Quantum Machine Learning Beyond Kernel Methods

Sofiene Jerbi,[1] Lukas J. Fiderer,[1] Hendrik Poulsen Nautrup,[1] Jonas M. Kübler,[2]
Hans J. Briegel,[1] and Vedran Dunjko[3]

[1] *Institute for Theoretical Physics, University of Innsbruck, Technikerstr. 21a, A-6020 Innsbruck, Austria*
[2] *Max Planck Institute for Intelligent Systems, Tübingen, Germany*
[3] *Leiden University, Niels Bohrweg 1, 2333 CA Leiden, Netherlands*

(Dated: October 27, 2021)

With noisy intermediate-scale quantum computers showing great promise for near-term applications, a number of machine learning algorithms based on parametrized quantum circuits have been suggested as possible means to achieve learning advantages. Yet, our understanding of how these quantum machine learning models compare, both to existing classical models and to each other, remains limited. A big step in this direction has been made by relating them to so-called kernel methods from classical machine learning. By building on this connection, previous works have shown that a systematic reformulation of many quantum machine learning models as kernel models was guaranteed to improve their training performance. In this work, we first extend the applicability of this result to a more general family of parametrized quantum circuit models called data re-uploading circuits. Secondly, we show, through simple constructions and numerical simulations, that models defined and trained variationally can exhibit a critically better generalization performance than their kernel formulations, which is the true figure of merit of machine learning tasks. Our results constitute another step towards a more comprehensive theory of quantum machine learning models next to kernel formulations.

Introduction— In the current Noisy Intermediate-Scale Quantum (NISQ) era [1], a few methods have been proposed to construct useful quantum algorithms that are compatible with mild hardware restrictions [2, 3]. Most of these methods involve the specification of a quantum circuit Ansatz, optimized in a classical fashion to solve specific computational tasks. Next to variational quantum eigensolvers in chemistry [4] and variants of the quantum approximate optimization algorithm [5], machine learning approaches based on such parametrized quantum circuits [6] stand as some of the most promising practical applications to yield quantum advantages.

In essence, a supervised machine learning problem often reduces to the task of fitting a parametrized function – also referred to as the machine learning model – to a set of previously labeled points, called a training set. Interestingly, many problems in physics and beyond, from the classification of phases of matter [7] to predicting what structures proteins fold into [8], can be phrased as such machine learning tasks. In the domain of quantum machine learning [9, 10], an emerging approach for this type of problem is to use parametrized quantum circuits to define a hypothesis class of functions [11–16]. The hope is for these parametrized models to offer classification power beyond what is possible with classical models, including the highly successful deep neural networks. And indeed, we have substantial evidence of such a quantum learning advantage for artificial problems [16–21], but the next frontier is to show that quantum models can be advantageous in solving real-world problems as well. To achieve this, we first need a deeper understanding of these quantum methods and how they relate.

Much progress has been made in this direction by exploiting a connection between quantum models and kernel methods from classical machine learning [22].

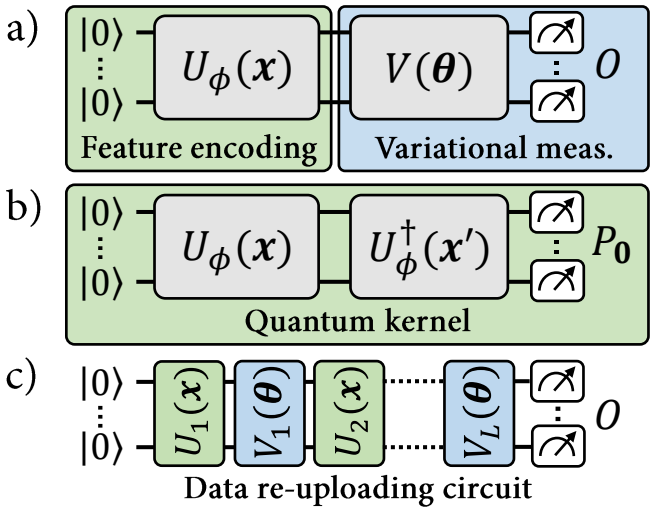

FIG. 1. The three types of quantum machine learning models studied in this Letter. a) An explicit quantum classifier, where the label of a data point $\boldsymbol{x}$ is specified by the expectation value of a variational measurement on its associated quantum feature state $\rho(\boldsymbol{x})$. b) The quantum kernel associated to these quantum feature states. The expectation value of the projector $P_\mathbf{0} = |\mathbf{0}\rangle\langle\mathbf{0}|$ corresponds to the inner product between encoded data points $\rho(\boldsymbol{x})$ and $\rho(\boldsymbol{x'})$. An implicit quantum classifier is defined by a linear combination of such inner products, for $\boldsymbol{x}$ an input point and $\boldsymbol{x'}$ training data points. c) A data re-uploading classifier, interlaying data encoding and variational unitaries before a final measurement.

Many quantum models operate by encoding data in a high-dimensional Hilbert space and using solely inner products evaluated in this feature space to model properties of the data. This is also how kernel methods work. Building on this similarity, the authors of [23, 24] noted

that a given quantum encoding can be used to define two types of models (see Fig. 1): (a) *explicit* quantum classifiers, where an encoded data point is measured according to a variational observable that specifies its label, or (b) *implicit* kernel classifiers, where weighted inner products of encoded data points are used to assign labels instead.[1] In the quantum machine learning literature, much emphasis has been placed on implicit models [20, 25–31], in part due to a fundamental result known as the representer theorem [22]. This result shows that implicit models can always achieve a smaller labeling error than explicit models, when evaluated on the same training set. Seemingly, this suggests that implicit models are systematically more advantageous than their explicit counterparts in solving machine learning tasks [20, 25].

In recent times, there has also been progress in so-called *data re-uploading* models [32] which have demonstrated their importance in designing expressive models, both analytically [33] and empirically [15, 16, 32], and proving that (even single-qubit) parametrized quantum circuits are universal function approximators [34, 35]. However, these models do not naturally fit in the kernel paradigm, as they involve in general several data-encoding layers, interlaid with variational unitaries.

In this Letter, we introduce a unified picture for explicit, implicit and data re-uploading quantum models. We show that all function families stemming from these can be formulated as linear models in suitably defined quantum feature spaces, which allows explicit and data re-uploading models to be systematically mapped into kernel models. However, we show that this mapping can lead to implicit kernel models with a very poor generalization performance, therefore highlighting the potential advantages of quantum models beyond their kernel formulation. We also provide numerical demonstrations of these advantages in engineered learning tasks.

Linear quantum classifiers— Let us first understand how explicit and implicit quantum classifiers can both be described as linear quantum models [25, 36]. Consider a feature encoding unitary $U_\phi(\boldsymbol{x}) : \mathcal{X} \times \mathcal{F} \to \mathcal{F}$ that maps input vectors $\boldsymbol{x} \in \mathcal{X}$, e.g., images represented in $\mathbb{R}^d$, to n-qubit quantum states $\rho(\boldsymbol{x}) = U_\phi(\boldsymbol{x}) \left|\boldsymbol{0}\right\rangle\!\left\langle\boldsymbol{0}\right| U_\phi^\dagger(\boldsymbol{x})$ in the Hilbert space $\mathcal{F}$ of $2^n \times 2^n$ Hermitian operators. An explicit quantum classifier [23, 24] using this feature encoding is defined by a variational family of unitaries $V(\boldsymbol{\theta})$ and an observable O, such that the expectation values

$$f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \mathrm{Tr}[\rho(\boldsymbol{x}) O_{\boldsymbol{\theta}}], \tag{1}$$

for $O_{\boldsymbol{\theta}} = V(\boldsymbol{\theta})^\dagger O V(\boldsymbol{\theta})$, specify its labeling function. We note that this classifier is a linear model in the quantum feature space $\mathcal{F}$. Indeed, one can see from Eq. (1) that $f_{\boldsymbol{\theta}}(x)$ corresponds to the Hilbert-Schmidt inner product between the Hermitian matrices $\rho(\boldsymbol{x})$ and $O_{\boldsymbol{\theta}}$, which is by definition a linear model of the form $\langle \phi(\boldsymbol{x}), w_{\boldsymbol{\theta}} \rangle_{\mathcal{F}}$ in $\mathcal{F}$, for $\phi(\boldsymbol{x}) = \rho(\boldsymbol{x})$ and $w_{\boldsymbol{\theta}} = O_{\boldsymbol{\theta}}$. From this perspective, restricting the family of variational observables $O_{\boldsymbol{\theta}}$ is equivalent to restricting the vectors $w_{\boldsymbol{\theta}}$ accessible to the linear quantum classifier.

Implicit quantum classifiers [23, 24] are constructed from the quantum feature states $\rho(\boldsymbol{x})$ in a different way. Their definition depends explicitly on the data points $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(M)}\}$ in a given training set $\mathcal{D}$, as they take the form of a linear combination

$$f_{\boldsymbol{\alpha},\mathcal{D}}(\boldsymbol{x}) = \sum_{m=1}^{M} \alpha_m k(\boldsymbol{x}, \boldsymbol{x}^{(m)}), \tag{2}$$

for $k(\boldsymbol{x}, \boldsymbol{x}^{(m)}) = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x}^{(m)}) \rangle_{\mathcal{F}} = \mathrm{Tr}[\rho(\boldsymbol{x})\rho(\boldsymbol{x}^{(m)})]$ the *kernel function* associated to the feature encoding $U_\phi(\boldsymbol{x})$. By linearity of the trace however, we can express any such implicit classifier as a linear model in $\mathcal{F}$, defined by the observable:

$$O_{\boldsymbol{\alpha},\mathcal{D}} = \sum_{m=1}^{M} \alpha_m \rho(\boldsymbol{x}^{(m)}). \tag{3}$$

Therefore, both explicit and implicit quantum classifiers belong to the general family of linear models in the quantum feature space $\mathcal{F}$.

