npj | quantum information

Article

Published in partnership with The University of New South Wales

https://doi.org/10.1038/s41534-026-01202-5

QKAN: quantum Kolmogorov-Arnold networks with applications in machine learning and multivariate state preparation

Petr Ivashkov[1,2] ✉, Po-Wei Huang[1,3], Kelvin Koor[1], Lirandë Pira[1] ✉ & Patrick Rebentrost[1,4] ✉

We introduce quantum Kolmogorov-Arnold networks (QKAN), a quantum algorithmic framework inspired by the recently proposed Kolmogorov-Arnold Networks (KAN). QKAN inherits the compositional structure of KAN and is based on block-encodings, constructed recursively from a single layer using quantum singular value transformation. We demonstrate the algorithmic utility of QKAN in two applications. First, we introduce and analyze QKAN as a quantum learning model, treating the eigenvalues of block-encoded matrices as neurons and applying parametrized activation functions on the edges of the network. We show that QKAN is a wide-and-shallow neural architecture, where shallow depth is compensated by exponentially wide layers whenever efficient block-encodings of inputs are available. We further discuss how to parametrize and train QKAN using parametrized quantum circuits and quantum linear algebra subroutines. Second, we demonstrate that QKAN can serve as a multivariate quantum state-preparation protocol for functions with shallow compositional structure. We demonstrate this by efficiently preparing a multivariate Gaussian quantum state using a two-layer QKAN. Looking forward, we anticipate that QKAN's compositional and modular design will enable new applications in quantum machine learning and quantum state preparation.

Kolmogorov-Arnold representation theorem (KART) states that any continuous function of multiple variables can be decomposed using two layers of composition and summation of univariate functions[1–4]. Recently, Liu et al.[5] extended this compositional structure beyond two layers, providing an alternative neural network design aimed at offering advantages over traditional feedforward multilayer perceptrons (MLPs)[6–9]. Although KANs do not inherit the universal representation property of KART, their structure, based on compositions of parametrized univariate activation functions, can yield better interpretability and improved accuracy on small-scale tasks[5]. In scientific applications, where many target functions admit symbolic formulas, KANs can reveal modular structure and potentially aid in the discovery of new physical laws, making them a promising tool for scientific discovery[10]. The KAN architecture has inspired multiple extensions and applications, including Convolutional KANs[11], Graph KANs[12,13], Chebyshev KANs[14], KANs for quantum circuits[15] and others[16–22].

In this work, motivated by the potential of KANs in the classical setting, we introduce a quantum version, QKAN, a structured quantum architecture that leverages the quantum singular value transformation (QSVT) to apply nonlinear transformations. QSVT applies polynomial transformations to the singular values of a matrix encoded as a block of a unitary (a *block-encoding*), utilizing the power of quantum computers to manipulate exponentially large unitary operators efficiently[23]. QSVT has seen widespread adoption as a quantum meta-algorithm, both rederiving previous algorithms[24,25] and designing new efficient quantum algorithms[26–30]. QKAN uses block-encodings as its input and output model, representing both the input and output vectors as block-encoded diagonal operators, which can be manipulated using quantum linear algebra subroutines, as illustrated in Fig. 1. We demonstrate the algorithmic utility of QKAN in two applications.

First, we introduce and analyze QKAN as a quantum learning model. In quantum machine learning, models are developed using quantum

[1]Centre for Quantum Technologies, National University of Singapore, Singapore, Singapore. [2]Department of Information Technology and Electrical Engineering, ETH Zürich, Zürich, Switzerland. [3]Mathematical Institute, University of Oxford, Oxford, UK. [4]Department of Computer Science, National University of Singapore, Singapore, Singapore. ✉e-mail: pivashkov@ethz.ch; lpira@nus.edu.sg; cqtfpr@nus.edu.sg

mechanical principles[31–33]. Existing approaches include variational quantum algorithms (VQAs) employing parametrized quantum circuits whose parameters are optimized to minimize a cost function[34–39], similar to MLPs. Their generalization, expressibility, and interpretability have been extensively studied in refs. 40–46. In the fault-tolerant regime, various quantum implementations of classical machine learning algorithms have been proposed, including support vector machines[47], deep convolutional neural networks[48], transformers[49], and various others[50–53]. Contrary to previous architectures, QKAN treats the eigenvalues of block-encoded matrices as neurons and applies parametrized activation functions on network edges via linear combinations of Chebyshev polynomials, or other basis functions that can be realized efficiently using QSVT. The gate complexity of QKAN scales linearly with the cost of constructing the block-encoding of an N-dimensional input vector, which in certain cases, such as for inherently quantum inputs, can be $\mathcal{O}(\text{polylog}(N))$. At the same time, composing layers incurs an exponential overhead in depth due to the recursive QSVT-based construction, so QKAN is naturally constrained to be shallow. This makes QKAN a wide-and-shallow architecture: when efficient block-encodings are available, a shallow QKAN can realize exponentially wide layers at a polylogarithmic cost, a regime that is inaccessible to classical neural networks. For example, given access to a quantum unitary that prepares a N-dimensional quantum state of interest efficiently, we can process that state by computing a multivariate function of its amplitudes in $\mathcal{O}(\text{polylog}(N))$ time, assuming that the target function admits an efficient polynomial approximation. Such an operation generally requires $\mathcal{O}(N)$ classical runtime. We note that although we implement QKAN with Chebyshev polynomials to facilitate training and interpretability, QKAN is not restricted to the Chebyshev basis and can employ any bounded-degree, bounded-range polynomials realizable via QSVT.

Second, we demonstrate that QKAN can serve as a multivariate quantum state-preparation protocol. The goal of quantum state preparation is to prepare a quantum state, for example, for use in other quantum algorithms. The problem of loading univariate functions has been extensively investigated in the prior literature[54–60]. However, extensions to multivariate state preparation remain scarce despite their importance[61–63]. We illustrate how QKAN's compositional circuitry can efficiently prepare families of multivariate high-dimensional distributions by exploiting their compositional structure. As a concrete example, we show that a two-layer QKAN can efficiently load a D-dimensional Gaussian distribution into a quantum state.

Results

Notation and preliminaries

Throughout this manuscript, N denotes the dimension of input, assumed to be a power of two, and $n = \log_2(N)$ represents the number of qubits. K denotes the dimension of the output and, similarly, $k = \log_2(K)$. The subscript n in $|\psi\rangle_n$ and U_n indicates the size of the system in terms of qubits. For a vector v, $\|v\|_p$ is the ℓ_p-norm of v. For a matrix A, $\|A\|$ is the spectral norm of A. Further, $\text{diag}(x_1, x_2, \ldots, x_N)$ represents a diagonal matrix whose diagonal entries are $x_1, x_2, \ldots, x_N$. Here, $T_r(x) \in \mathbb{R}[x]$ is the r-th Chebyshev polynomial of the first kind, defined as $T_r(x) := \cos(r \arccos(x))$. We denote $\mathbb{R}[x]$ as the set of all polynomials with real coefficients in the variable x. We adopt the convention that indices in summations run from 1 to the upper limit of the sum and use $[N]$ to denote the set $\{1, \ldots, N\}$.

Kolmogorov-Arnold Networks (KAN)

Kolmogorov-Arnold representation theorem states that any continuous multivariate function can be represented as a composition of univariate functions with the summation[1,3,64]. Formally, for any continuous function $f : [0,1]^N \to \mathbb{R}$, there exist continuous inner functions $\phi_{pq} : [0,1] \to \mathbb{R}$ (independent of f) and outer functions $g_q : \mathbb{R} \to \mathbb{R}$ (dependent of f) such that

$$f(x_1, \cdots, x_N) = \sum_{q=1}^{2N+1} g_q \left(\sum_{p=1}^{N} \phi_{pq}\left(x_p\right) \right). \tag{1}$$

In the context of neural networks, KART has been studied to explain how deep learning can overcome the curse of dimensionality, with one approach involving the approximation of the inner and outer functions of the KART representation using neural networks. However, the practicality of this approach is limited by the high nonsmoothness of the inner and outer functions, even when the original function is smooth, posing significant challenges in accurate approximation and robustness to noise.

Liu et al.[5] proposed generalizing the compositional structure of KART to include more layers. This architecture, called Kolmogorov-Arnold Networks, can contain an arbitrary number of layers, as opposed to the two layers guaranteed in KART. Here, we define KAN as outlined in the original study and only slightly adapt the notation of ref. 5.

Definition 1. (KAN layer,[5]). Define a KAN layer as a transformation $\Phi : \mathbb{R}^N \to \mathbb{R}^K$ that takes a real vector $\vec{x}$ as input and outputs a real vector $\Phi(\vec{x})$ such that

$$\Phi(\vec{x}) = \left(\sum_{p=1}^{N} \phi_{p1}(x_p), \ldots, \sum_{p=1}^{N} \phi_{pK}(x_p) \right), \tag{2}$$

where $\phi_{pq} : \mathbb{R} \to \mathbb{R}$ are univariate functions.

This transformation can be interpreted as placing activation functions ϕ_{pq} on the edges connecting the input nodes $p \in [N]$ to the output nodes $q \in [K]$ of a single layer neural network and applying summation on the output nodes.

Definition 2. (KAN,[5]). Define KAN as a neural network architecture consisting of concatenated KAN layers, where the output of the previous layer serves as the input to the next one. Let L be the number of KAN layers and let an integer array $[N^{(0)}, N^{(1)}, \ldots, N^{(L)}]$ be given, where $N^{(l)}$ denotes the number of nodes in the l-th KAN layer. The KAN output, denoted by $\text{KAN}(\vec{x})$, is the composition of the individual layers:

$$\text{KAN}(\vec{x}) = \Phi^{(L)} \circ \cdots \circ \Phi^{(1)}(\vec{x}), \tag{3}$$

where each $\Phi^{(l)} : \mathbb{R}^{N^{(l-1)}} \to \mathbb{R}^{N^{(l)}}$ is specified by an array of univariate functions $\{\phi_{pq}^{(l)}\}$.

As per the definition above, in KAN, the univariate functions ϕ_{pq} are parametrized as linear combinations of basis functions. The choice of a basis can be tailored to the specific application. For example, the original KAN implementation used B-splines defined on a grid. Subsequent works considered wavelets, Chebyshev polynomials, and Fourier expansion.

