

---

[9] We distinguish N, the number of qubits used to evaluate the projected quantum kernel, from n, the number of qubits required to approximate it as an EQK.

states using only logarithmically many qubits in the length of the classical vector. That means that single exponential $g_1(d)$ and inverse doubly-exponential $g_2(d)$ would not be enough to require exponential embedding-quantum complexity in the number of qubits. In order to have the number of qubits to be, e.g. $n \in \mathcal{O}(\exp(d))$, we would require either $g_1(d) \in \mathcal{O}(\exp(\exp(d)))$, or $g_2(d) \in \Omega(1/\exp(\exp(\exp(d))))$, which would prevent our ability to evaluate the projected kernel even further.

With these considerations, we have shown that projected quantum kernels, despite their using a different feature map and inner product, can be efficiently realized as EQKs. As a recapitulation, figure 3 summarizes all our contributions as a collection of set inclusions in a Venn diagram, and figure 4 displays the inner workings of the Algorithms we propose. With this, to the best of our knowledge, we conclude that all quantum kernels in the literature are either EQKs directly, or they can be efficiently realized as EQKs.

7. Outlook

This manuscript so far offers restricted positive answers to question 1. In this section, we list some promising directions to search for other restricted answers to the same question. Answering question 1 negatively would involve proving an non-existence result, for which one needs a different set of tools than the ones we have used so far. In particular, one would need to wield lower bounds for non-existence, which in this corner of the literature appear to be trickier to find.

7.1. Time efficient EQKs
It should be possible to come up with a concrete construction that ensures also time efficiency of algorithm 2 while applying corollary 8. For that to happen, it would be enough to give not only the kernel function, but also an efficient algorithm to sample from its inverse FT. We recognize this question as the clearest next step in this new research direction.

7.2. Non-stationary kernels and indefinite kernels
The theory of integral kernel operators and their eigenvalues has been already extensively developed in the context of functional analysis [51–53], also with applications to random processes and correlation functions [54]. In particular, for 1-dimensional kernel functions, it is known that reasonable smoothness assumptions lead to kernel spectra which are concentrated on few eigenfunctions. This fact alone hints strongly toward the existence of low-dimensional EQK-based approximations for a larger class of kernel functions than the ones we explored here. While we restricted ourselves to shift-invariant and composition kernels, the theory of eigenvalues of integral kernel operators seems to suggest that similar results could be obtained for any smooth kernel function. That being said, the generalization from 1 to d-dimensional data might well come with factors which grow exponentially with d, somewhat akin to the well-known *curse of dimensionality*.

Alternatively, instead of arbitrary PSD kernel functions, one might also consider other restricted classes of known kernels. For example rotation-invariant kernels, of which polynomial kernels are a central example, offer an interesting starting point. Polynomial kernels are not shift-invariant, but they are often derived with an explicit feature map in mind. A potentially interesting research line would be to generalize polynomial kernels in a way that is less straightforward to turn into an EQK.

Our results relied strongly on the *random Fourier features* (RFF) algorithm of [44]. The RFF approximation in turn rests on a sampling protocol that owes its success to Bochner's theorem for shift-invariant PSD functions (theorem 4). Yaglom's theorem (theorem A.4 in appendix A) is referred to as the generalization of Bochner's theorem to functions that are not shift-invariant. It would be interesting to see whether there are other, non shift-invariant kernels for which Yaglom's theorem could be used to furbish an explicit feature map with approximation guarantees akin to those of theorem 5.

The extension from PSD to non-PSD functions could be similar to the extension from shift-invariant to smooth functions we just described. Intuitively, finding an EQK approximation for a kernel is similar to finding its singular value decomposition in an infinite-dimensional function space. In this space, PSD functions result in the 'left'- and 'right'-hand matrices in the singular value decomposition to be equal (up to conjugation). Conversely, for non-PSD functions, the main difference would be the difference between left and right feature maps.

In this direction, we identify four promising research lines: first, generalize our study to the general class of 1-dimensional smooth functions; second, study restricted classes of higher-dimensional kernel functions for which the integral kernel operator retains desirable qualities for approximation; third, find restricted cases where a kernel approximation based on Yaglom's theorem is possible; and fourth, extend similar results for indefinite kernel functions. We comment on these directions in appendix A.