Linear realizations of data re-uploading classifiers— Data re-uploading classifiers [32] on the other hand do not naturally fit this formulation. These models generalize explicit classifiers by increasing the number of encoding layers $U_\ell(\boldsymbol{x}), 1 \leq \ell \leq L$ in the circuit, and interlaying them with variational unitaries $V_\ell(\boldsymbol{\theta})$. This results in functions of the form:

$$f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \mathrm{Tr}[\rho(\boldsymbol{x}, \boldsymbol{\theta}) O_{\boldsymbol{\theta}}], \tag{4}$$

for $\rho(\boldsymbol{x}, \boldsymbol{\theta}) = U(\boldsymbol{x}, \boldsymbol{\theta}) \left|\boldsymbol{0}\right\rangle\!\left\langle\boldsymbol{0}\right| U^\dagger(\boldsymbol{x}, \boldsymbol{\theta})$ and $U(\boldsymbol{x}, \boldsymbol{\theta}) = \prod_{\ell=1}^{L} V_\ell(\boldsymbol{\theta}) U_\ell(\boldsymbol{x})$. Given that the unitaries $U_\ell(\boldsymbol{x})$ and $V_{\ell'}(\boldsymbol{\theta})$ do not commute in general, one cannot straightforwardly gather *all* trainable gates in a final variational observable $O'_{\boldsymbol{\theta}} \in \mathcal{F}$ as to obtain a linear model $\tilde{f}_{\boldsymbol{\theta}}(\boldsymbol{x}) = \langle \phi(\boldsymbol{x}), O'_{\boldsymbol{\theta}} \rangle_{\mathcal{F}}$ with a *fixed* quantum feature encoding $\phi(\boldsymbol{x})$. Our first contribution is to show that, by augmenting the dimension of the Hilbert space $\mathcal{F}$ (i.e., considering circuits that act on a larger number of qubits), one can construct such explicit linear realizations $\tilde{f}_{\boldsymbol{\theta}}$ of data re-uploading models. That is, given a family of data re-uploading classifiers $\{f_{\boldsymbol{\theta}}(\cdot) = \mathrm{Tr}[\rho(\cdot, \boldsymbol{\theta}) O_{\boldsymbol{\theta}}]\}_{\boldsymbol{\theta}}$, we can construct an equivalent family of explicit classifiers $\{\tilde{f}_{\boldsymbol{\theta}}(\cdot) = \mathrm{Tr}[\rho'(\cdot) O'_{\boldsymbol{\theta}}]\}_{\boldsymbol{\theta}}$ that represents all functions in the original family, along with a mapping from the former classifiers to the latter.

We first present a simple construction that leads to *approximate* mappings, meaning that these only guarantee $|\tilde{f}_{\boldsymbol{\theta}}(\boldsymbol{x}) - f_{\boldsymbol{\theta}}(\boldsymbol{x})| \leq \delta, \ \forall \boldsymbol{x}, \boldsymbol{\theta}$ for some (adjustable)

[1] Throughout this Letter, we refer to all learning models as classifiers, whether these assign discrete or continuous labels to input data points.

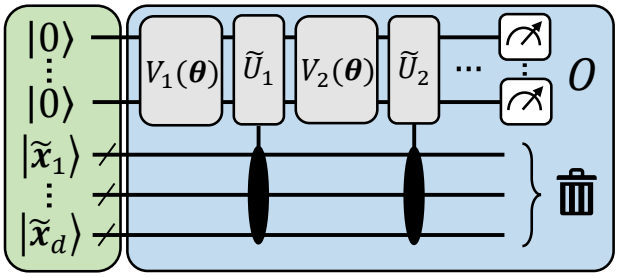

FIG. 2. The illustrative explicit classifier used in our analysis. A quantum circuit acts on $n + dp$ qubits, divided between n working qubits and dp encoding qubits. Pauli-X rotations encode bit-string representations $\widetilde{\boldsymbol{x}}_i \in \{0,1\}^p$ of the d input components $x_i \in \mathbb{R}$, which realizes the feature encoding of the explicit classifier. Fixed and data-independent controlled rotations, interlaid with arbitrary variational unitaries and followed by a measurement of the working qubits can result in a good approximation of any parametrized quantum circuit.

error of approximation δ. In this construction, depicted in Fig. 2, we consider parametrized quantum circuits where all data components $x_i \in \mathbb{R}$ of an input vector $\boldsymbol{x} = (x_1, \ldots, x_d)$ are encoded as bit-strings $|\widetilde{\boldsymbol{x}}_i\rangle = |b_0 b_1 b_2 b_3 \ldots\rangle \in \{0,1\}^p$, to some precision $\varepsilon = 2^{-p}$ (e.g., using $R_x(b_j)$ rotations on $|0\rangle$ states). Now, using p *fixed* rotations, e.g., of the form $R_z(\pi 2^{-j})$, controlled by the bits $|b_j\rangle$ and acting on n additional "working" qubits, one can encode every x_i in arbitrary (multi-qubit) rotations $e^{-\mathrm{i}x_i H}$, e.g., $R_z(x_i)$, arbitrarily many times. Given that all these fixed rotations are data-independent, the feature encoding of any such circuit hence reduces to the encoding of the classical bit-strings $\widetilde{\boldsymbol{x}}_i$, prior to all variational operations. By reproducing the variational unitaries appearing in a data re-uploading circuit and replacing its encoding gates with such controlled rotations, we can then approximate any data re-uploading classifier of the form of Eq. (4). The approximation error δ of this mapping originates from the finite precision ε of encoding $\boldsymbol{x}$, which results in an imperfect implementation of the encoding gates in the original circuit. But as $\varepsilon \to 0$, we also have $\delta \to 0$.

We also present *exact* mappings to explicit classifiers, i.e., that achieve $\delta = 0$ with finite resources. These mappings rely on measurement-based quantum computation [37], a universal model of quantum computation that allows to simulate quantum circuits in a particularly interesting way. Using its gate-teleportation variants notably, each gate in a circuit can be encoded on ancillary qubits and teleported back (via entangled measurements) onto the working qubits when needed. Therefore, by teleporting the data-dependent gates of a data re-uploading circuit in such a way, we can, similarly to our previous mapping, gather all data-encoding gates on ancillas. As we detail in Appendix B however, for gate teleportation to succeed with unit probability, some gate-dependent (and hence data-dependent) corrections on the working qubits are needed for certain measurement outcomes, which hinders our mapping to explicit models.

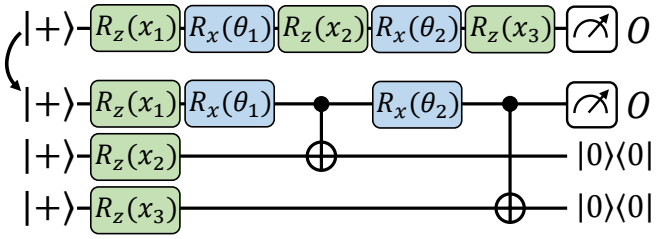

FIG. 3. An exact mapping from a data re-uploading classifier to an equivalent explicit classifier, using gate teleportation.

A straightforward way to get around these corrections is to replace the gate teleportation measurements by projections on the sub-spaces that do not require any corrections. This indeed leads to valid explicit classifiers of the form $\mathrm{Tr}[\rho'(\boldsymbol{x}) O'_{\boldsymbol{\theta}}]$, where the projections have been included in $O'_{\boldsymbol{\theta}}$, such that:

$$\mathrm{Tr}[\rho'(\boldsymbol{x}) O'_{\boldsymbol{\theta}}] = \mathrm{Tr}[\rho(\boldsymbol{x}, \boldsymbol{\theta}) O_{\boldsymbol{\theta}}], \ \forall \boldsymbol{x}, \boldsymbol{\theta}. \tag{5}$$

However, this results in explicit classifiers with a major practical drawback: their implementation requires post-selecting all "correction-free" measurement outcomes, which has an exponentially decaying probability of success in the number of teleported gates D. Mathematically, this means that, in order for Eq. (5) to hold, we need an observable $O'_{\boldsymbol{\theta}}$ such that $\|O'_{\boldsymbol{\theta}}\|_\infty^2 = 2^D \|O_{\boldsymbol{\theta}}\|_\infty^2$, leading to inner products that are exponentially harder to estimate. Nonetheless, we show in Appendix B that, by allowing each encoding gate in the circuit to appear more than once in the explicit feature encoding, we can boost this success probability arbitrarily close to 1, or equivalently, get $\|O'_{\boldsymbol{\theta}}\|_\infty^2$ arbitrarily close to $\|O_{\boldsymbol{\theta}}\|_\infty^2$. We achieve this using a "nested" gate-teleportation scheme.