It is crucial to emphasize that KAN does not inherit the universal representation power of KART because the inner and outer functions of KART may not be learnable in practice[65–67]. Therefore, there is no guarantee that a deep KAN can represent any given multivariate function. Nevertheless, KAN appears to be successful in certain applications, particularly in science. For example, KAN has outperformed MLPs in learning symbolic functions commonly found in physics[5,10]. Additionally, KAN offers a significant interpretability advantage: individual activation functions can be inspected, and the network can be pruned by removing functions that closely resemble zero functions, potentially discovering sparse compositional structures. Compared to MLPs, KANs have displayed the property of efficiency in certain cases and exhibit a lower spectral bias toward lower frequencies[68].

Quantum Kolmogorov-Arnold Network (QKAN)

In this section, we establish the main contribution of our work. Namely, we develop a quantum implementation of the Kolmogorov-Arnold Network (QKAN), illustrated in Fig. 1. QKAN is designed to realize the classical KAN on a quantum computer and, as such, consists solely of unitary transformations. This implementation leads to several technical differences between KAN and QKAN. Firstly, QKAN operates on block-encodings of vectors rather than directly on vectors themselves. In this representation, the vector

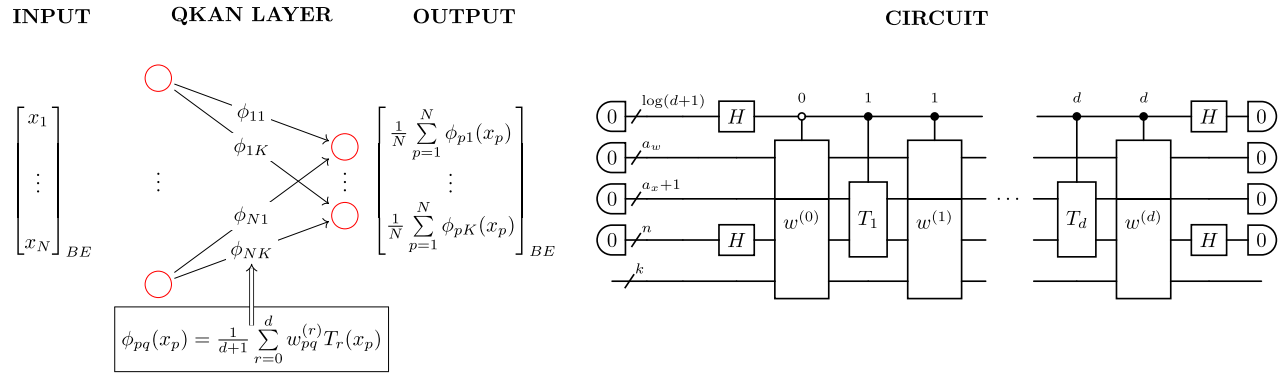

Fig. 1 | Construction of a CHEB-QKAN layer with the corresponding quantum circuit. The input to the QKAN model is a diagonal block-encoding of an N-dimensional real vector $\vec{x}$. The CHEB-QKAN layer applies univariate activation functions ϕ_{pq} to each input component x_p, where $p \in [N]$ indexes input nodes and $q \in [K]$ indexes output nodes. The output vector is computed as a sum over activated input nodes. This operation yields a block-encoded real K-dimensional output vector. The quantum circuit implementation requires $1 + \log_2(d+1)$ qubits for the construction and linear combination of weighted Chebyshev polynomials, $a_w + a_x$ qubits for the block-encodings of input and weights, $n = \log_2 N$ qubits for input vector encoding, and $k = \log_2 K$ qubits for output. The circuit consists of a series of multi-controlled block-encodings of Chebyshev polynomials, interspersed with diagonal block-encodings of the corresponding real weights. The entire circuit represents a block-encoding of the K-dimensional vector corresponding to the CHEB-QKAN layer, with auxiliary qubits initialized and measured in the $|0\rangle$ state.

$$U_x = \left[\begin{array}{cccc:c} x_1 & 0 & \cdots & 0 & \\ 0 & x_2 & \cdots & 0 & \\ \vdots & \vdots & \ddots & \vdots & * \\ 0 & 0 & \cdots & x_N & \\ \hdashline & & * & & * \end{array}\right] =: \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix}_{\mathrm{BE}}$$

Fig. 2 | Diagonal block-encoding. The top left block is a diagonal matrix whose entries are the components of an N-dimensional vector $\vec{x}$, while the remaining matrix blocks, denoted by asterisks ($*$), are unspecified.

is encoded in the diagonal of the top-left block of a unitary matrix. Secondly, due to the constraints of unitarity, the elements of the vectors are bounded in magnitude by one. Throughout this manuscript, we frequently utilize block-encodings of diagonal matrices. For clarity, we adopt the term "diagonal block-encoding of a vector" to refer to a block-encoding of a diagonal matrix where the diagonal elements correspond to the elements of the vector, as shown in Fig. 2.

In this work, we limit the discussion to real values; in particular, the input, output, and weights of the QKAN model are assumed to be real. A generalization to complex numbers is possible without a significant increase in complexity by treating real and imaginary parts separately, and is a direction for future work.

We define the QKAN layer and the full QKAN model in analogy to their classical counterparts.

Definition 3. (QKAN Layer). Define a QKAN layer as a transformation that, given query access to a diagonal $(1, a_x, \varepsilon_x)$-block-encoding of $\vec{x} \in [-1,1]^N$, constructs a diagonal $(1, a_x', \varepsilon_x')$-block-encoding of $\Phi(\vec{x}) \in [-1,1]^K$ such that

$$\Phi(\vec{x}) = \left(\frac{1}{N}\sum_{p=1}^{N}\phi_{p1}(x_p), \ldots, \frac{1}{N}\sum_{p=1}^{N}\phi_{pK}(x_p)\right), \tag{4}$$

where ϕ_{pq}: $[-1,1] \to [-1,1]$ are univariate functions.

Definition 4. (QKAN). Define QKAN as a composition of QKAN layers, where the block-encoding produced by one layer serves as the input to the next one. Let L be the number of layers in the QKAN architecture, and let an integer array $[N^{(0)}, N^{(1)}, \ldots, N^{(L)}]$ be given, where $N^{(l)}$ represents the number of nodes in the l-th layer. The QKAN output, denoted by QKAN($\vec{x}$), is a diagonal block-encoding of an $N^{(L)}$-dimensional vector, constructed recursively from the composition of layers:

$$\mathrm{QKAN}(\vec{x}) = \Phi^{(L)} \circ \cdots \circ \Phi^{(1)}(\vec{x}), \tag{5}$$

where each $\Phi^{(l)} : [-1,1]^{N^{(l-1)}} \to [-1,1]^{N^{(l)}}$ is specified by an array of univariate functions $\{\phi_{pq}^{(l)}\}$.

Remark. It is important to note that the term "layer" in QKAN may be slightly misleading. Unlike in classical KAN, where layers are concatenated, a QKAN layer serves as a primitive building block for the subsequent layer, leading to a recursive construction.

Similarly to KAN, in QKAN, the univariate functions ϕ_{pq} are parametrized as linear combinations of basis functions. For QKAN, Chebyshev polynomials are a natural choice of basis because they can be efficiently implemented within the qubitization framework. We define CHEB-QKAN as a QKAN where the activation functions ϕ_{pq} are expressed as linear combinations of Chebyshev polynomials, an idea previously explored in ref. 14:

$$\phi_{pq}(x) = \frac{1}{d+1}\sum_{r=0}^{d} w_{pq}^{(r)} T_r(x), \tag{6}$$

where $w_{pq}^{(r)} \in [-1,1]$ are linear coefficients. With these definitions, we can now state the main result of this work.

Theorem 1. (CHEB-QKAN). Given access to a controlled diagonal $(1, a_x, \varepsilon_x)$-block-encoding U_x of an input vector $\vec{x} \in [-1,1]^N$, and access to $d+1$ controlled diagonal $(1, a_w, \varepsilon_w)$-block-encodings $U_{w^{(r)}}$ of weight vectors $\vec{w}^{(r)} \in [-1,1]^{NK}$, we can construct a diagonal $(1, a_x + 1 + a_w + \log_2(d+1) + n, 4d\sqrt{\varepsilon_x} + \varepsilon_w)$-block-encoding of a vector $\Phi(\vec{x}) \in [-1,1]^K$ corresponding to the *CHEB-QKAN* layer

$$\Phi(\vec{x}) = \left(\frac{1}{N}\sum_{p=1}^{N}\phi_{p1}(x_p), \ldots, \frac{1}{N}\sum_{p=1}^{N}\phi_{pK}(x_p)\right), \tag{7}$$

where d is the maximal degree of Chebyshev polynomials used in the parameterization of activation functions ϕ_{pq}, using $\mathcal{O}(d^2)$ applications of controlled-U_x and controlled-$U_{w^{(r)}}$ and their adjoint versions.

In the above theorem, we construct QKAN using Chebyshev polynomials. However, we emphasize that QKAN is not limited to this particular basis set. In fact, due to the versatility of the QSVT framework, quantum implementations of other versions of KANs can be realized using efficient polynomial approximation of a wide range of basis functions[23,25,69,70]. For example, to implement the B-spline construction for KANs in Liu et al.[5]'s original paper, each individual spline can be implemented by first separating each piecewise section and then taking its sum by the linear combinations of unitaries method (LCU)[71]. Each individual piecewise polynomial can then be implemented by multiplying a polynomial function constructed via QSVT with a threshold function formed by the sum of two Heaviside functions, which can in turn be approximated using polynomial approximations to the erf function via Lemma 10 and Corollary 5 of ref. 72.

Implementing CHEB-QKAN

We prove the main Theorem 1 by presenting a detailed construction of CHEB-QKAN in "CHEB-QKAN construction". Our construction of CHEB-QKAN relies on three basic operations: addition, multiplication, and QSVT. In a nutshell, we implement parametrized activation functions between nodes of two layers by taking linear combinations of a fixed set of basis functions, where each basis function is realized through QSVT. After having constructed a single CHEB-QKAN layer that transforms a diagonal block-encoding of an N-dimensional input vector into a diagonal block-encoding of a K-dimensional output vector, the obtained block-encoding can be used as the input to the next layer by serving as the starting point for the next layer's construction. One can immediately see that recursively transforming block-encodings in this manner results in a gate complexity that grows exponentially with the number of layers. This is because every output block-encoding is used as the elementary building block in the subsequent layer. Additionally, the total number of auxiliary qubits required increases linearly with the number of layers L. Finally, if the block-encodings of the input and weights are non-perfect, the error propagates recursively with every new layer, resulting in an amplified error in the output block-encoding. In summary, the recursive error propagation and the exponential dependency of circuit depth on the number of layers limit QKAN to a shallow, i.e., $L = \mathcal{O}(1)$, albeit wide, architecture. These considerations are made precise in Supplementary Note 2.