7.3. The variational QML lens

When studying EQK-based approximations to given kernel functions, we have not restricted ourselves to PQC-based functions. Even though the literature on quantum kernel methods up to this point has not dealt exclusively with variational circuits, a large body of work has indeed focused on functions estimated as the expectation value of a fixed observable with respect to some parametrized quantum state. In this sense, combining the results from [31] and the rules for generating PSD-functions from [47], one can develop a framework for how to approach quantum kernel functions holistically. We give some first steps explicitly in appendix E. We expect many discoveries to result from an earnest study of what are the main recipes to construct PQC-based quantum kernels other than by the established EQK principles.

8. Conclusion

In this manuscript we raise and partially answer a fundamental question regarding the expressivity of mainstream quantum kernels. We identify that most quantum kernel approaches are of a restricted type, which we call *embedding quantum kernels* (EQKs), and we ask whether this family covers all quantum kernel functions. If we leave notions of efficiency aside, we show that all kernel functions can be realized as EQKs, proving their universality. Universality is an important ground fact to establish, as it softly supports the usage of EQKs in practice.

Learning whether EQKs are indeed everything we need for QML tasks or whether we should look beyond them is important in the context of model selection. In ML for kernel methods, model selection boils down to choosing a kernel. For ML to be successful, it is long known that models should be properly aligned with the data; for different learning tasks, different models should be used. This notion is captured by the concept of *inductive bias* [55, 56], which is different for every kernel. Given a learning task with data, the first step should always be to gather information about potential structures in the data in order to select a model with a beneficial inductive bias. In this scenario, it becomes crucial to then have access to a model with good data-alignment. In [55] some simple EQKs were found to possess inductive biases which are uninteresting for practically relevant data sets. These findings fueled the need to ask whether EQKs can also have interesting inductive biases, which is softly confirmed based on their universality. When searching for new quantum advantages in ML, the more interesting classes of kernels we have access to, the better.

We propose question 1 as a new line of research in *quantum machine learning* (QML). Characterizing a relation of order between general quantum kernel functions and the structured EQK functions when considering computational efficiency could have an important impact in the quest for quantum advantages. Indeed, if it were found that all practically relevant kernels can be realized as EQKs, then researchers in quantum kernel methods would not need to look for novel, different models to solve learning tasks with. Nevertheless, for now there is still room for practically interesting kernels beyond efficient EQKs to exist.

Operationally, the results presented in this manuscript aim at confirming that the class of kernels currently used in practice already provably covers large, promising families of kernel functions. The question we focus on is what is ultimately possible with efficient EQKs, and not so much what is the best way of evaluating kernel functions, nor how to construct new, essentially different quantum kernels. For instance, given a kernel function and an efficient algorithm to evaluate it, it is not our goal to provide a new evaluation algorithm which is more efficient in some sense. Instead, we are interested in knowing whether there exists another similarly efficient algorithm but which fulfills the added structural restriction of being based on an efficient quantum embedding. Whether the embedding-based evaluation algorithm has a slightly higher or lower computational cost is beyond our scope, we are interested in the embedding-based algorithm to be efficient by itself.

After raising question 1, we give an answer for the restricted case of shift-invariant kernels. In corollary 6 we show that, under reasonable assumptions, all shift-invariant kernels admit a memory-efficient approximation as EQKs. Shift-invariant kernels are widely used also in classical ML, so showing that EQKs are still universal for shift-invariant kernels with efficiency considerations is an important milestone.

While shift-invariant kernels enjoy a privileged position in the classical literature, they have not been as instrumental in QML so far. Indeed, many well-known quantum feature maps lead to EQK functions which are not shift-invariant. Also, the milestone work [28] has introduced the so-called projected quantum kernel, which has been used to prove learning separations between classical and quantum ML models. The projected quantum kernel is not shift-invariant. In proposition 9, we adapt our result for shift-invariant kernels so that they carry over to another class of kernel functions, which we call composition kernels, in which the projected quantum kernel is contained. This way, we present the phenomenon that even kernel functions that did not arise from an explicit quantum feature map can admit an efficient approximation as EQKs.

In all, we have seen that two relevant classes of kernels, namely shift-invariant and composition kernels, always admit an efficient approximation as EQKs. Our results are clear manifestations of the expressive

power of EQKs. Nevertheless, many threads remain open to fully characterize the landscape of all efficient quantum kernel functions. We invite researchers to join us in this research line by listing promising approaches as outlook in section 7.

Data availability statement

No new data were created or analysed in this study.