As our findings indicate, such mappings from data re-uploading to explicit models are not unique, and we believe that many others can also be constructed. In any case, we demonstrated that linear quantum classifiers can describe not only explicit and implicit models, but also data re-uploading circuits. More specifically, we showed that any hypothesis class of data re-uploading models can be mapped to an equivalent class of explicit models, that is, linear models with a restricted family of observables. In Appendix C, we extend this result and show that explicit models can also approximate any *computable* (classical or quantum) hypothesis class.

These findings bring us to the essential question of whether unrestricting the family of observables of an explicit classifier, or turning them into implicit classifiers (that is, "kernelizing" a linear quantum model) can lead to a (systematically) better learning performance. This is the question we address in the remainder of this Letter.

RKHS and the representer theorem— Interestingly, a piece of functional analysis from learning theory gives us a way of characterizing any family of linear quantum models [25]. Namely, the so-called *reproducing kernel Hilbert space*, or RKHS [22], is the Hilbert space $\mathcal{H}$

spanned by all functions of the form $f(\boldsymbol{x}) = \langle \phi(\boldsymbol{x}), w \rangle_{\mathcal{F}}$, for all $w \in \mathcal{F}$. It includes any explicit and implicit models defined by the quantum feature states $\phi(\boldsymbol{x}) = \rho(\boldsymbol{x})$. From this point of view, a relaxation of any learning task using implicit or explicit models as a hypothesis family consists in finding the function in the RKHS $\mathcal{H}$ that has optimal learning performance. For the supervised learning task of modeling a target function $g(\boldsymbol{x})$ using a training set $\{(\boldsymbol{x}^{(1)}, g(\boldsymbol{x}^{(1)})), \ldots, (\boldsymbol{x}^{(M)}, g(\boldsymbol{x}^{(M)}))\}$, this learning performance is usually measured in terms of a training loss of the form, e.g.,

$$\widehat{\mathcal{L}}(f) = \frac{1}{M} \sum_{m=1}^{M} \Big(f(\boldsymbol{x}^{(m)}) - g(\boldsymbol{x}^{(m)}) \Big)^2 . \tag{6}$$

The true figure of merit of this problem however is in minimizing the *expected* loss $\mathcal{L}(f)$, defined similarly as a probability-weighted average over the entire data space $\mathcal{X}$. For this reason, a so-called regularization term $\lambda \|f\|_{\mathcal{H}}^2 = \lambda \|O\|_{\mathcal{F}}^2$ is often added to the training loss $\widehat{\mathcal{L}}_\lambda(f) = \widehat{\mathcal{L}}(f) + \lambda \|O\|_{\mathcal{F}}^2$ to incentivize the model not to overfit on the training data. Here, $\lambda \geq 0$ is a hyperparameter that controls the strength of this regularization.

Learning theory also allows us to characterize the linear models in $\mathcal{H}$ that are optimal with respect to the regularized training loss $\widehat{\mathcal{L}}_\lambda(f)$, for any $\lambda \geq 0$. Specifically, the *representer theorem* [22] states that the model $f_{\text{opt}} \in \mathcal{H}$ minimizing $\widehat{\mathcal{L}}_\lambda(f)$ is always a kernel model of the form of Eq. (2) (see Appendix A for a formal statement). A direct corollary of this result is that implicit quantum classifiers are guaranteed to achieve a lower (or equal) regularized training loss than any explicit quantum classifier using the same feature encoding [25]. Moreover, the optimal weights α_m of this model can be computed using $\mathcal{O}(M^2)$ evaluations of inner products on a quantum computer (that is, by estimating the expectation value in Fig. 1.b for all pairs of training points[2]) and with classical post-processing in time $\mathcal{O}(M^3)$ using, e.g., ridge regression or support vector machines [22]. This result may be construed to suggest that, in our study of quantum machine learning models, we only need to worry about implicit models, where the only real question is what feature encoding circuit we use to compute the kernel, and all machine learning is otherwise classical.

In previous works, it was noted that explicit models may nonetheless have advantages in terms of efficiency of training, as the optimization of their variational parameters is expected to terminate in $\mathcal{O}(M)$ optimization steps and would therefore only cost $\mathcal{O}(M)$ inner product evaluations [25]. Here, we show that the value of explicit models is much more fundamental. We start our discussion with an illustrative example clarifying the implications of the representer theorem.

[2] For this work, we ignore the required precision of the estimations. We note however that these can require exponentially many measurements, both in explicit [38] and implicit [27] models.

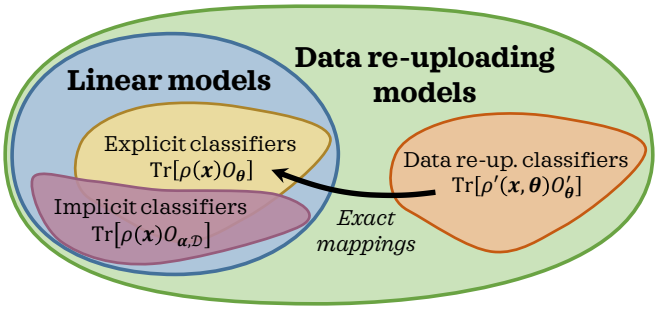

FIG. 4. Visualization of the model families studied in this work. Our mappings show that a family of data re-uploading classifiers can be mapped exactly to an equivalent family of explicit classifiers. Kernelizing the explicit model corresponds to turning its observables into linear combinations of feature states $\rho(\boldsymbol{x})$ for $\boldsymbol{x}$ in a dataset $\mathcal{D}$. As the size of $\mathcal{D}$ grows, the implicit model family covers more linear models in the RKHS associated to $\rho(\cdot)$.

Explicit can outperform implicit classifiers— We turn our attention back to the explicit classifiers constructed in Fig. 2. Note that the kernel function associated to the bit-string encodings is trivially

$$k(\boldsymbol{x}, \boldsymbol{x'}) = \prod_{i=1}^{d} |\langle \widetilde{\boldsymbol{x}}_i | \widetilde{\boldsymbol{x}}'_i \rangle|^2 = \delta_{\widetilde{\boldsymbol{x}}, \widetilde{\boldsymbol{x}}'}, \tag{7}$$

that is, the Kronecker delta function of the bit-strings $\widetilde{\boldsymbol{x}}$ and $\widetilde{\boldsymbol{x}}'$. Let us emphasize that, for an appropriate precision ε of encoding input vectors $\boldsymbol{x}$, the family of explicit classifiers resulting from our construction includes good approximations of *virtually any* parametrized quantum circuit model acting on n qubits. Yet, all of these result in the same kernel function of Eq. (7). This is a rather surprising result, for two reasons. First, this kernel is classically computable, which, in light of the representer theorem, seems to suggest that a simple classical classifier of the form of Eq. (2) can outperform *any* explicit quantum classifier stemming from our construction, and hence any quantum classifier in the limit $\varepsilon \to 0$. Second, this kernel classifier always takes the form

$$f_{\boldsymbol{\alpha},\mathcal{D}}(\boldsymbol{x}) = \sum_{m=1}^{M} \alpha_m \delta_{\widetilde{\boldsymbol{x}}, \widetilde{\boldsymbol{x}}^{(m)}}, \tag{8}$$

which is a model that overfits the training data and fails to generalize to unseen data points, as, for $\varepsilon \to 0$ and any choice of $\boldsymbol{\alpha}$, $f_{\boldsymbol{\alpha},\mathcal{D}}(\boldsymbol{x}) = 0$ for any $\boldsymbol{x}$ outside the training set. As we detail in Appendix B, similar observations can be made for the kernels resulting from our gate-teleportation construction.

These last remarks force us to rethink our interpretation of the representer theorem. When restricting our attention to the regularized training loss, implicit models do indeed lead to better training performance

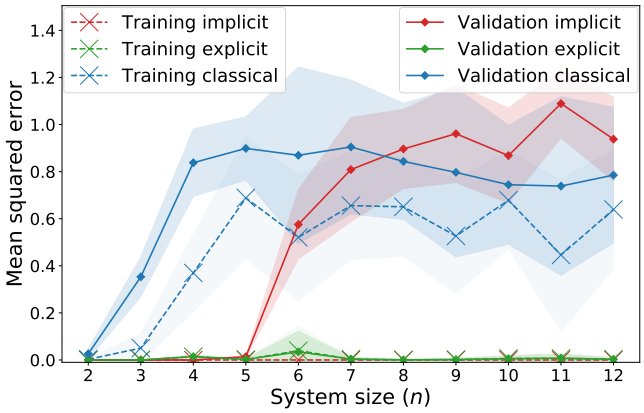

FIG. 5. Regression performance of explicit, implicit and classical models on an engineered learning task. For all system sizes, each model has access to a training set of $M = 1000$ pre-processed and re-labeled fashion-MNIST images. Validation loss is computed on a testing set of size 100. Shaded regions indicate the standard deviation over 10 labeling functions.

due to their increased expressivity.[3] But, as our construction shows, this expressivity can dramatically harm the generalization performance of the learning model, despite the use of regularization during training. Hence, restricting the set of observables accessible to a linear quantum classifier (or, equivalently, restricting the accessible manifold of the RKHS) can potentially provide a substantial learning advantage.