Application I: Quantum learning model

In this section, we introduce QKAN as an end-to-end quantum learning framework and outline its input and output models. In "Parametrization of the QKAN learning model", Section IV E, and Section IV F, we further detail the parameterization, training, and interpretability of QKAN.

By Theorem 1, QKAN implements a unitary whose diagonal entries block-encode a K-dimensional output vector. To recover these outputs classically, we apply the unitary to a quantum computational basis state and estimate a designated amplitude that encodes a multivariate function of the input that can serve for regression or classification. An end-to-end quantum speedup arises when the quantum implementation of this multivariate function requires exponentially fewer resources than a classical algorithm. The four core components enabling this speedup in QKAN are the quantum input encoding, the parametrization via Chebyshev expansions, the training algorithm, and the output extraction. Here we focus on the single-layer CHEB-QKAN case; extending to an L-layer CHEB-QKAN – and to general QKAN architectures – proceeds similarly, with an additional exponential dependence on L as discussed in Supplementary Note 2.

Constructing a diagonal block-encoding of a generic N-dimensional classical vector requires at least $\mathcal{O}(N)$ gates[55]. Because QKAN's complexity is measured in queries to the input block-encoding, we must therefore restrict to inputs that admit efficient block-encodings that can be prepared in $\mathcal{O}(\text{polylog}(N))$ time. A natural setting is when the input is inherently quantum: for example, a unitary produced by a variational quantum algorithm that prepares an approximate ground state; the time-evolution unitary e^{-iHt} encoding dynamical information; a block-encoding of a quantum Gibbs state $e^{-\beta H}$; or a block-encoding of a Hamiltonian H via the LCU method. Given such a unitary, we propose two methods for efficient diagonal block-encoding of the input vector. The first method treats U as an efficiently implementable state-preparation unitary and constructs a diagonal block-encoding of the amplitude vector $|\psi\rangle_n = U|0\rangle_n$ ("Input block-encoding for the learning model" Lemma 10). As a result, QKAN computes a multivariate function of the quantum amplitudes. The second method forms the diagonal block-encoding by taking the Hadamard (entry-wise) product with the identity $U \circ I$, retaining only the diagonal entries of the unitary ("Input block-encoding for the learning model" Lemma 11).

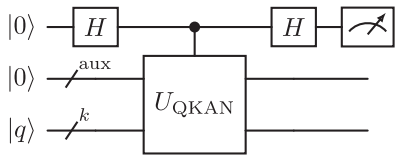

Fig. 3 | Circuit for solution extraction via Hadamard test. By estimating the expectation value of Pauli-Z on the top qubit, the circuit retrieves the value $U_{\text{QKAN},qq}$ to additive δ-precision.

To estimate the values within the diagonally block-encoded output vector $\text{QKAN}(\vec{x})$ to additive error δ, one can leverage the Hadamard test[73], using $\mathcal{O}(1/\delta^2)$ queries to the controlled diagonal block-encoding U_{QKAN}. Specifically, to obtain the value of the q-th entry of the output vector, we prepare the state $|\psi\rangle = (H \otimes I_{\text{aux}+k}) C U_{\text{QKAN}} (H \otimes I_{\text{aux}+k}) |0\rangle|0\rangle_{\text{aux}}|q\rangle_k$ and estimate the expectation value $\langle Z\rangle$ of the top qubit, as depicted in Fig. 3. As a result, we obtain $\text{QKAN}(\vec{x})_q = \text{Re}(\langle 0|_{\text{aux}}\langle q|_k U_{\text{QKAN}}|0\rangle_{\text{aux}}|q\rangle_k)$ to δ-precision. In addition, the Hadamard test can be combined with amplitude estimation[74] to reduce the number of queries to CU_{QKAN} to $\mathcal{O}(1/\delta)$. Theorem 2 makes this statement precise, with the proof deferred to Supplementary Note 3.

Theorem 2. (Output estimation of CHEB-QKAN). Given access to a controlled diagonal $(1, a_x, \varepsilon_x)$-block-encoding U_x of an input vector $\vec{x} \in [-1,1]^N$, and access to $d+1$ controlled diagonal $(1, a_w, \varepsilon_w)$-block-encodings $U_{w^{(r)}}$ of weight vectors $\vec{w}^{(r)} \in [-1,1]^{NK}$, we can estimate the value $\Phi(\vec{x})_q = \frac{1}{N}\sum_{p=1}^{N} \phi_{pq}(x_p)$ of the q-th component of the $CHEB-QKAN$ layer to $\left(4d\sqrt{\varepsilon_x} + \varepsilon_w + \delta\right)$-precision using $\mathcal{O}(d^2/\delta)$ applications of controlled-U_x and controlled-$U_{w^{(r)}}$ and their adjoint versions.

Any potential quantum speed-up is contingent on the cost of reading out the K entries produced by the final layer. First, the classical post-processing cost must remain sub-exponential. We therefore restrict the output dimension to $K = \mathcal{O}(\text{polylog}(N))$, since estimating an exponential number of values would itself take exponential time. Fortunately, setting $K = \mathcal{O}(1)$ is already sufficient for most regression and classification tasks. Second, consider the precision with which each amplitude $\alpha_q = \frac{1}{N}\sum_{p=1}^{N}\phi_{pq}(x_p)$ is estimated. Using the estimation procedure described in Theorem 2, an additive δ approximation requires $\mathcal{O}(d^2/\delta)$ queries, independent of $|\alpha_q|$. If a multiplicative (relative) error is required, i.e., $|\widehat{\alpha}_q - \alpha_q| < \delta\,|\alpha_q|$, the query count increases to $\mathcal{O}\left(d^2/(\delta\,|\alpha_q|)\right)$, because the amplitude must now be resolved to a fixed fraction of its value. Consequently, the potential quantum speed-up is preserved as long as α_q does not decay exponentially, that is, provided $|\alpha_q|^{-1} = \mathcal{O}(\text{polylog}(N))$. The last requirement is not an artifact of QKAN; it is analogous to the inverse-amplitude overhead in amplitude-amplification/estimation, which scales as $1/\sqrt{a}$ with the marked-state probability a[74], and black-box state-preparation, that scales as $1/\mathcal{F}$ with the ℓ_2 filling fraction $\mathcal{F}$ of the target function[58].

Application II: Multivariate state preparation

On the other hand, given the quantum nature of our algorithm, QKAN can also output a quantum state as the solution. While QKAN can be seen as a machine learning model, the algorithm itself leads to a form of multivariate

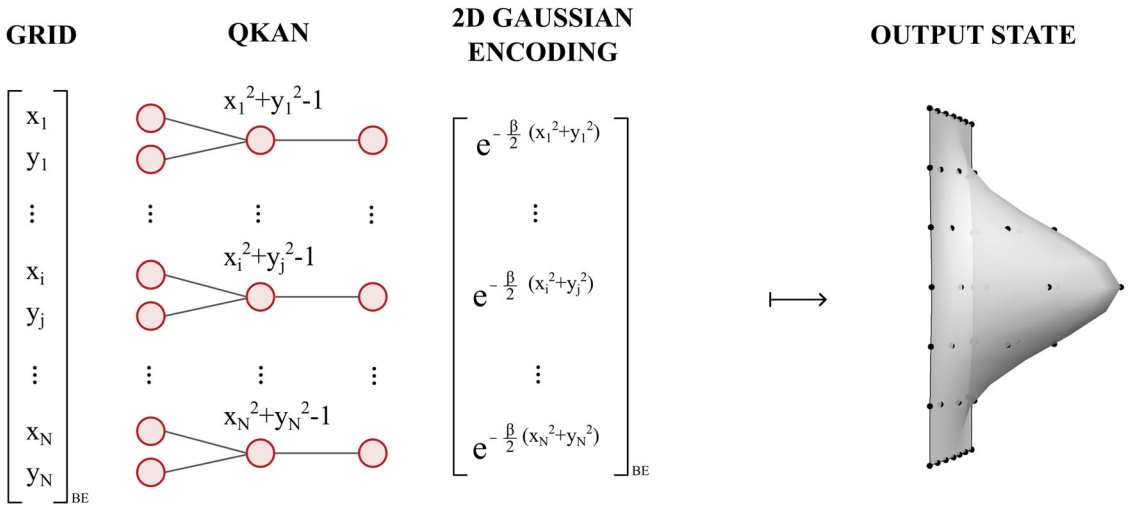

Fig. 4 | Example: 2D Gaussian state preparation via QKAN. Starting from a vectorized 2D grid of points $\{(x_i, y_j)\}$ encoded as a diagonal block-encoding (left), the first QKAN layer applies Chebyshev polynomial T_2 and sums over the two dimensions, computing $\frac{1}{2}(T_2(x_i) + T_2(y_j)) = x_i^2 + y_j^2 - 1$. A second layer uses a polynomial approximation of the exponential $e^{-\frac{\beta}{2}(x+1)}$ to block-encode the values $e^{-\frac{\beta}{2}(x_i^2+y_j^2)}$. Finally, applying this block-encoding to the uniform superposition and amplitude amplifying yields the desired 2D Gaussian distribution (right).

state preparation[61]. The resulting block-encoding can be applied to the uniform superposition $|+\rangle_k := H_k|0\rangle_k$, in combination with amplitude amplification[74,75], to produce a quantum state with amplitudes encoding multivariate functions of the input. In the following, we show how the compositional framework of QKAN introduced in "Implementing CHEB-QKAN" can be used to prepare quantum states encoding multivariate functions on a D-dimensional regular grid. Specifically, we work through the special case of a D-dimensional Gaussian. Efficient Gaussian state preparation has been studied extensively, ranging from Grover-Rudolph[76] and Kitaev-Webb[77] to more recent approaches improving depth and ancilla costs[78–80]. Our use of the Gaussian example is not meant to compete with these tailored methods but to illustrate how QKAN's modular, compositional structure enables the assembling of multivariate amplitudes from elementary components. Finally, in "Generalized state preparation via CHEB-QKAN", we remove the grid restriction and show that any CHEB-QKAN layer – acting on an arbitrary input register – yields a valid multivariate state-preparation routine.