Acknowledgments

ORCID iDs

Elies Gil-Fuster ● https://orcid.org/0000-0003-0411-9757
Jens Eisert ● https://orcid.org/0000-0003-3033-1292
Vedran Dunjko ● https://orcid.org/0000-0002-2632-7955

References

[1] Shor P W 1994 Algorithms for quantum computation: discrete logarithms and factoring *Proc. 35th Annual Symp. on Foundations of Computer Science* (IEEE) pp 124–34
[2] Nielsen M A and Chuang I L 2000 *Quantum Information and Quantum Computation* (Cambridge University Press)
[3] Montanaro A 2016 Quantum algorithms: an overview *npj Quantum Inf.* **2** 15023
[4] Arute F *et al* 2019 Quantum supremacy using a programmable superconducting processor *Nature* **574** 505
[5] Wu Y *et al* 2021 Strong quantum computational advantage using a superconducting quantum processor *Phys. Rev. Lett.* **127** 180501
[6] Hangleiter D and Eisert J 2023 Computational advantage of quantum random sampling *Rev. Mod. Phys.* **95** 035001
[7] Biamonte J, Wittek P, Pancotti N, Rebentrost P, Wiebe N and Lloyd S 2017 Quantum machine learning *Nature* **549** 195
[8] Dunjko V and Briegel H J 2018 Machine learning and artificial intelligence in the quantum domain: a review of recent progress *Rep. Prog. Phys.* **81** 074001
[9] Carleo G, Cirac I, Cranmer K, Daudet L, Schuld M, Tishby N, Vogt-Maranto L and Zdeborová L 2019 Machine learning and the physical sciences *Rev. Mod. Phys.* **91** 045002
[10] Schuld M and Petruccione F 2021 *Machine Learning With Quantum Computers* (Springer)
[11] Cerezo M *et al* 2021 Variational quantum algorithms *Nat. Rev. Phys.* **3** 625
[12] Bharti K *et al* 2022 Noisy intermediate-scale quantum algorithms *Rev. Mod. Phys.* **94** 015004
[13] Schuld M, Fingerhuth M and Petruccione F 2017 Implementing a distance-based classifier with a quantum interference circuit *Europhys. Lett.* **119** 60002
[14] Havlíček V *et al* 2019 Supervised learning with quantum-enhanced feature spaces *Nature* **567** 209
[15] Schuld M and Killoran N 2019 Quantum machine learning in feature Hilbert spaces *Phys. Rev. Lett.* **122** 4 040504
[16] Benedetti M, Garcia-Pintos D, Perdomo O, Leyton-Ortega V, Nam Y and Perdomo-Ortiz A 2019 A generative modeling approach for benchmarking and training shallow quantum circuits *npj Quantum Inf.* **5** 45
[17] Pérez-Salinas A, Cervera-Lierta A, Gil-Fuster E and Latorre J I 2020 Data re-uploading for a universal quantum classifier *Quantum* **4** 226
[18] Lloyd S, Schuld M, Ijaz A, Izaac J and Killoran N 2020 Quantum embeddings for machine learning (arXiv:2001.03622)
[19] Hubregtsen T, Wierichs D, Gil-Fuster E, Derks P-J H S, Faehrmann P K and Meyer J J 2022 Training quantum embedding kernels on near-term quantum computers *Phys. Rev.* A **106** 04243
[20] Benedetti M, Lloyd E, Sack S and Fiorentini M 2019 Parameterized quantum circuits as machine learning models *Quantum Sci. Technol.* **4** 043001
[21] Schuld M 2021 Quantum machine learning models are kernel methods (arXiv:2101.11020)
[22] Jerbi S, Fiderer L J, Nautrup H P, Kübler J M, Briegel H J and Dunjko V 2023 Quantum machine learning beyond kernel methods *Nat. Commun.* **14** 517
[23] Altares-López S, Ribeiro A and García-Ripoll J J 2021 Automatic design of quantum feature maps *Quantum Sci. Technol.* **6** 045015
[24] Gyurik C, Vreumingen, van D and Dunjko V 2023 Structural risk minimization for quantum linear classifiers *Quantum* **7** 893
[25] Landman J, Thabet S, Dalyac C, Mhiri H and Kashefi E 2022 Classically approximating variational quantum machine learning with random Fourier features (arXiv:2210.13200)
[26] Shin S, Teo Y S and Jeong H 2023 Analyzing quantum machine learning using tensor network (arXiv:2307.06937)
[27] Sweke R, Recio E, Jerbi S, Gil-Fuster E, Fuller B, Eisert J and Meyer J J 2023 Potential and limitations of random fourier features for dequantizing quantum machine learning (arXiv:2309.11647)