Numerical comparison— The illustrative example above showcases an extreme scenario where the "kernelization" of quantum classifiers systematically leads to poor generalization performance. It remains however an open question whether this phenomenon can also be observed in more practical use cases, e.g., for feature encodings and learning tasks previously proposed in quantum advantage experiments. To assess this hypothesis, we undertake a similar numerical study to that of Huang *et al.* [20], where we generate a learning task using explicit classifiers, and compare the performance of our quantum models as well as additional classical models on this task.

As to mimic a real-world data distribution, we consider input data from the fashion-MNIST dataset [39], composed of 28x28-dimensional images of clothing items. Using principal component analysis, we first reduce the dimension of these images to obtain n-dimensional vectors, for $2 \leq n \leq 12$. We then label the images using an explicit classifier acting on n qubits. For this, we use the feature encoding proposed by Havlíček *et al.* [23], which is conjectured to lead to classically intractable kernels, followed by a hardware-efficient variational unitary [4]. The expectation value of a Pauli Z observable on the first qubit then produces the data labels.[4] On this newly defined learning task, we test the performance of explicit classifiers from the same hypothesis family as the generating functions, and compare it to that of implicit classifiers using the same feature encoding, as well as a list of standard classical machine learning algorithms that are hyperparametrized for the task (see Appendix E). The results of this experiment are presented in Fig. 5.

The training losses we observe are consistent with our previous findings: the implicit models systematically achieve a lower training loss than their explicit counterparts (for an unregularized loss notably, implicit models achieve a training loss of 0). With respect to validation loss on the other hand, which is representative of the expected loss, we see a clear separation between implicit and explicit models starting from $n = 6$ qubits. This disparity between learning performances therefore showcases the advantage of a restricted observable family, when appropriately suited to the learning task. The limited size of the training set had little impact on the generalization performance of the explicit model, while this led the implicit model to overfit. Note that the addition of regularization to the training loss of the implicit model does not impact the separation we observe here (see Appendix F).

Discussion– In this Letter, we present a unifying framework for several quantum machine learning models by expressing them as linear models in quantum feature spaces. In particular, we show that data re-uploading circuits can be represented exactly by explicit linear models in larger feature spaces, which allows them to be reformulated as implicit kernel methods. Going beyond the advantages in training performance guaranteed by the representer theorem, we also illustrate how a systematic "kernelization" of explicit quantum models can be harmful in terms of generalization performance.

Our results suggest that the power of parametrized quantum circuits can lie not only in the way they encode data in quantum states, but also in a restricted variational processing. Reflecting this restriction on generalization bounds for quantum classifiers [36, 40] could help better characterize this learning advantage. Nonetheless, kernelizing a linear model also has its advantages, namely generalizing its hypothesis family and turning the learning task into a convex optimization problem. It remains an open question whether and when this increased expressivity can also be guaranteed to increase generalization performance.

We also observe that mappings from data re-uploading to explicit linear models are not unique, and that they lead to different feature encodings. While the choice of mapping (when exact) has no impact on the learning performance of the resulting explicit model, it does impact that of its kernelization. So far, all the mappings we

[3] On a classification task with labels $g(\boldsymbol{x}) = \pm 1$, the kernel classifier of Eq. (8) is optimal with respect to any regularized training loss for $\alpha_m = g(\boldsymbol{x}^{(m)})$ $\forall m$ such that $\widehat{\mathcal{L}}(f) = 0$ and $\|f\|^2_{\mathcal{H}} = M$.

[4] Note that we additionally normalize the labels as to obtain a standard deviation of 1 for all system sizes.

presented resulted in trivial classical kernels with poor generalization performance. But more interesting feature encodings may yet to be found, e.g., by imprinting more structure from the variational measurements onto them. In Appendix D, we show that, by allowing feature encodings to include tracing-out operations (making them non-unitary maps), a trivial unitary feature encoding can be used to generate non-trivial kernels. This observation is in line with recent findings about quantum kernels derived from non-unitary feature encodings [20, 27].

The results presented here broaden our perspectives on quantum machine learning and we believe will help us design quantum models with practical learning advantages.

CODE AVAILABILITY

The code used to run the numerical simulations, implemented using TensorFlow Quantum [41], is available at https://github.com/sjerbi/QML-beyond-kernel.

ACKNOWLEDGMENTS

The authors would like to thank Isaac D. Smith, Casper Gyurik, Matthias C. Caro, Elies Gil-Fuster, Ryan Sweke, and Maria Schuld for helpful discussions and comments, as well as Hsin-Yuan Huang for clarifications on their numerical simulations [20]. SJ, LJF, HPN and HJB acknowledge support from the Austrian Science Fund (FWF) through the projects DK-ALM:W1259-N27 and SFB BeyondC F7102. SJ also acknowledges the Austrian Academy of Sciences as a recipient of the DOC Fellowship. This work was in part supported by the Dutch Research Council (NWO/OCW), as part of the Quantum Software Consortium program (project number 024.003.037). VD acknowledges the support by the project NEASQC funded from the European Union's Horizon 2020 research and innovation programme (grant agreement No 951821). VD also acknowledges support through an unrestricted gift from Google Quantum AI.

[1] J. Preskill, Quantum computing in the nisq era and beyond, Quantum **2**, 79 (2018).
[2] M. Cerezo, A. Arrasmith, R. Babbush, S. C. Benjamin, S. Endo, K. Fujii, J. R. McClean, K. Mitarai, X. Yuan, L. Cincio, *et al.*, Variational quantum algorithms, arXiv:2012.09265 (2020).
[3] K. Bharti, A. Cervera-Lierta, T. H. Kyaw, T. Haug, S. Alperin-Lea, A. Anand, M. Degroote, H. Heimonen, J. S. Kottmann, T. Menke, *et al.*, Noisy intermediate-scale quantum (nisq) algorithms, arXiv:2101.08448 (2021).
[4] A. Peruzzo, J. McClean, P. Shadbolt, M.-H. Yung, X.-Q. Zhou, P. J. Love, A. Aspuru-Guzik, and J. L. O'brien, A variational eigenvalue solver on a photonic quantum processor, Nature communications **5**, 1 (2014).
[5] E. Farhi, J. Goldstone, and S. Gutmann, A quantum approximate optimization algorithm, arXiv:1411.4028 (2014).
[6] M. Benedetti, E. Lloyd, S. Sack, and M. Fiorentini, Parameterized quantum circuits as machine learning models, Quantum Science and Technology **4**, 043001 (2019).
[7] J. Carrasquilla and R. G. Melko, Machine learning phases of matter, Nature Physics **13**, 431 (2017).
[8] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko, *et al.*, Highly accurate protein structure prediction with alphafold, Nature **596**, 583 (2021).
[9] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, Quantum machine learning, Nature **549**, 195 (2017).
[10] V. Dunjko and H. J. Briegel, Machine learning & artificial intelligence in the quantum domain: a review of recent progress, Reports on Progress in Physics **81**, 074001 (2018).
[11] M. Schuld, A. Bocharov, K. M. Svore, and N. Wiebe, Circuit-centric quantum classifiers, Physical Review A **101**, 032308 (2020).
[12] E. Farhi and H. Neven, Classification with quantum neural networks on near term processors, arXiv:1802.06002 (2018).
[13] J.-G. Liu and L. Wang, Differentiable learning of quantum circuit born machines, Physical Review A **98**, 062324 (2018).
[14] D. Zhu, N. M. Linke, M. Benedetti, K. A. Landsman, N. H. Nguyen, C. H. Alderete, A. Perdomo-Ortiz, N. Korda, A. Garfoot, C. Brecque, *et al.*, Training of quantum circuits on a hybrid quantum computer, Science advances **5**, eaaw9918 (2019).
[15] A. Skolik, S. Jerbi, and V. Dunjko, Quantum agents in the gym: a variational quantum algorithm for deep q-learning, arXiv:2103.15084 (2021).
[16] S. Jerbi, C. Gyurik, S. Marshall, H. J. Briegel, and V. Dunjko, Variational quantum policies for reinforcement learning, arXiv:2103.05577 (2021).
[17] Y. Liu, S. Arunachalam, and K. Temme, A rigorous and robust quantum speed-up in supervised machine learning, Nature Physics , 1 (2021).
[18] Y. Du, M.-H. Hsieh, T. Liu, and D. Tao, Expressive power of parametrized quantum circuits, Physical Review Research **2**, 033125 (2020).
[19] R. Sweke, J.-P. Seifert, D. Hangleiter, and J. Eisert, On the quantum versus classical learnability of discrete distributions, Quantum **5**, 417 (2021).
[20] H.-Y. Huang, M. Broughton, M. Mohseni, R. Babbush, S. Boixo, H. Neven, and J. R. McClean, Power of data in quantum machine learning, Nature communications **12**, 1 (2021).
[21] H.-Y. Huang, R. Kueng, and J. Preskill, Information-theoretic bounds on quantum advantage in machine learning, Physical Review Letters **126**, 190505 (2021).
[22] B. Schölkopf, A. J. Smola, F. Bach, *et al.*, *Learning with kernels: support vector machines, regularization, optimization, and beyond* (MIT press, 2002).

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

[24] M. Schuld and N. Killoran, Quantum machine learning in feature hilbert spaces, Physical review letters **122**, 040504 (2019).