Our aim is to prepare a quantum state of the form $\sum_{i_1,\ldots,i_D} f(x_{(i_1,\ldots,i_D)})\,|i_1,\ldots,i_D\rangle$ with $x_{(i_1,\ldots,i_D)} = (-1 + i_j s)_{j\in[D]} \in [-1,1]^D$ and $(i_1,\ldots,i_D) \in \{0,\ldots,2^n-1\}^D$. To do this, we first encode the vectorized D-dimensional grid points $x_{(i_1,\ldots,i_D)}$ as a diagonal operator G_D, treating the vectorized grid as the classical input $\vec{x}$ to QKAN. In Supplementary Note 4.A, we provide the proof of Lemma 3 by extending the one-dimensional construction of Rosenkranz et al.[61] to D dimensions.

Lemma 3. (Multivariate grid encoding). Let $G_D = \mathrm{diag}\left(x_{(i_1,\ldots,i_D)}\right)_{(i_1,\ldots,i_D)\in\{0,\ldots,2^n-1\}^D}$, where

$$x_{(i_1,\ldots,i_D)} = \left(-1 + i_1 s,\ -1 + i_2 s,\ \ldots,\ -1 + i_D s\right) \in [-1,1]^D, \tag{8}$$

be a uniform (vectorized) D-dimensional grid on $[-1,1]^D$ with step size $s = \frac{2}{2^n-1}$ in every direction. The dimension of G_D is $D\,2^{nD}$. Then,

$$\begin{aligned} G_1 &= \sum_{i=1}^{n}\left(\tfrac{2^{i-1}}{2^n-1}\right) I_{i-1}\otimes XZX\otimes I_{n-i} \text{ and} \\ G_D &= \sum_{j=1}^{D} I_{n(j-1)}\otimes G_1\otimes I_{n(D-j)}\otimes |j\rangle\langle j|, \end{aligned} \tag{9}$$

and we can create a $(1, D\lceil\log n\rceil, 0)$-block-encoding of G_D using $\mathcal{O}(Dn(\log n + \log D))$ two-qubit gates.

In the following, we illustrate how the QKAN architecture can be used to provide a multivariate Gaussian quantum state. The main result is summarized in Theorem 4:

Theorem 4. We can prepare a Dn-qubit quantum state $|\psi\rangle$ with amplitudes corresponding to a D-dimensional Gaussian distribution on a regular square grid of size $(2^n)^D$ such that

$$\left\| |\psi\rangle - \frac{1}{\widetilde{F}_{\exp}} \sum_{i_1,\ldots,i_D}^{2^n} \exp\left(-\frac{\beta}{2}\sum_{j=1}^{D} x_{i_j}^2\right) |i_1,\ldots,i_D\rangle \right\|_2 \le \delta, \tag{10}$$

where $\widetilde{F}_{\exp}$ normalizes the target state. The procedure succeeds with arbitrarily high probability by using $\widetilde{\mathcal{O}}\left(\beta^{\frac{D+1}{4}+\frac{1}{2}}\, n\, \log\frac{1}{\delta}\right)$ two-qubit gates and $D\lceil\log n\rceil + \lceil\log D\rceil + 4$ ancilla qubits.

Here, the $\widetilde{\mathcal{O}}$ notation suppresses the logarithmic factors $\log n$ and $\log\beta$, and treats D as a constant. The details of gate and qubit complexity can be found in Supplementary Note 4.D. The proof of Theorem 4 is an explicit four-step construction, and the entire procedure can be viewed as an instance of a two-layer QKAN architecture. For example, Fig. 4 gives an illustration for the $D = 2$ case: starting from a vectorized 2D grid of points, the first layer computes $x_i^2 + y_j^2 - 1$, and the second layer applies a polynomial approximation of the exponential to produce the Gaussian output state.

The vectorized D-dimensional grid serves as the input to the first layer. By Lemma 3, we start by creating a $(1, D\lceil\log n\rceil, 0)$-block-encoding U_{G_D} of G_D using $\mathcal{O}(Dn(\log n + \log D))$ two-qubit gates. The first layer applies the activation function $T_2(x) = 2x^2 - 1$ to all entries of the multivariate grid, followed by the summation over the D dimensions, realizing the transformation $G_D \mapsto \Phi_1(G_D)$. Specifically, by Theorem 9, we first construct a $(1, D\lceil\log n\rceil + 1, 0)$-block-encoding of $T_2(G_D)$ according to the *CHEB* step in the QKAN construction:

$$G_D \mapsto T_2(G_D) = \sum_{j=1}^{D} I_{n(j-1)}\otimes T_2(G_1)\otimes I_{n(D-j)}\otimes |j\rangle\langle j|. \tag{11}$$

This step uses $\mathcal{O}(1)$ queries to the controlled version of U_{G_D} and $\mathcal{O}(D\log n)$ other two-qubit gates. We then sum over the D dimensions by applying Hadamards on the last k qubits of $T_2(G_D)$ and absorbing them into the

auxiliary register, according to the *SUM* step:

$$T_2(G_D) \mapsto \Phi_1(G_D) = \left(I_{Dn} \otimes \langle 0|_k H_k\right) T_2(G_D) \left(I_{Dn} \otimes H_k |0\rangle_k\right) = \frac{1}{D} \sum_{j=1}^{D} I_{n(j-1)} \otimes T_2(G_1) \otimes I_{n(D-j)}. \tag{12}$$

The above transformation yields a $(1, D\lceil \log n \rceil + \lceil \log D \rceil + 1, 0)$-block-encoding of $\Phi_1(G_D)$. In component-wise form, the transformation is $x_{(i_1,\ldots,i_D)} \mapsto \frac{2}{D}\sum_{j=1}^{D}(-1+i_j s)^2 - 1$. The layer dimension is reduced from $\dim G_D = D2^{nD}$ to $\dim g\Phi_1(G_D) = 2^{nD}$. The second layer implements the exponential decay:

$$\Phi_1(G_D) \mapsto \Phi_2(G_D) \approx \exp\left(-\widetilde{\beta}\left[\Phi_1(G_D) + 1\right]\right) \text{ with } \widetilde{\beta} = \frac{D}{4}\beta. \tag{13}$$

In Supplementary Note 4.B we show that one can find an approximating polynomial $P_d(x)$ with degree $d = \mathcal{O}\left(\sqrt{D\beta}\log\frac{1}{\varepsilon}\right)$ such that $|P_d(x) - e^{-\beta(x+1)}| \le \varepsilon$ on $[-1, 1]$. Such a polynomial can be realized by QSVT invoking Theorem 9 using $\mathcal{O}(d)$ queries to the block-encoding of $\Phi_1(G_D)$, constructed in the previous step, and $\mathcal{O}\left(d \times (D\log n + \log D)\right)$ other two-qubit gates. By Theorem 9, we obtain a $(1, D\lceil \log n \rceil + \lceil \log D \rceil + 3, 0)$-block-encoding of $\Phi_2(G_D)$. In component-wise form, the transformation is

$$\frac{2}{D}\sum_{j=1}^{D}(-1+i_j s)^2 - 1 \mapsto P_d\left(\frac{2}{D}\sum_{j=1}^{D}(-1+i_j s)^2 - 1\right) \approx \exp\left(-\frac{\beta}{2}\sum_{j=1}^{D}(-1+i_j s)^2\right). \tag{14}$$

Therefore, the diagonal entries of $\Phi_2(G_D)$ correspond to the amplitudes of a D-dimensional Gaussian up to a maximal error ε. In Supplementary Note 4.D, we show that d must be chosen as a function of D, β, and δ to obtain the target state preparation accuracy δ: $d = \mathcal{O}\left(\sqrt{D\beta}\log\beta^{\frac{D}{4}}\log\frac{\infty}{\delta}\right)$. By applying the block-encoding to the uniform superposition $|+\rangle_{Dn}$ and post-selecting on the auxiliary register being in the $|0\rangle_a$ state, we prepare the desired state

$$|\psi\rangle = \frac{\Phi_2(G_D)|+\rangle_{Dn}}{\left\|\Phi_2(G_D)|+\rangle_{Dn}\right\|_2} \tag{15}$$

probabilistically, with success probability $p = \left\|\Phi_2(G_D)|+\rangle_{Dn}\right\|_2^2$. We can boost p to an arbitrarily high success probability by using fixed-point amplitude amplification with $\mathcal{O}(1/\sqrt{p})$ queries to the controlled block-encodings of $\Phi_2(G_D)$ (and their adjoint versions)[75]. In Supplementary Note 4.C we show that $1/\sqrt{p}$ can be upper-bounded by $\widetilde{\mathcal{O}}(\beta^{\frac{D}{4}})$ using a lower bound on the continuous version of the ℓ_2-filling fraction of the D-dimensional Gaussian. The total gate complexity arises from $\widetilde{\mathcal{O}}(\beta^{\frac{D}{4}})$ queries to the (controlled) block-encodings of $\Phi_2(G_D)$, as shown in Supplementary Note 4.D.

This compositional approach readily extends beyond Gaussian amplitudes to a broader class of multivariate functions. Any target map

$$f(x_{(i_1,\ldots,i_D)}) = g_L\left(g_{L-1}(\cdots g_1(x_{(i_1,\ldots,i_D)}))\right), \tag{16}$$

with each g_i admitting an efficient polynomial approximation of degree d_i, can be implemented by cascading L QKAN layers. This modularity allows one to leverage known polynomial expansions for elementary functions, such as $\sin(x)$, $\exp(x)$, or $\log(x)$, and assemble them into more complex amplitudes via successive QKAN layers. Crucially, because QKAN composes recursively by invoking the block-encoding constructed in the previous layer as the elementary building block for the next one, the overall two-qubit-gate cost scales multiplicatively as $\mathcal{O}(d_1\, d_2 \cdots d_L)$, making it essential to keep the compositional depth L shallow.

Finally, in addition to the fully explicit state-preparation constructions presented above, any intermediate QKAN layer—when viewed as a parametrized quantum learning model with trainable activation functions—can itself produce a parametrized quantum state when applied to a uniform superposition; this provides a variational multivariate state-preparation method for generic functions, as formalized in "Generalized state preparation via CHEB-QKAN".

Discussion

In this work, we have defined and implemented a quantum version of the recently proposed KAN architecture in ref. 5. Our proposed QKAN architecture is built on block-encodings, where both the input and the output are block-encodings. More specifically, it employs a recursive construction where block-encodings obtained in the previous layer serve as a primitive building block in the next layer. The potential applications of QKAN will be reliant on the availability of efficient block-encodings of the inputs. Specifically, we demonstrated that QKAN can serve as a quantum learning model by giving an explicit construction for encoding and training its parameters. Moreover, we demonstrated that QKAN has a broader algorithmic utility by serving as a multivariate state preparation protocol, exemplified by an explicit construction of a D-dimensional Gaussian distribution.