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

[29] Suzuki Y, Kawaguchi H and Yamamoto N 2022 Quantum Fisher kernel for mitigating the vanishing similarity issue (arXiv:2210.16581)

[30] Mengoni R and Pierro A D 2019 Kernel methods in quantum machine learning *Quantum Mach. Int.* **1** 65

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

[32] Caro M C, Gil-Fuster E, Meyer J J, Eisert J and Sweke R 2021 Encoding-dependent generalization bounds for parametrized quantum circuits *Quantum* **5** 582

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

[34] Micchelli C A, Xu Y and Zhang H 2006 Universal kernels *J. Mach. Learn. Res.* **7** 2651 (available at: www.jmlr.org/papers/v7/micchelli06a.html)

[35] Harrow A W, Hassidim A and Lloyd S 2009 Quantum algorithm for linear systems of equations *Phys. Rev. Lett.* **103** 150502

[36] Rebentrost P, Mohseni M and Lloyd S 2014 Quantum support vector machine for big data classification *Phys. Rev. Lett.* **113** 130503

[37] Sweke R, Seifert J-P, Hangleiter D and Eisert J 2021 On the quantum versus classical learnability of discrete distributions *Quantum* **5** 417

[38] Thanasilp S, Wang S, Cerezo M and Holmes Z 2022 Exponential concentration and untrainability in quantum kernel methods (arXiv:2208.11060)

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

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

[41] Yamasaki H, Subramanian S, Sonoda S and Koashi M 2020 Learning with optimized random features: exponential speedup by quantum machine learning without sparsity and low-rank assumptions *Proc. 34th Int. Conf. on Neural Information Processing Systems (NIPS'20)* pp 13674–87 (available at: https://dl.acm.org/doi/10.5555/3495724.3496871)

[42] Bach F 2017 On the equivalence between kernel quadrature rules and random feature expansions *J. Mach. Learn. Res.* **18** 1 (available at: http://jmlr.org/papers/v18/15-178.html)

[43] Nakaji K, Tezuka H and Yamamoto N 2022 Deterministic and random features for large-scale quantum kernel machine (arXiv:2209.01958)

[44] Rahimi A and Recht B 2007 Random features for large-scale kernel machines *Advances in Neural Information Processing Systems* vol 20

[45] Huang H-Y, Kueng R, Torlai G, Albert V V and Preskill J 2022 Provably efficient machine learning for quantum many-body problems *Science* **377** eabk3333

[46] Rudin W 1990 *Fourier Analysis on Groups* (Wiley)

[47] Schölkopf B and Smola A J 2002 *Learning With Kernels: Support Vector Machines, Regularization, Optimization and Beyond* (*Adaptive Computation and Machine Learning*) (MIT Press)

[48] Troyer M and Wiese U-J 2005 Computational complexity and fundamental limitations to fermionic quantum Monte Carlo simulations *Phys. Rev. Lett.* **94** 170201

[49] Kalai G 2013 The complexity of sampling (approximately) the Fourier transform of a Boolean function *StackExchange, Theoretical Computer Science* (https://cstheory.stackexchange.com/questions/17431/the-complexity-of-sampling-approximately-the-fourier-transform-of-a-boolean-fu)

[50] Schwarz M and Van den Nest M 2013 Simulating quantum circuits with sparse output distributions (arXiv:1310.6749)

[51] Karlin S 1964 Total positivity, absorption probabilities and applications *Trans. Am. Math. Soc.* **111** 33

[52] Hansen P C 2010 *Discrete Inverse Problems* (Society for Industrial and Applied Mathematics)

[53] Ha C-W 1986 Eigenvalues of differentiable positive definite kernels *SIAM J. Math. Anal.* **17** 415

[54] Yaglom A M 1987 *Correlation Theory of Stationary and Related Random Functions, Volume I: Basic Results* vol 131 (Springer)

[55] Kübler J, Buchholz S and Schölkopf B 2021 The inductive bias of quantum kernels *Advances in Neural Information Processing Systems* vol 34 p 12661

[56] Peters E and Schuld M 2022 Generalization despite overfitting in quantum machine learning models (arXiv:2209.05523)