[25] M. Schuld, Supervised quantum machine learning models are kernel methods, arXiv:2101.11020 (2021).

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

[27] J. M. Kübler, S. Buchholz, and B. Schölkopf, The inductive bias of quantum kernels, arXiv:2106.03747 (2021).

[28] E. Peters, J. Caldeira, A. Ho, S. Leichenauer, M. Mohseni, H. Neven, P. Spentzouris, D. Strain, and G. N. Perdue, Machine learning of high dimensional data on a noisy quantum processor, arXiv:2101.09581 (2021).

[29] T. Haug, C. N. Self, and M. Kim, Large-scale quantum machine learning, arXiv:2108.01039 (2021).

[30] K. Bartkiewicz, C. Gneiting, A. Černoch, K. Jiráková, K. Lemr, and F. Nori, Experimental kernel-based quantum machine learning in finite feature space, Scientific Reports **10**, 1 (2020).

[31] T. Kusumoto, K. Mitarai, K. Fujii, M. Kitagawa, and M. Negoro, Experimental quantum kernel trick with nuclear spins in a solid, npj Quantum Information **7**, 1 (2021).

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

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

[34] A. Pérez-Salinas, D. López-Núñez, A. García-Sáez, P. Forn-Díaz, and J. I. Latorre, One qubit as a universal approximant, Physical Review A **104**, 012405 (2021).

[35] T. Goto, Q. H. Tran, and K. Nakajima, Universal approximation property of quantum machine learning models in quantum-enhanced feature spaces, Phys. Rev. Lett. **127**, 090506 (2021).

[36] C. Gyurik, D. van Vreumingen, and V. Dunjko, Structural risk minimization for quantum linear classifiers, arXiv:2105.05566 (2021).

[37] H. J. Briegel, D. E. Browne, W. Dür, R. Raussendorf, and M. Van den Nest, Measurement-based quantum computation, Nature Physics **5**, 19 (2009).

[38] J. R. McClean, S. Boixo, V. N. Smelyanskiy, R. Babbush, and H. Neven, Barren plateaus in quantum neural network training landscapes, Nature communications **9**, 1 (2018).

[39] H. Xiao, K. Rasul, and R. Vollgraf, Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, arXiv:1708.07747 (2017).

[40] M. C. Caro, E. Gil-Fuster, J. J. Meyer, J. Eisert, and R. Sweke, Encoding-dependent generalization bounds for parametrized quantum circuits, arXiv:2106.03880 (2021).

[41] M. Broughton, G. Verdon, T. McCourt, A. J. Martinez, J. H. Yoo, S. V. Isakov, P. Massey, M. Y. Niu, R. Halavati, E. Peters, *et al.*, Tensorflow quantum: A software framework for quantum machine learning, arXiv:2003.02989 (2020).

[42] M. A. Nielsen and I. Chuang, *Quantum computation and quantum information* (American Association of Physics Teachers, 2002).

[43] S. Bravyi and D. Gosset, Improved classical simulation of quantum circuits dominated by clifford gates, Physical review letters **116**, 250501 (2016).

[44] M. A. Nielsen and I. L. Chuang, Programmable Quantum Gate Arrays, Physical Review Letters **79**, 321 (1997).

[45] Y. Yang, R. Renner, and G. Chiribella, Optimal Universal Programming of Unitary Gates, Physical Review Letters **125**, 210501 (2020).

[46] G. Brassard, P. Hoyer, M. Mosca, and A. Tapp, Quantum amplitude amplification and estimation, Contemporary Mathematics **305**, 53 (2002).

[47] D. P. Kingma and J. Ba, Adam: A method for stochastic optimization, arXiv:1412.6980 (2014).

[48] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, *et al.*, Scikit-learn: Machine learning in python, the Journal of machine Learning research **12**, 2825 (2011).

Appendix A: Representer theorem

In this appendix, we give a formal statement of the representer theorem from learning theory.

Theorem A.1 (Representer theorem [22]) *Let $g : \mathcal{X} \to \mathcal{Y}$ be a target function with input and output domains $\mathcal{X}$ and $\mathcal{Y}$, $\mathcal{D} = \{(\boldsymbol{x}^{(1)}, g(\boldsymbol{x}^{(1)}), \ldots, (\boldsymbol{x}^{(M)}, g(\boldsymbol{x}^{(M)})\}$ a training set of size M, and $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ a kernel function with a corresponding reproducing kernel Hilbert space (RKHS) $\mathcal{H}$. For any strictly monotonic increasing regularization function $h : [0, \infty) \to \mathbb{R}$ and any loss function $\mathcal{L} : \mathcal{X} \times \mathcal{Y} \times \mathbb{R} \to \mathbb{R} \cup \infty$, we have that any minimizer of the regularized training loss, from the RKHS,*

$$f_{\text{opt}} = \underset{f \in \mathcal{H}}{\operatorname{argmin}} \left\{ \widehat{\mathcal{L}}(f) + h(\|f\|^2_{\mathcal{H}}) \right\}$$

admits a representation of the form

$$f_{\text{opt}}(\boldsymbol{x}) = \sum_{m=1}^{M} \alpha_m k(\boldsymbol{x}, \boldsymbol{x}^{(m)})$$

where $\alpha_m \in \mathbb{R}$ for all $1 \leq m \leq M$.

A common choice for the regularization function is simply $h(\|f\|^2_{\mathcal{H}}) = \lambda \|f\|^2_{\mathcal{H}}$, where $\lambda \geq 0$ is a hyperparameter adjusting the strength of the regularization.

Appendix B: Mappings from data re-uploading to explicit classifiers

In this section, we detail possible mappings from data re-uploading classifiers to explicit classifiers. We focus here on mappings relying on measurement-based quantum computation [37].

1. Mappings based on gate teleportation

In this approach, we restrict our attention to encoding gates of the form $e^{-\mathrm{i}h(\boldsymbol{x})H_n/2}$, for H_n an arbitrary Pauli string acting on n qubits, e.g., $H_3 = X \otimes Z \otimes I$, and $h : \mathbb{R}^d \to \mathbb{R}$ an arbitrary function mapping real-valued input vectors $\boldsymbol{x}$ to rotation angles $h(\boldsymbol{x})$. Using known techniques (see Sec. 4.7.3 in [42]), we can show that in order to implement any such gate $e^{-\mathrm{i}h(\boldsymbol{x})H_n/2}$ *exactly*, one only needs to perform a Pauli-Z rotation $e^{-\mathrm{i}h(\boldsymbol{x})Z/2}$ on a single of these n qubits, along with $\mathcal{O}(n)$ operations that are independent of $\boldsymbol{x}$. Therefore, in our gate-teleportation scheme, we only need to focus on teleporting gates of the form $R_z(h(\boldsymbol{x})) = e^{-\mathrm{i}h(\boldsymbol{x})Z/2}$.

This gate teleportation can easily be implemented using the gadget depicted in Fig. 6. It is easy to check that for an arbitrary input qubit $|\psi\rangle = \alpha |0\rangle + \beta |1\rangle$ and $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$, the state generated by this gadget

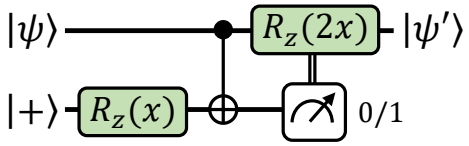

FIG. 6. A gate-teleportation gadget (inspired by the T-gadget in [43]) that implements the unitary $|\psi\rangle \mapsto |\psi'\rangle = R_z(x) |\psi\rangle$.

before the computational basis measurement (and correction) is:

$$\begin{aligned} &\frac{1}{\sqrt{2}}(\alpha |0\rangle + \beta e^{\mathrm{i}x} |1\rangle) \otimes |0\rangle + \frac{1}{\sqrt{2}}(\alpha e^{\mathrm{i}x} |0\rangle + \beta |1\rangle) \otimes |1\rangle \\ &\quad = \frac{1}{\sqrt{2}} \left(R_z(x) |\psi\rangle \otimes |0\rangle + e^{\mathrm{i}x} R_z(-x) |\psi\rangle \otimes |1\rangle \right) \end{aligned} \tag{B1}$$

which results in the correct outcome $|\psi'\rangle = R_z(x) |\psi\rangle$ for a $|0\rangle$ measurement, and a state that can be corrected (up to a global phase) via a $R_z(2x)$ rotation, in case of a $|1\rangle$ measurement.

Putting aside the corrections required by this gadget, we note the interesting property that, when used to simulate every encoding gate in the data re-uploading circuit, this gadget moves all data-dependent parts of the circuit on additional ancilla qubits, essentially turning it into an explicit model. However, this gadget still requires data-dependent corrections (of the form $R_z(2h(\boldsymbol{x}))$) in the case of $|1\rangle$ measurement outcomes, which happen with probability $1/2$ for each gate teleportation. To get around this problem, we simply replace the computational basis measurement in the gadget by a projection $P_0 = |0\rangle\langle 0|$ on the $|0\rangle$ state. While these projections cannot be implemented deterministically in practice, the resulting model is still a valid explicit model, in the sense that including the projections $P_0^{\otimes D}$ in the observable $O_{\boldsymbol{\theta}}$, for D uses of our gadget, still leads to a valid observable $O'_{\boldsymbol{\theta}} = O_{\boldsymbol{\theta}} \otimes P_0^{\otimes D}$.