The QKAN architecture has several strengths. Firstly, it depends linearly on the cost of constructing block-encodings of input and weights. This dependency can lead to efficient implementation procedures assuming efficient block-encoding methods. Additionally, we propose that QKAN is potentially suitable for direct quantum input, for instance, quantum states whose analysis would be intractable classically. For example, in phase classification tasks[81], if a state corresponding to an unknown quantum phase of a physical system can be prepared efficiently, one may attempt to train QKAN in a supervised manner to classify the phase of that state. This classification would be achieved by computing a multivariate function of the state's amplitudes, potentially leading to the discovery of new order parameters. In addition, QKAN is a versatile architecture that admits different ways to encode data, parameterize weights, and perform training. Lastly, its underlying mechanism, the QSVT framework, allows the implementation of different sets of basis functions tailored to different applications.

On the other hand, QKAN exhibits several caveats. Firstly, it inherits limitations from the classical KANs, whose full potential remains to be established, even though symbolic regression tasks in science applications are brought forth. Secondly, the query complexity of a multilayer QKAN scales exponentially in the number of layers, limiting QKAN to a shallow, albeit wide architecture. Finally, some caveats arise due to the nature of the quantum subroutines involved. Quantum computers are good at representing polynomials, but not all functions can be efficiently approximated by polynomials. Therefore, the available basis functions for QKAN may generally be less powerful in approximating arbitrary functions directly compared to, for example, using splines as basis functions.

Within the broader quantum machine learning literature and even quantum algorithms, QKAN is a novel result in multiple directions. Firstly, it departs from variational architectures and steps into the more powerful quantum linear algebra toolset. From another perspective, it brings the aspect of parameterization into fault-tolerant subroutines for quantum machine learning. Secondly, it is a quantum learning model built on a new paradigm — that of having a decomposition of a function into single transformed features and summations. Finally, QKAN enables a version of multivariate state preparation, thereby serving as an algorithmic subroutine. We hope that this work will serve as a motivation to further investigate our QKAN architecture and other types of QKAN architecture, and build quantum models beyond near-term techniques.

Methods

Block-encoding and quantum subroutines

In the following, we outline some known results that are used in different parts of the construction and parameterization of QKAN. We begin by formally defining block-encoding:

Definition 5. (Block-encoding – Definition 24[23], see also refs. 82,83). Let A be an n-qubit matrix, $\alpha, \varepsilon \in \mathbb{R}_+$ and $a \in \mathbb{N}$. We say that the $(n+a)$-qubit unitary U is an (α, a, ε)-block-encoding of A if

$$\| A - \alpha(\langle 0|_a \otimes I_n) U (|0\rangle_a \otimes I_n) \| \leq \varepsilon. \tag{17}$$

Given block-encodings of operators A_i, we can construct a block-encoding of their linear combination using an auxiliary tool known as a "state-preparation pair". Recall that $\| \cdot \|_1$ is the ℓ_1/Manhattan norm.

Definition 6. (State preparation pair – Definition 28,[23]). Let $\vec{y} \in \mathbb{C}^m$ and $\| \vec{y} \|_1 \leq \beta$. The pair of unitaries (P_L, P_R) is called a $(\beta, b, \varepsilon_{\mathrm{SP}})$-state-preparation-pair for $\vec{y}$ if

$$P_L|0^b\rangle = \sum_{j=1}^{2^b} c_j |j\rangle,\ P_R|0^b\rangle = \sum_{j=1}^{2^b} d_j |j\rangle, \tag{18}$$

such that $\sum_{j=1}^m |y_j - \beta c_j^* d_j| \leq \varepsilon_{\mathrm{SP}}$ and $c_j^* d_j = 0$ for $j = m+1, \ldots, 2^b$.

One can think of a state preparation pair as encoding the desired state/vector $\vec{y}$ in the first m elements of a length-2^b column vector whose elements are $c_j^* d_j$, up to an error of $\varepsilon_{\mathrm{SP}}$. The role of β is to take care of normalization.

Lemma 5. (Linear combination of block-encodings – Lemma 29,[23]). Let $A = \sum_{j=1}^m y_j A_j$ be an n-qubit operator and $\varepsilon \in \mathbb{R}^+$. Suppose that (P_L, P_R) is a $(\beta, b, \varepsilon_1)$-state-preparation-pair for $\vec{y}$ and

$$W = \sum_{j=1}^m |j\rangle\langle j| \otimes U_j + ((I - \sum_{j=1}^m |j\rangle\langle j|) \otimes I_a \otimes I_n) \tag{19}$$

is an $n + a + b$ qubit unitary such that for all $j \in 1, \ldots, m$ we have that U_j is an $(\alpha, a, \varepsilon_2)$-block-encoding of A_j. Then we can implement a $(\alpha\beta, a+b, \alpha\varepsilon_1 + \beta\varepsilon_2)$-block-encoding of A, with a single use of W, P_R and $P_L^\dagger$.

We can also construct a block-encoding of a product of two block-encoded matrices.

Lemma 6. (Product of block-encodings – Lemma 30,[23]). Let U_A be a $(\alpha, a, \varepsilon_A)$-block-encoding of A and U_B be a $(\beta, b, \varepsilon_B)$-block-encoding of B, where A, B are n-qubit operators. Then, $(I_b \otimes U_A)(I_a \otimes U_B)$ is a $(\alpha\beta, a+b, \alpha\varepsilon_B + \beta\varepsilon_A)$-block-encoding of AB.

In Lemma 6, identity operators act on each other's auxiliary qubits, slightly abusing the notation. Formally, $I_b \otimes U_A := I_b \otimes \sum_{i,j=1}^{2^a} |i\rangle\langle j| \otimes (U_A)_{ij}$, while $I_a \otimes U_B := \sum_{i,j=1}^{2^b} |i\rangle\langle j| \otimes I_a \otimes (U_B)_{ij}$.

Lemma 7. (Hadamard product of block-encodings – Theorem 4,[49]). Let U_A be a $(\alpha, a, \varepsilon_A)$-block-encoding of A and U_B be a $(\beta, b, \varepsilon_B)$-block-encoding of B, where A, B are n-qubit operators. Then

$$(P \otimes I_{a+b}) U_A \otimes U_B (P^\dagger \otimes I_{a+b}) \tag{20}$$

is a $(\alpha\beta, a+b+n, \alpha\varepsilon_B + \beta\varepsilon_A)$-block-encoding of $A \circ B$. Here U_A and U_B are the block-encodings of A and B respectively and $P := \sum_{i,j=1}^N |i\rangle\langle i| \otimes |i \oplus j\rangle\langle j|$ can be constructed using n CNOT gates, namely one CNOT gate between each pair of corresponding qubits from the first and second registers.

Polynomial functions can be applied to the singular values of a block-encoded matrix (or eigenvalues of a Hermitian matrix) through quantum signal processing (QSP)[82,84] and QSVT[23]. For a more detailed account of QSVT and its applications in quantum algorithms, we refer the reader to refs. 23,25,85.

Lemma 8. (Constructing Chebyshev polynomials via QSP – Lemma 6,[23]). Let $T_d \in \mathbb{R}[x]$ be the d-th Chebyshev polynomial of the first kind. Let $\Phi \in \mathbb{R}^d$ be such that $\phi_1 = (1-d)\frac{\pi}{2}$, and for all $i \in [d] \backslash \{1\}$, let $\phi_i := \frac{\pi}{2}$. Then

$$\prod_{j=1}^d \left(e^{i\phi_j \sigma_z} R(x)\right) = \begin{bmatrix} T_d(x) & \cdot \\ \cdot & \cdot \end{bmatrix}, \text{ where } R(x) := \begin{bmatrix} x & \sqrt{1-x^2} \\ \sqrt{1-x^2} & -x \end{bmatrix}, \tag{21}$$

is a $(1, 1, 0)$-block-encoding of $T_d(x)$.

Theorem 9. (Polynomial Eigenvalue Transformation – Theorem 31,[23]). Let U be an (α, a, ε)-encoding of a Hermitian matrix A and $P \in \mathbb{R}[x]$ be a degree-d polynomial satisfying $|P(x)| \leq \frac{1}{2}$ on $[-1, 1]$. Then, one can construct a quantum circuit $\widetilde{U}$ which is a $(1, a+2, 4d\sqrt{\varepsilon/\alpha})$-encoding of $P(A/\alpha)$. $\widetilde{U}$ consists of d U and $U^\dagger$ gates, one controlled-U, and $\mathcal{O}((a+1)d)$ other one- and two-qubit gates.

Note that for Chebyshev polynomials, the $a+1$ auxiliary qubits are sufficient, and the $|P(x)| \leq \frac{1}{2}$ constraint is relaxed since $|T_d(x)| \leq 1$.

CHEB-QKAN construction

In the following, we present a detailed construction of CHEB-QKAN. Our model takes as input a real N-dimensional vector $\vec{x} = (x_1, x_2, \ldots, x_N) \in [-1, 1]^N$. We encode and process this input data on the diagonal of a matrix, assuming it is provided as a diagonal $(1, a_x, \varepsilon_x)$-block-encoding U_x such that $\| \langle 0|_a U_x |0\rangle_a - \mathrm{diag}(x_1, \ldots, x_N) \| \leq \varepsilon_x$ (Note that if U is an (α, a, ε)-block-encoding of A, then equivalently it is a $(1, a, \frac{\varepsilon}{\alpha})$-block-encoding of $\frac{A}{\alpha}$. Therefore, we can factor α into the input vector $\vec{x}$ and restrict the discussion to $\alpha = 1$.). For example, the input could be the amplitudes of a quantum state: given access to an amplitude-encoding unitary U where $U|0\rangle_n = \sum_{i=1}^N x_i |i\rangle_n$ with $x_i \in [-1, 1]$, we can efficiently construct a diagonal $(1, n+2, 0)$-block-encoding of $\{x_i\}$, as detailed in Lemma 10. Alternatively, techniques for constructing exact block-encodings for sparse matrices may also be employed[86]. Since such constructions may not be diagonal, one first removes the off-diagonal entries of the constructed block-encoding via Lemma 11. In the following, we detail the steps to build a CHEB-QKAN layer.