However, given that each of these projections does not account for the re-normalization of the resulting quantum state (i.e., the factor $1/\sqrt{2}$ in Eq. (B1)), this means that, in order to enforce $\mathrm{Tr}[\rho'(\boldsymbol{x}) O'_{\boldsymbol{\theta}}] = \mathrm{Tr}[\rho(\boldsymbol{x}, \boldsymbol{\theta}) O_{\boldsymbol{\theta}}]$, $\forall \boldsymbol{x}, \boldsymbol{\theta}$, we need to multiply $O'_{\boldsymbol{\theta}}$ by a factor of $2^{D/2}$. This implies that the evaluation of the resulting explicit model is exponentially harder than that of the original data re-uploading model, in the number of encoding gates D. As we show next, this factor can however be made arbitrarily close to 1, by allowing each encoding gate/angle to be used more than once in the feature encoding.

To achieve this feat, we transform our previous gadget as to implement its data-dependent corrections using gate teleportation again. A single such nested use of gate teleportation now has probability $1 - 1/4 = 3/4$ of succeeding without corrections, as opposed to the probability $1/2$ of the previous gadget. For N nested uses (see Fig. 7), the success probability is then boosted to $1 - 2^{-N}$, which can be made arbitrarily close to 1. If we use this nested gadget for all D encoding gates in the circuit, the probability that all of them are implemented

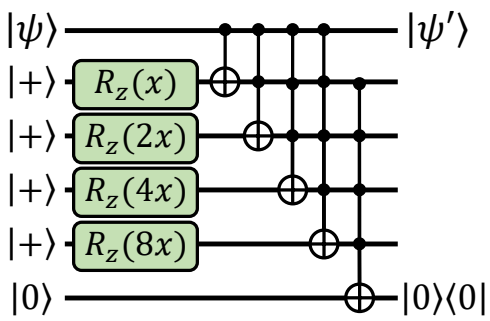

FIG. 7. A gate-teleportation gadget without data-dependent corrections and success probability $1-2^{-4}\approx 0.94$. The last qubit acts as a witness that at least one of the nested gate teleportations succeeded without the need for corrections.

successfully without corrections is then $p=(1-2^{-N})^D$. This probability p can be made larger than $1-\delta'$, for any $\delta'>0$, by choosing $N=\lceil \log_2\left((1-\sqrt[D]{1-\delta'})^{-1}\right)\rceil \leq \lceil \log_2(D/\delta')\rceil$. This also leads to a normalization factor $\sqrt{p^{-1}} \leq \sqrt{(1-\delta')^{-1}}$, which can be made arbitrarily close to 1. Note as well that this normalization factor is always known exactly, such that we can guarantee $\mathrm{Tr}[\rho'(\boldsymbol{x})O'_{\boldsymbol{\theta}}]=\mathrm{Tr}[\rho(\boldsymbol{x},\boldsymbol{\theta})O_{\boldsymbol{\theta}}],\ \forall \boldsymbol{x},\boldsymbol{\theta}$.

2. Kernels resulting from our mappings

In the main text, we showed how our illustrative mapping based on bit-string encodings of $\boldsymbol{x}$ resulted in trivial Kronecker-delta kernel functions and implicit models with very poor generalization performance. In this subsection, we derive a similar result for our gate-teleportation mappings.

We first note that these mappings lead to feature encodings of the form:

$$U_\phi(\boldsymbol{x})\left|0^{\otimes n+ND+D}\right\rangle = \left|0^{\otimes n+D}\right\rangle \bigotimes_{\substack{1\leq i\leq D\\ 1\leq j\leq N}} R_z(2^{j-1}h_i(\boldsymbol{x}))\left|+\right\rangle$$

for D encoding gates with encoding angles $h_i(\boldsymbol{x})$, and using N nested gate teleportations for each of these gates. While less generic than the feature states resulting from our binary encodings, these still generate kernels

$$k(\boldsymbol{x},\boldsymbol{x'})=\prod_{i,j}\cos\left(2^{j-1}(h_i(\boldsymbol{x})-h_i(\boldsymbol{x'}))\right)^2 \tag{B2}$$

that are again classically simulatable and $k(\boldsymbol{x},\boldsymbol{x'})\to\delta_{\boldsymbol{x},\boldsymbol{x'}}$ for $ND\to\infty$.

Moreover, in the case where the angles $h_i(\boldsymbol{x})$ are linear functions of components x_i of $\boldsymbol{x}$, we can directly apply Theorem 1 of [27] to show the following. For a number of encoding gates D and a number of nested gate teleportations N large enough (i.e., ND larger than some d_0), and for a dataset that is at most polynomially large in ND, no function can be learned using the implicit model resulting from this kernel. Note that, for this theorem to be applicable, we also need to assume non-degenerate data distributions μ (i.e., that do not have support on single data points) that are separable on all components x_i of $\boldsymbol{x}$, i.e., $\mu=\bigotimes_i \mu_i$, such that the mean embeddings $\rho_{\mu_i}=\int \rho_i(\boldsymbol{x})\mu_i(d\boldsymbol{x})$ for each component x_i are all mixed.

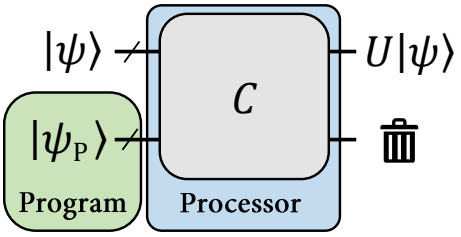

FIG. 8. A programmable quantum processor. A state $|\psi_\mathrm{P}\rangle$ is fed to the processor as a program instructing the processor to implement a unitary map U on another input state. The no-programming theorem states that programmable processors $\mathcal{C}$ capable of implementing any unitary map U cannot exist.

3. Link to no-programming

It may seem to the informed reader as though our mappings from data re-uploading to explicit models are in violation of the so-called no-programming theorem from quantum information theory. In this section, we will shortly outline this theorem and explain why our mappings do not violate it.

A programmable quantum processor is defined as a CPTP map $\mathcal{C}:\mathcal{H}_\mathrm{S}\otimes\mathcal{H}_\mathrm{P}\to\mathcal{H}_\mathrm{S}$, where $\mathcal{H}_\mathrm{S}$ and $\mathcal{H}_\mathrm{P}$ denote the system and program Hilbert spaces. The purpose of such a quantum processor is to implement unitary maps $U:\mathcal{H}_\mathrm{S}\to\mathcal{H}_\mathrm{S}$ where the information about U is fed to the processor solely by a program state $|\psi_\mathrm{P}\rangle$ (see Fig. 8). The no-programming theorem [44, 45] rules out the existence of perfect universal quantum processors, in the sense that there cannot exist a processor $\mathcal{C}$ that can implement infinitely many different unitary maps U deterministically using finite-dimensional program states $|\psi_\mathrm{P}\rangle$.

The explicit models resulting from our mappings have properties that are reminiscent of quantum processors. In Fig. 2 for instance, the bit-string encodings $|\widetilde{\boldsymbol{x}}\rangle$ are used to implement data-encoding unitaries in an otherwise data-independent circuit. Thus, one may interpret these as the program states $|\psi(\boldsymbol{x})\rangle$ of a quantum processor $\mathcal{C}$ given by the rest of the circuit. Same goes for the quantum states $R_z(x_2)\left|+\right\rangle R_z(x_3)\left|+\right\rangle$ in Fig. 3.

In light of the no-programming theorem, it is quite remarkable that these explicit models can "program" a *continuous* set of unitaries $\{U(\boldsymbol{x},\boldsymbol{\theta})=\prod_\ell V_\ell(\boldsymbol{\theta})U_\ell(\boldsymbol{x})\}_{\boldsymbol{x}\in\mathbb{R}^d}$ (and particularly the unitaries $U_\ell(\boldsymbol{x})$ for $\ell\geq 2$). Note however that, in the case of our bit-string mappings, these unitaries are only implemented *approximately* and that, in our gate-teleportation mappings, they are only implemented *probabilistically*. Our gate-teleportation mappings are only exact from the point of view of *models*, i.e., expectation values of observables, which are not

covered by no-programming. The approximation errors $\delta > 0$ and the normalization factors $(1-\delta')^{-1} \neq 1$ that we obtain in our mappings are indeed symptomatic of our inability to program data re-uploading unitaries both exactly and deterministically. On the other hand, our results show that, contrary to unitary maps, expectation values can be "programmed" exactly.

Appendix C: Explicit models are universal function (family) approximators

From the universality of data re-uploading models as function approximators [33, 34] and our exact mappings from data re-uploading to explicit models, it trivially derives that explicit models are also universal function approximators. That is, for any integrable function $g \in \mathrm{L}([0,2\pi]^d)$ with a finite number of discontinuities, and for any $\varepsilon > 0$, there exists an n-qubit feature encoding of the form $U_\phi(\boldsymbol{x}) = I \bigotimes_{i,j} R_z(2^j x_i) H$ and an observable O such that $|\mathrm{Tr}[\rho(\boldsymbol{x})O] - g(\boldsymbol{x})| \leq \varepsilon, \forall \boldsymbol{x}$, for $\rho(\boldsymbol{x}) = U_\phi(\boldsymbol{x}) |\boldsymbol{0}\rangle\langle\boldsymbol{0}| U_\phi^\dagger(\boldsymbol{x})$ (this result derives specifically from Theorem 2 in [34]). A similar result was independently obtained in Ref. [35].