Suppose that the layer has K output nodes. Then, we need NK different parametrized activation functions, one between every two input and output nodes. To accommodate this, we first dilate the input block-encoding by appending $k = \log_2 K$ auxiliary qubits (Supplementary Note 1). This produces a $(1, a_x, \varepsilon_x)$-block-encoding of

$$\mathrm{diag}(\underbrace{x_1, \ldots, x_1}_{K}, \ldots, \underbrace{x_N, \ldots, x_N}_{K}) = \sum_{p=1}^N \sum_{q=1}^K x_p |p\rangle\langle p|_n \otimes |q\rangle\langle q|_k. \tag{22}$$

The obtained block-encoding $U_x \otimes I_k$ serves as the foundation for subsequent operations, as illustrated in Fig. 5.

Our trainable activation functions are linear combinations of Chebyshev polynomials of the first kind because these polynomials can be natively realized using QSVT[82]. To implement these polynomials, we alternately apply U_x and $U_x^\dagger$, interleaved by reflection operators according to Lemma 8. By utilizing Theorem 9, we obtain a $(1, a_x + 1, 4r\sqrt{\varepsilon_x})$-block-encoding of the diagonal matrix

$$\mathrm{diag}(\underbrace{T_r(x_1), \ldots, T_r(x_1)}_{K}, \ldots, \underbrace{T_r(x_N), \ldots, T_r(x_N)}_{K}) = \sum_{p=1}^N \sum_{q=1}^K T_r(x_p) |p\rangle\langle p|_n \otimes |q\rangle\langle q|_k, \tag{23}$$

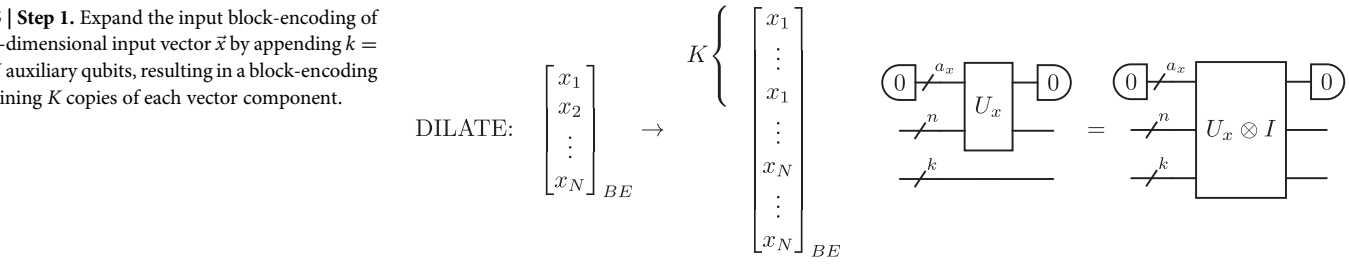

Fig. 5 | Step 1. Expand the input block-encoding of the N-dimensional input vector $\vec{x}$ by appending $k = \log_2 K$ auxiliary qubits, resulting in a block-encoding containing K copies of each vector component.

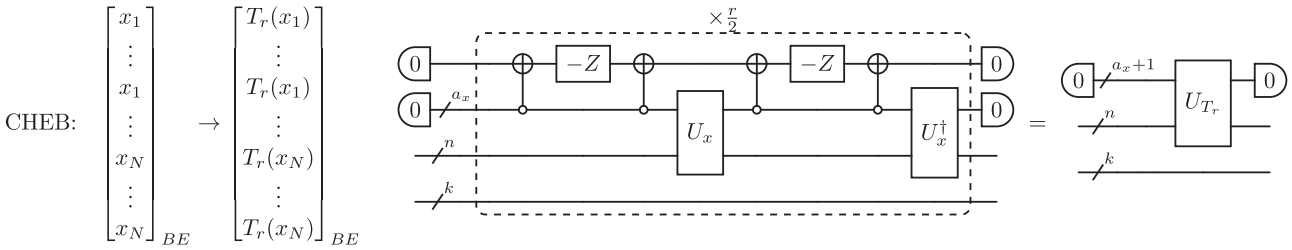

Fig. 6 | Step 2. Apply Chebyshev polynomials of degrees r ∈ {1, …, d} to the diagonal block-encoding from Step 1 by interleaving the input block-encoding Ux and its adjoint with reflection operators. For even r, apply $(U_x^\dagger Z_\pi U_x Z_\pi)^{\frac{r}{2}}$; for odd r, apply $U_x Z_\pi (U_x^\dagger Z_\pi U_x Z_\pi)^{\lfloor \frac{r}{2} \rfloor}$[98]. The k auxiliary qubits from Step 1 remain unused in this construction, serving only to maintain the expanded dimension of the block-encoding.

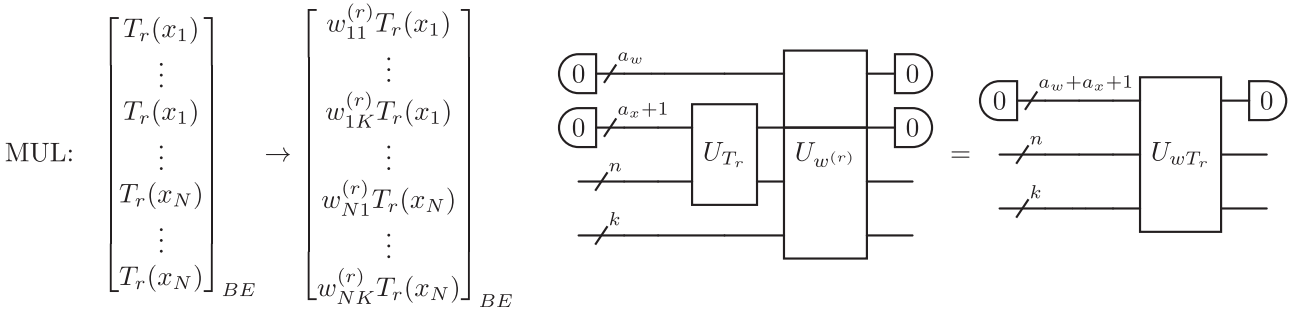

Fig. 7 | Step 3. Multiply the block-encoded Chebyshev polynomials from Step 2 by an NK-dimensional weight vector. The weight $w_{pq}^{(r)}$ corresponds to the coefficient in front of the r-th Chebyshev polynomial in the activation function ϕ_{pq}. In the circuit, the respective block-encodings do not "overlap" on their auxiliary qubits and, therefore, the $a_x + 1$ wire goes "through" $U_{w^{(r)}}$.

using $\mathcal{O}(r)$ applications of U_x and $U_x^\dagger$ and a single auxiliary qubit, as illustrated in Fig. 6. We denote this resulting block-encoding by U_{T_r}, which serves as the building block for our activation functions.

We repeat the previous to prepare separate block-encodings for each of the $d + 1$ Chebyshev polynomials, with degrees r ranging from 0 to d. Note that $T_0(x) = 1$ is trivial and corresponds to identity. Therefore, a general linear combination requires a total of $(d + 1)NK$ linear coefficients. For this construction, we assume the weights are provided in the form of $d + 1$ diagonal $(1, a_w, \varepsilon_w)$-block-encodings of NK-dimensional real weight vectors $\vec{w}_r$, denoted by U_{w_r}. We intentionally defer the discussion of the precise construction of these block-encodings to parametrization "Parametrization of the QKAN learning model", in order to maintain the generality of our approach. Applying Lemma 6, we multiply each Chebyshev polynomial encoding by its respective weight block-encoding, as illustrated in Fig. 7. This yields a $(1, a_x + 1 + a_w, 4r\sqrt{\varepsilon_x} + \varepsilon_w)$-block-encoding for each diagonal matrix:

$$\begin{aligned}&\mathrm{diag}\Big(\underbrace{w_{11}^{(r)}T_r(x_1),\ldots,w_{1K}^{(r)}T_r(x_1)}_{K},\ldots,\underbrace{w_{N1}^{(r)}T_r(x_N),\ldots,w_{NK}^{(r)}T_r(x_N)}_{K}\Big)\\&=\sum_{p=1}^{N}\sum_{q=1}^{K}w_{pq}^{(r)}T_r(x_p)|p\rangle\langle p|_n\otimes|q\rangle\langle q|_k.\end{aligned}\tag{24}$$

Here, $w_{pq}^{(r)}$ is the weight of the r-th Chebyshev polynomial in the activation function between the p-th input node and the q-th output node. We must have $|w_{pq}^{(r)}| \le 1$ for all p, q, r due to unitarity.

Finally, we combine the $d + 1$ weighted block-encodings of Chebyshev polynomials by taking the linear combination of block-encodings with an equal superposition using Lemma 5, as illustrated in Fig. 8. This process yields the desired $(1, a_x + 1 + a_w + \log_2(d + 1), 4d\sqrt{\varepsilon_x} + \varepsilon_w)$-block-encoding of the diagonal matrix containing NK activation functions:

$$\begin{aligned}&\mathrm{diag}\Big(\underbrace{\phi_{11}(x_1),\ldots,\phi_{1K}(x_1)}_{K},\ldots,\underbrace{\phi_{N1}(x_N),\ldots,\phi_{NK}(x_N)}_{K}\Big)\\&=\sum_{p=1}^{N}\sum_{q=1}^{K}\phi_{pq}(x_p)|p\rangle\langle p|_n\otimes|q\rangle\langle q|_k.\end{aligned}\tag{25}$$

Here, $\phi_{pq}(x)$ denotes the activation function between the pth input node and the q-th output node:

$$\phi_{pq}(x) := \frac{1}{d+1}\sum_{r=0}^{d} w_{pq}^{(r)}T_r(x_p)\tag{26}$$

The LCU procedure requires $\log_2(d + 1)$ auxiliary qubits, assuming $d + 1$ is a power of two, with the equal superposition created by applying $H_{\log_2(d+1)}$.

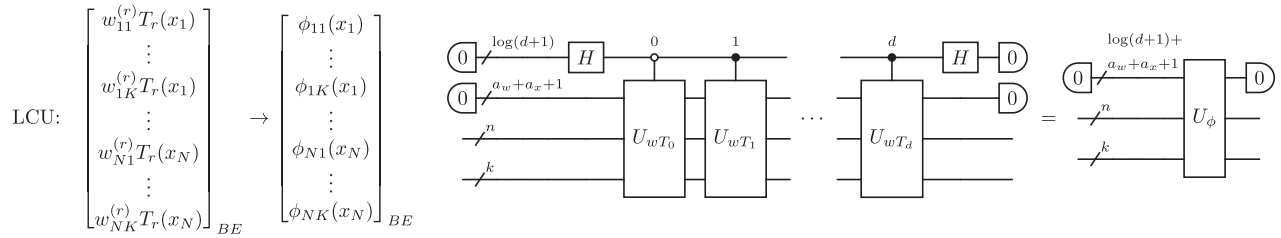

Fig. 8 | Step 4. Add the (d + 1) block-encodings of weighted Chebyshev polynomials obtained in Step 3 using LCU. This requires $\log_2(d+1)$ control qubits, which are initialized in an equal superposition state using multi-qubit Hadamard gates.

In "Parametrization of the QKAN learning model", we consider generalizing this step by replacing $H_{\log_2(d+1)}$ with a parametrized unitary to control the contribution of each basis function. We denote the obtained block-encoding by U_ϕ.

In the final step of the construction, we want to produce a diagonal block-encoding corresponding to the output of the CHEB-QKAN layer, as illustrated in Fig. 9. Starting from the block-encoding U_ϕ of individual activation functions obtained in the previous step, we apply a layer of Hadamards H_n on the n 'input' qubits, corresponding to the summation of N inputs for each of the K output nodes:

$$\left(H_n \otimes I_k\right)\left(\sum_{p=1}^{N}\sum_{q=1}^{K}\phi_{pq}(x_p)\left|p\right\rangle\left\langle p\right|_n \otimes \left|q\right\rangle\left\langle q\right|_k\right)\left(H_n \otimes I_k\right). \tag{27}$$

This results in a block-encoding of a matrix whose diagonal elements hold the desired summation:

$$\begin{aligned}&\langle 0|_n \otimes I_k\left(\sum_{p=1}^{N}\sum_{q=1}^{K}\phi_{pq}(x_p)H_n\left|p\right\rangle\left\langle p\right|_n H_n \otimes \left|q\right\rangle\left\langle q\right|_k\right)|0\rangle_n \otimes I_k \\ &= \sum_{q=1}^{K}\left(\frac{1}{N}\sum_{p=1}^{N}\phi_{pq}(x_p)\right)\left|q\right\rangle\left\langle q\right|_k.\end{aligned} \tag{28}$$

In particular, the obtained unitary, denoted by U_Φ, is a $(1, a_x + 1 + a_w + \log_2(d+1) + n, 4d\sqrt{\varepsilon_x} + \varepsilon_w)$ block-encoding of the diagonal matrix

$$\mathrm{diag}\left(\frac{1}{N}\sum_{p=1}^{N}\phi_{p1}(x_p), \ldots, \frac{1}{N}\sum_{p=1}^{N}\phi_{pK}(x_p)\right), \tag{29}$$

since

$$\begin{aligned}&\| \langle 0|_n\left(H_n \otimes I_k\right)\mathrm{diag}(\phi_{11}, \ldots, \phi_{NK})\left(H_n \otimes I_k\right)|0\rangle_n - \langle 0|_n\left(H_n \otimes I_k\right)\langle 0|_{\mathrm{aux}}U_\phi|0\rangle_{\mathrm{aux}}\left(H_n \otimes I_k\right)|0\rangle_n \| \\ &= \left\|\mathrm{diag}\left(\tfrac{1}{N}\sum_{p=1}^{N}\phi_{p1}(x_p), \ldots, \tfrac{1}{N}\sum_{p=1}^{N}\phi_{pK}(x_p)\right) - \tfrac{1}{N}\sum_{p=1}^{N}\left(\left\langle p\right|_n \otimes I_k\right)\langle 0|_{\mathrm{aux}}U_\phi|0\rangle_{\mathrm{aux}}\left(\left|p\right\rangle_n \otimes I_k\right)\right\| \\ &\leq \max_{p,q}|\phi_{pq}(x_p) - \left\langle p\right|_n\left\langle q\right|_k\langle 0|_{\mathrm{aux}}U_\phi|0\rangle_{\mathrm{aux}}\left|p\right\rangle_n\left|q\right\rangle_k| \\ &\leq 4d\sqrt{\varepsilon_x} + \varepsilon_w.\end{aligned} \tag{30}$$

Notice that we have absorbed the n "input" qubits into the auxiliary register.

Input block-encoding for the learning model

We formalize the two proposed methods for efficient diagonal block-encoding of the input vector.

Lemma 10. (Diagonal block-encoding of amplitudes – Theorem 2,[59]). Given an n-qubit quantum state specified by a state preparation unitary U, such that $\left|\psi\right\rangle_n = U|0\rangle_n = \sum_{j=1}^{N}\psi_j\left|j\right\rangle_n$ (with $\psi_j \in \mathbb{C}$), we can prepare a (1, n + 2, 0)-block-encoding U_A of the diagonal matrix $A = \mathrm{diag}(\mathrm{Re}(\psi_1), \ldots, \mathrm{Re}(\psi_N))$ with $O(n)$ circuit depth and a total of $O(1)$ queries to a controlled-U gate.

Lemma 11. (Removing off-diagonal matrix elements). Let U_A be an (α, a, ε)-block-encoding for an n-qubit operator A. Then

$$\left(I_a \otimes P\right)U_A \otimes I_n\left(I_a \otimes P\right) \tag{31}$$

is a (α, a + n, ε)-block-encoding of diag(A_{11}, …, A_{NN}), where $P = \sum_{i,j=1}^{N}|i\rangle\langle i| \otimes \left|i \oplus j\right\rangle\left\langle j\right|$.

Proof. Noting that $\mathrm{diag}(A_{11}, \ldots, A_{NN}) = A \circ I_n$, this follows immediately from Lemma 7, where $A \leftarrow A$ and $B \leftarrow I_n$. Note that I_a acts on the a auxiliary qubits of U_A and that, trivially, I_n is a (1, 0, 0)-block-encoding of itself. □

Parametrization of the QKAN learning model

The parameterization of QKAN depends on being able to encode the relevant parameters into a diagonal block-encoding and update it. Rather than classically computing and reconstructing the block-encoding at each iteration – which can be costly or technically challenging[86] – we combine parametrized quantum circuits (PQCs)[38] with our block-encoding propositions in "Input block-encoding for the learning model" to enable efficient updates in the *MUL* step of "CHEB-QKAN construction". We first provide two methods to parameterize the diagonal block-encodings of the weight vectors $\vec{w}$ used in the *MUL* step in "CHEB-QKAN construction", which correspond to different norm constraints, namely $\|\vec{w}\|_1 \leq 1$, $\|\vec{w}\|_2 \leq 1$, and $\|\vec{w}\|_\infty \leq 1$.

The first method encodes the real parts of the amplitudes of a parametrized state

$$\left|w(\vec{\theta})\right\rangle_{n+k} = U(\vec{\theta})|0\rangle_{n+k} \tag{32}$$

as a diagonal $(1, n+k+2, 0)$-block-encoding via Lemma 10. The step can be done efficiently using $O(1)$ queries to the controlled-$U(\vec{\theta})$ gate and its adjoint version. The ℓ_2 normalization the pure quantum state $\left|w(\vec{\theta})\right\rangle_{n+k}$ implies a strict equality constraint $\|\vec{w}\|_2 = 1$. However, by encoding only the real parts by Lemma 10, the constraint on the weights is relaxed to an inequality $\|\mathrm{Re}(\vec{w})\|_2 \leq 1$. Such regularization is utilized in classical machine learning to prevent overfitting[87,88], but on the other hand, it may limit the expressibility of the QKAN model, which can be hypothesized from the effects of ℓ_2 upper bounds of weights on Rademacher complexity[89].

Alternatively, we can impose an ℓ_1 regularization by encoding the real values of the state onto the diagonal as described above, and then squaring it by applying the block-encoding twice (Lemma 6) to create a block-encoding with $\|\vec{w}^2\|_1 \leq 1$. This, however, places the weight within a probability simplex since $w_j^2 \geq 0$. To generalize the parameterization from the simplex to the ℓ_1 ball, we can multiply the simplex-constrained block-encoding by an alternatively parametrized block-encoding with $\{1, -1\}$ on its diagonal, which can be prepared by applying the QSVT-approximated sign function to another parametrized block-encoding. We note that there is a limitation to the precision of the approximation of the sign function when the

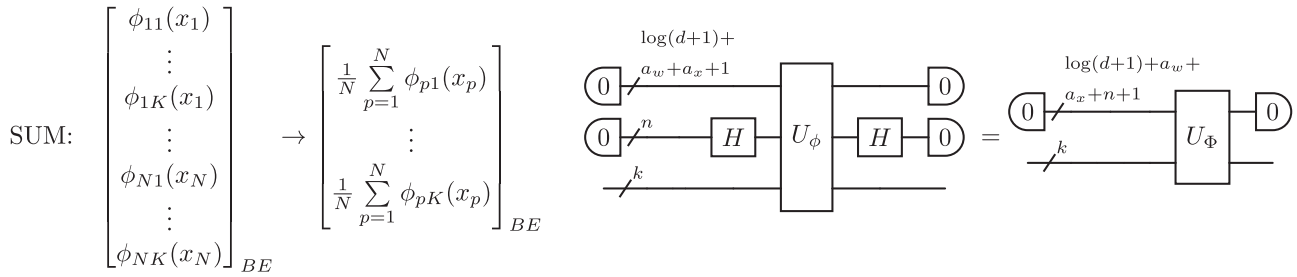

Fig. 9 | Step 5. Sum the individual activation functions over N input nodes for each output node, creating the desired diagonal block-encoding UΦ of the K-dimensional output vector. This is achieved by sandwiching the block-encoding from Step 4 with two n-qubit Hadamard gates. The dimension reduction occurs as the n qubits originally used for input block-encoding are moved to the auxiliary register.

parameters are close to 0, and we leave further improvements on such parameterizations to future work.

The second method is to utilize Lemma 7 and replace *MUL* with a Hadamard product between a parametrized unitary (or parametrized block-encoding) and our prepared diagonal block-encoding U_{T_r} of the r-th Chebyshev polynomial. A parametrized unitary $U(\vec{\theta})$ is a $(1, 0, 0)$-block-encoding of itself. As the off-diagonal elements of the diagonal block-encoding U_{T_r} are zero, so is the Hadamard product

$$U(\vec{\theta}) \circ \langle 0|_{\text{aux}} U_{T_r} |0\rangle_{\text{aux}}, \tag{33}$$

so the result remains a diagonal block-encoding. We then effectively create a parametrized diagonal. We conjecture that the diagonal block-encoding produced by this second method can be more expressive compared to the first method because, in principle, all diagonal entries of $U(\vec{\theta})$ could be in the range of $[-1, 1]$ such that the ℓ_∞ norm of the weights is upper bounded by 1: $\|\vec{w}\|_\infty \leq 1$. This provides a much weaker regularization effect on the weights compared to the ℓ_2 and ℓ_1 regularizations in the first method, which would result in higher expressibility while retaining a regularization effect.