In this section, we show that this universal approximation property of explicit classifiers also applies to computable hypothesis classes, i.e., function families of a known classical or quantum model. More precisely, we show that for any family $\{g_{\boldsymbol{\theta}}\}_{\boldsymbol{\theta}}$ of *computable* functions $g_{\boldsymbol{\theta}} : [0,2\pi]^d \to \mathbb{R}$ specified by a Boolean or quantum circuit parametrized by a vector $\boldsymbol{\theta} \in [0,2\pi]^L$, and for any $\varepsilon > 0$, there exists an n-qubit feature encoding $U_\phi(\boldsymbol{x})$ using single-qubit rotations of the form $R_y(x_i)$ to encode $\boldsymbol{x}$, and a family of observables $O_{\boldsymbol{\theta}}$ parametrized by single-qubit rotations of the form $R_y(\theta_i)$, such that $|\mathrm{Tr}[\rho(\boldsymbol{x})O_{\boldsymbol{\theta}}] - g_{\boldsymbol{\theta}}(\boldsymbol{x})| \leq \varepsilon, \forall \boldsymbol{x}, \boldsymbol{\theta}$.

The proof of this result is rather simple. Using quantum amplitude estimation [46] as a subroutine, we can, starting from a $|\boldsymbol{0}\rangle$ state and using $R_y(x_i)$ rotations, create a bit-string representation $|\widetilde{\boldsymbol{x}}\rangle$ of $\boldsymbol{x}$. This constitutes the feature encoding $U_\phi(\boldsymbol{x})$. On a different register, we create similarly a bit-string representation $|\widetilde{\boldsymbol{\theta}}\rangle$ of $\boldsymbol{\theta}$ using $R_y(\theta_i)$ rotations. These bit-strings, for an appropriately chosen precision of representation (which depends on the number of $R_y(x_i), R_y(\theta_i)$ rotations used), can then be used to approximate any computable function $g_{\boldsymbol{\theta}}(\boldsymbol{x})$ to some error ε.[5] Indeed, when $g_{\boldsymbol{\theta}}$ is computed via a parametrized quantum circuit, we can use a similar construction to that depicted in Fig. 2 in the main text. When $g_{\boldsymbol{\theta}}$ is instead computed classically (e.g., using a neural network), we can either simulate this computation with a quantum circuit (see Sec. 3.2.5 of [42]), or simply include it in the observables $O_{\boldsymbol{\theta}}$ as a post-processing of a computational basis measurement of $|\widetilde{\boldsymbol{x}}\rangle$ and $|\widetilde{\boldsymbol{\theta}}\rangle$.

[5] Note that, for functions $g_{\boldsymbol{\theta}}$ that are computed using binary representations of $\boldsymbol{x}$ and $\boldsymbol{\theta}$, this construction can be made exact.

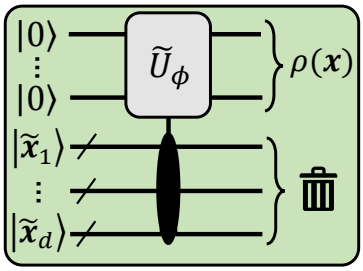

FIG. 9. A CPTP feature map based on bit-string encodings. Using data-independent controlled rotations, we can implement an approximation of any unitary feature encoding $U_\phi(\boldsymbol{x})$ by further tracing out the bit-string registers.

While the explicit classifiers constructed in this proof are quite unnatural, they showcase how powerful explicit quantum classifiers can be, when parametrized rotations are used to encode input data and as variational gates.

Appendix D: Beyond unitary feature encodings

So far, in our definition of linear quantum classifiers, we only considered *unitary* feature encodings, i.e., where feature states are defined as $\rho(\boldsymbol{x}) = U_\phi(\boldsymbol{x}) |\boldsymbol{0}\rangle\langle\boldsymbol{0}| U_\phi^\dagger(\boldsymbol{x})$ for a certain unitary map $U_\phi(\boldsymbol{x})$. In this section, we make the case that more general feature encodings, namely encodings for which the map $U_\phi(\boldsymbol{x})$ is allowed to be any completely positive trace preserving (CPTP) map, can lead to more interesting kernels $k(\boldsymbol{x}, \boldsymbol{x'}) = \mathrm{Tr}[\rho(\boldsymbol{x})\rho(\boldsymbol{x'})]$.

We illustrate this point by focusing on the bit-string feature encoding $U_\phi(\boldsymbol{x}) \left|0^{\otimes n+dp}\right\rangle = |0^{\otimes n}\rangle \bigotimes_{i=1}^d |\widetilde{\boldsymbol{x}}_i\rangle$ that we presented in the main text. We start by noting that augmenting this feature encoding with an arbitrary unitary V always leads to the same kernel function:

$$\begin{aligned} k(\boldsymbol{x}, \boldsymbol{x'}) &= \left| \langle \boldsymbol{0}| U_\phi^\dagger(\boldsymbol{x'}) V^\dagger V U_\phi(\boldsymbol{x}) |\boldsymbol{0}\rangle \right|^2 \\ &= \left| \langle \boldsymbol{0}| U_\phi^\dagger(\boldsymbol{x'}) U_\phi(\boldsymbol{x}) |\boldsymbol{0}\rangle \right|^2 \end{aligned}$$

given that $V^\dagger V = I$. If we however allow for a non-unitary operation such as tracing out part of the quantum system (which is allowed by CPTP maps), we can use this bit-string encoding to construct kernels $k(\boldsymbol{x}, \boldsymbol{x'})$ that approximate *virtually any* quantum kernel resulting from a unitary feature encoding on n qubits. To see this, suppose for instance that we want to approximate the quantum kernel proposed by Havlíček *et al.* [23] (resulting from the unitary feature encoding of Eq. (E1)). In this case, we can, similarly to our construction in Fig. 2, use data-independent rotations, controlled by the bit-string registers and acting on the n working qubits, to simulate the data-dependent gates of Eq. (E1). Then, by tracing out the bit-string register, we effectively obtain an (arbitrarily good) approximation of the original feature encoding on the working qubits. This CPTP feature encoding is depicted in Fig. 9.

Appendix E: Details of the numerical simulations

In this section, we provide more details on the numerical simulations presented in the main text. We first describe how the training and validation data of the learning task are generated, then specify the quantum and classical models that we trained on this task.

1. Dataset generation

a. *Generating data points*

We generate our training and validation data by preprocessing the fashion MNIST dataset [39]. All 28×28-pixels grayscale images in the dataset are first subject to a dimensionality reduction via principal component analysis (PCA), where only their n principal components are preserved, $2 \leq n \leq 12$. This PCA gives rise to data vectors $\boldsymbol{x} \in \mathbb{R}^n$ that are further normalized componentwise as to each be centered around 0 and have a standard deviation of 1. To create a training set, we sample $M = 1000$ of these vectors without replacement. The validation set, of size $M' = 100$, is sampled similarly from the (pre-processed) fashion MNIST testing data.

b. *Generating labels*

The labels $g(\boldsymbol{x})$ of the data points $\boldsymbol{x}$ in the training and validation sets are generated using the explicit classifiers depicted in Fig. 10, for a number of qubits n equal to the dimension of $\boldsymbol{x}$, and uniformly random parameters $\boldsymbol{\theta} \in [0, 2\pi]^{3nL}$. The feature encoding takes the form [23]:

$$U_\phi(\boldsymbol{x}) \left|0^{\otimes n}\right\rangle = U_z(\boldsymbol{x}) H^{\otimes n} U_z(\boldsymbol{x}) H^{\otimes n} \left|0^{\otimes n}\right\rangle \tag{E1}$$

for

$$U_z(\boldsymbol{x}) = \exp\left(-\mathrm{i}\pi \left[\sum_{i=1}^{n} x_i Z_i + \sum_{\substack{j=1,\\ j>i}}^{n} x_i x_j Z_i Z_j \right] \right). \tag{E2}$$

As for the variational unitaries $V(\boldsymbol{\theta})$, these are composed of L layers of single-qubit rotations $R(\boldsymbol{\theta}_{i,j})$ on each of the qubits, interlaid with $CZ = |1\rangle\langle 1| \otimes Z$ gates between nearest neighbours in the circuit. We choose the number of layers L as a function of the number of qubits n in the circuit, such that the number of parameters $(3nL)$ is approximately 90 at all system sizes (except for $n = 5$ to 8, where this led to a poorer performance of the explicit models). Specifically, for $n = 2$ to 12, we have $L = 15, 10, 7, 4, 3, 3, 3, 3, 3, 3, 2$, respectively.

Finally, the labels of the data points are specified by the expectations values

$$g(\boldsymbol{x}) = w_{\mathcal{D},\boldsymbol{\theta}} \mathrm{Tr}[\rho(\boldsymbol{x}) V(\boldsymbol{\theta})^\dagger Z_1 V(\boldsymbol{\theta})] \tag{E3}$$

where $w_{\mathcal{D},\boldsymbol{\theta}}$ is a re-normalization factor that sets the standard deviation of these labels to 1 over the training set.

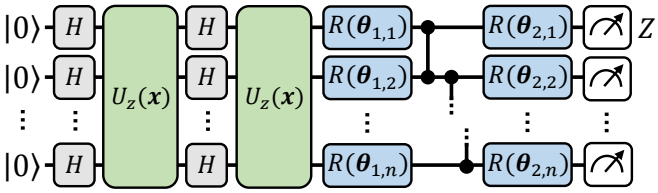

FIG. 10. The explicit model used in our numerical simulations. We use the feature encoding proposed by Havlíček *et al.* [23] (see Eq. (E1)), followed by a hardware-efficient variational circuit, where arbitrary single-qubit rotations $R(\boldsymbol{\theta}_{i,j}) = R_x(\theta_{i,j,0}) R_y(\theta_{i,j,1}) R_z(\theta_{i,j,2})$ on each qubit are interlaid with nearest-neighbour CZ gates, for L layers (here $L = 2$). Finally, the expectation value of a Z_1 observable (with a renormalization) assigns labels to input data $\boldsymbol{x} \in \mathbb{R}^n$.

c. *Evaluating performance*

We evaluate the training loss of a hypothesis function f using the mean squared error

$$\widehat{\mathcal{L}}(f) = \frac{1}{M} \sum_{m=1}^{M} \left(f(\boldsymbol{x}^{(m)}) - g(\boldsymbol{x}^{(m)}) \right)^2 \tag{E4}$$

on the training data. The validation loss (indicative of the expected loss) is evaluated similarly on the validation data (of size M').

2. Quantum machine learning models

In our simulations, we compare the performance of two types of quantum machine learning models.

First, we consider explicit models from the same variational family as those used to label the data (i.e., depicted in Fig. 10), but initialized with *different* variational parameters $\boldsymbol{\theta}$, also uniformly sampled from $[0, 2\pi]^{3nL}$. As opposed to the generating functions of Eq. (E3), we replace the observable weight $w_{\mathcal{D},\boldsymbol{\theta}}$ by a free parameter w, initialized to 1 and trained along the variational parameters $\boldsymbol{\theta}$. We train the explicit models for 500 steps of gradient descent on the training loss of Eq. (E4). We also use an ADAM optimizer [47] with a learning rate $\alpha_{\boldsymbol{\theta}} = 0.01$ for the variational parameters $\boldsymbol{\theta}$ and a learning rate $\alpha_w = 0.1$ for the observable weight w.

Second, we also consider implicit models that rely on the same feature encoding $U_\phi(\boldsymbol{x})$ (Eq. (E1)) as the explicit models. I.e., these take the form

$$f_{\boldsymbol{\alpha},\mathcal{D}}(\boldsymbol{x}) = \mathrm{Tr}[\rho(\boldsymbol{x}) O_{\boldsymbol{\alpha},\mathcal{D}}] \tag{E5}$$

for the same encodings $\rho(\boldsymbol{x}) = U_\phi(\boldsymbol{x}) |\boldsymbol{0}\rangle\langle\boldsymbol{0}| U_\phi^\dagger(\boldsymbol{x})$, and an observable $O_{\boldsymbol{\alpha},\mathcal{D}}$ given by Eq. (3) in the main text. We train their parameters $\boldsymbol{\alpha}$ using the KernelRidge regression package of scikit-learn [48]. In the numerical simulations of Fig. 5, we we use an unregularized training loss, i.e., that of Eq. (E4). The learning performance of the implicit models trained with regularization is presented in Appendix F.

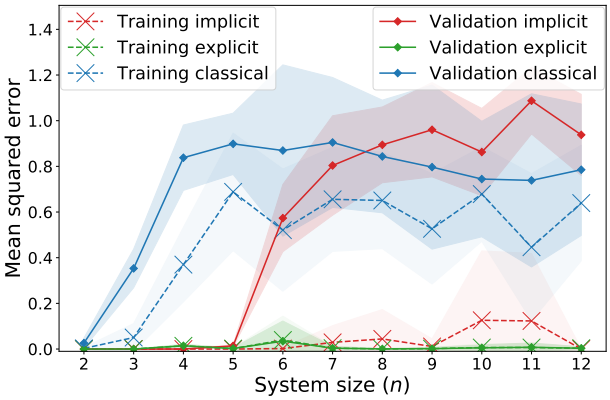

FIG. 11. Best performance of implicit classifiers for different regularization strengths.

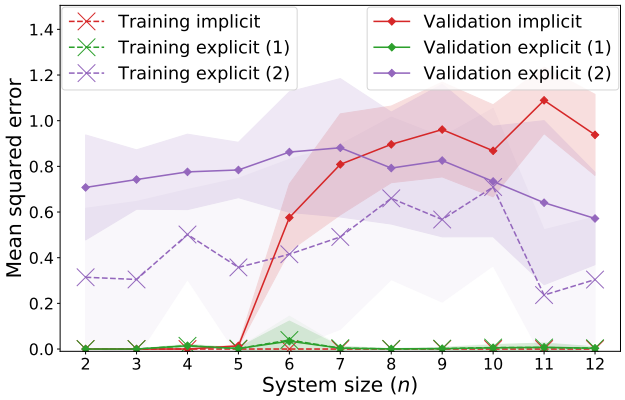

FIG. 12. Regression performance of explicit classifiers from the same variational family as the classifiers generating the data labels (1) and from a different variational family (2).

3. Classical machine learning models

We additionally compare the performance of our quantum machine learning models to a list of classical models, identical to that of Huang *et al.* [20]. For completeness, we list these models here, and the hyperparameters they were trained with. All of these models were trained using the default specifications of scikit-learn [48], unless stated otherwise.

- Neural network: We perform a grid search over two-layer feed-forward neural networks with hidden layers of size

$$h \in \{10, 25, 50, 75, 100, 125, 150, 200\}. \tag{E6}$$

  We use the MLPRegressor package with a maximum number of learning steps max_iter = 500.

- Linear kernel method: We perform a grid search over the regularization parameter

$$C \in \{0.006, 0.015, 0.03, 0.0625, 0.125, 0.25, 0.5, 1.0, 2.0, 4.0, 8.0, 16.0, 32.0, 64.0, 128.0, 256, 512, 1024\}.$$

  We select the best performance between the SVR and KernelRidge packages (both using the linear kernel).

- Gaussian kernel method: We perform a grid search over the regularization parameter

$$C \in \{0.006, 0.015, 0.03, 0.0625, 0.125, 0.25, 0.5, 1.0, 2.0, 4.0, 8.0, 16.0, 32.0, 64.0, 128.0, 256, 512, 1024\},$$

  and the RBF kernel hyperparameter

$$\gamma \in \{0.25, 0.5, 1.0, 2.0, 3.0, 4.0, 5.0, 20.0\}/(n\mathrm{Var}[\boldsymbol{x}]),$$

  where $\mathrm{Var}[\boldsymbol{x}]$ is the variance of all the components x_i, for all the data points $\boldsymbol{x}$ in the training set.
  We select the best performance between the SVR and KernelRidge packages (both using the RBF kernel).

- Random forest: We perform a grid search over the individual tree depth

$$\text{max\_depth} \in \{2, 3, 4, 5\},$$

  and the number of trees

$$\text{n\_trees} \in \{25, 50, 100, 200, 500\}.$$

  We use the RandomForestRegressor package.

- Gradient boosting: We perform a grid search over the individual tree depth

$$\text{max\_depth} \in \{2, 3, 4, 5\},$$

  and the number of trees

$$\text{n\_trees} \in \{25, 50, 100, 200, 500\}.$$

  We use the GradientBoostingRegressor package.

- Adaboost: We perform a grid search over the number of estimators

$$\text{n\_estimators} \in \{25, 50, 100, 200, 500\}.$$

  We use the AdaBoostRegressor package.

At each system size, we keep the learning performance of the model with the lowest validation loss.

Appendix F: Additional numerical simulations

In this section, we defer the results of our additional numerical simulations, in support of the claims made in the main text.

We first show that regularization does not improve the learning performance of the implicit models. In Fig. 11 we plot the best validation losses we obtained using regularization strengths $\lambda = 1/(2C)$ for $C \in \{0.006, 0.015, 0.03, 0.0625, 0.125, 0.25, 0.5, 1.0, 2.0, 4.0, 8.0, 16.0, 32.0, 64.0, 128.0, 256, 512, 1024\}$.

We also show that a variational family of observables not suited for the learning task can lead to poor learning performance. We illustrate this phenomenon by training explicit classifiers constructed using the same feature encoding as those generating the data labels, but different variational unitaries $V(\boldsymbol{\theta})$ (see Eq. (E3)). We take these variational unitaries to resemble a Trotter evolution of a 1D-Heisenberg model with circular boundary conditions:

$$V(\boldsymbol{\theta}) = \prod_{i=1}^{L} \left(\prod_{j=1}^{n} e^{\mathrm{i}\theta_{i,j,0} Z_j Z_{j+1}} e^{\mathrm{i}\theta_{i,j,1} Y_j Y_{j+1}} e^{\mathrm{i}\theta_{i,j,2} X_j X_{j+1}} \right)$$

for the same number of layers L and the same number of parameters $\boldsymbol{\theta} \in [0, 2\pi]^{3nL}$ as the generating classifier. These are followed by a (weighted) Z_1 observable. Their learning performance is presented in Fig. 12.