Given that, for the purpose of this paper, we assume that all the weights in our QKAN are real, we must restrict the diagonal entries of $U(\vec{\theta})$ to be real. In principle, one can limit the use of PQCs to consist of only real gates, e.g., RY, CNOT, and CZ gates. However, such limitations would greatly restrict the expressive power of PQCs as they would be confined to orthogonal transformations. Alternatively, one can prepare the adjoint $U(\vec{\theta})^\dagger$ of the PQC and combine the two using LCU, so that the diagonal entries of the resulting matrix, $\frac{1}{2}(U(\vec{\theta})^\dagger + U(\vec{\theta}))$, are real. Finally, the parameterization can be further augmented by composing the constructions outlined above, for example, by taking sums, products, or applying polynomial functions to the parametrized diagonal block-encoding as discussed in "Block-encoding and quantum subroutines".

Thus far, we have limited the parameterization of QKAN to the weight vectors. However, parameterizing other steps of QKAN could be advantageous. For instance, in the *LCU* step in "CHEB-QKAN construction", instead of using Hadamard transforms, we can parameterize the unitaries used as state preparation pairs (Definition 6) to control the global weights of Chebyshev polynomials of different degrees. With a sufficiently expressive Ansatz, a training strategy could involve iteratively adding higher-degree Chebyshev polynomials to the sum and optimizing their global weights. By inspecting the optimized weights, one could determine the optimal number of Chebyshev polynomials, e.g., if the weight of the newly added polynomial vanishes. Furthermore, while we focus on Chebyshev polynomials, the QSVT framework allows implementing any bounded polynomial using Theorem 9, and many functions of interest can be well approximated by polynomials in the QSVT framework[23]. This, of course, includes splines, which can be approximated via a polynomial approximation of the erf function, as mentioned in "Quantum Kolmogorov-Arnold Network (QKAN)". While a fixed basis can be selected, in principle, even the angles of QSVT could be made trainable parameters, which would allow the selection of basis functions to be part of learning as well.

Training of the QKAN learning model

The parameterization methods in "Parametrization of the QKAN learning model" remove the cost of constructing a new circuit that implements a diagonal block-encoding, given that we only need to update the parameters in the PQC. Considering the full circuit with the Hadamard test and amplitude estimation, we can view the entire circuit as a very large PQC that has parameters that are repeated multiple times throughout the circuit. Given this construction, we note that the same parameter would be repeated throughout the circuit due to sequential repetitions of circuit blocks from QSVT and amplitude estimation. To achieve the analytical derivative from parameter-shift rules[36,90,91], we note that from the product rule, the gradient can be obtained from the sum of gradients of individual sub-terms, and thus can be found by computing the sum of the parameter-shifted circuits[36,90,91] of gates that share the same classical parameter. Therefore, the evaluation of the gradient via parameter shift rule would then cost $\mathcal{O}(d)$ queries for single layer QKANs, and $\mathcal{O}(d^{2L})$ for L-layer QKANs. Note that the number of queries required to evaluate the gradient of a single parameter in the QKAN architecture also grows exponentially with the number of layers.

Given that it is costly to obtain the full analytical gradient, we can make use of gradient estimation techniques to achieve a much more efficient estimation. Instead of perturbing each occurrence of the variable individually, one can obtain an estimate of the gradient by finite difference methods, which would only require a perturbation of shared variables once. By extension, one can also use simultaneous perturbation stochastic approximation (known as SPSA)[92] to produce gradient estimates with a cost unrelated to the number of parameters (both free and repeated) to achieve a much more efficient training strategy.

The circuit parameters can then be updated using optimizer algorithms such as gradient descent or Adam[93]. Further, one can also incorporate quantum natural gradient methods[94,95] to achieve faster convergence by again using parameter shift rules or, with a constant cost, SPSA, to compute the quantum Fisher information matrix[96].

Interpretability of the QKAN learning model

Interpretability of KANs, as formalized in ref. 5, refers to identifying and pruning unimportant branches of the model. In QKAN, this can be achieved in a manner consistent with our block encoding parametrization: the same parametrized quantum state that defines the weights also provides sample access to their relative importance.

Consider the first method in "Parametrization of the QKAN learning model" where the weights are encoded from the real amplitudes of $|\psi(\vec{\theta})\rangle = U(\vec{\theta})|0\rangle$, i.e., $w_j = \text{Re}(\psi_j(\vec{\theta}))$. To obtain sampling access consistent with the encoded weights, we use a standard LCU combination of $U(\vec{\theta})$ and $U(\vec{\theta})^\dagger$ and a single ancilla qubit to prepare the state

$$|0\rangle\big|\text{Re}(\psi)\big\rangle + |1\rangle\big|\text{Im}(\psi)\big\rangle, \tag{34}$$

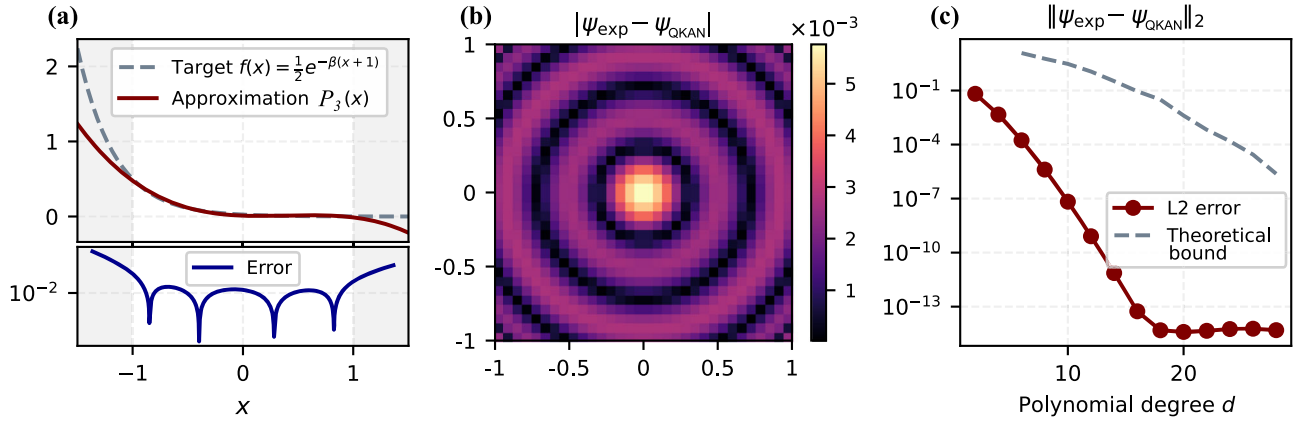

Fig. 10 | Numerical illustration of Gaussian state preparation via QKAN. **a** Degree-3 polynomial $P_3(x)$ approximating $\frac{1}{2}e^{-\beta(x+1)}$ on $[-1, 1]$. **b** Absolute amplitude error $|\psi_{exp}(i,j) - \psi_{QKAN}(i,j)|$ of the normalized 2D Gaussian state prepared using P_3. **c** ℓ_2-error between the prepared and target states as a function of polynomial degree d, compared to the theoretical bound from Supplementary Note 4.B; the empirical error decreases exponentially until saturating at machine precision.

where $|\mathrm{Re}(\psi)\rangle = \sum_j \mathrm{Re}(\psi_j)|j\rangle$. By measuring this state in the computational basis and postselecting on $|0\rangle$ in the first register, we sample indices with probability

$$\Pr[j] = |\mathrm{Re}(\psi_j(\vec{\theta}))|^2 = |w_j|^2. \tag{35}$$

The postselection succeeds with probability $p_{succ} = \sum_j w_j^2 = \|\vec{w}\|_2^2$, which, if desired, can be increased via amplitude amplification, yielding a sampling distribution arbitrarily close to $\Pr[j] = w_j^2/\|\vec{w}\|_2^2$. As a result, indices with large $|w_j|$ are sampled more frequently, while small weights are further suppressed when squared. Thus, basis functions with vanishing weights will have low sampling probability and can be pruned after the training is completed. Therefore, by sampling the trained state-preparation circuits, we can obtain a compressed and pruned model of the trained QKAN that can be explicitly interpreted. If one additionally requires the signed values w_j for selected indices j, they can be estimated using a standard Hadamard-test combined with amplitude estimation, analogous to the output-estimation step described in "Application I: Quantum learning model".

Finally, classical KANs support sparsity via the ℓ_1 regularization. To mirror this while preserving sampling access, we can impose the $\|\vec{w}\|_1 \le 1$ regularization via coherent squaring of the real amplitudes, as described in "Parametrization of the QKAN learning model".

Numerical illustration of Gaussian state preparation via QKAN

To complement our theoretical analysis, we provide a concise numerical example illustrating Gaussian state preparation via QKAN and highlighting several practical considerations. As discussed in "Application II: Multivariate state preparation", the only approximation in our two-layer construction arises in the second layer, where the exponential decay $e^{-\beta(x+1)}$ is implemented using a Chebyshev polynomial. The first layer, which prepares the D-dimensional grid and evaluates low-degree Chebyshev polynomials, is exact and incurs no approximation or block-encoding error. Supplementary Note 4.B provides a worst-case bound on the approximation error of the exponential; here we examine how the approximation behaves numerically and how it affects the resulting state.

We specialize to $D = 2$ and consider a 32×32 grid corresponding to $n = 5$ qubits per dimension, setting $\beta = 6$. This 2D example is also illustrated in Fig. 4. The first QKAN layer computes

$$z_{(i,j)} = \frac{1}{2}\left[T_2(x_i) + T_2(y_j)\right] = x_i^2 + y_j^2 - 1, \tag{36}$$

exactly at each grid point, producing a diagonal operator that encodes a (shifted) squared radius on the grid.

The second layer approximates $x \mapsto e^{-\widetilde{\beta}(x+1)}$ with $\widetilde{\beta} = \beta/2$ using a degree-d polynomial $P_d(x)$. The standard QSVT constructions require polynomials of definite parity, so we first decompose the target function into even and odd components,

$$f(x) = \frac{1}{2}\left[f_{even}(x) + f_{odd}(x)\right] = \frac{1}{2}\left[(f(x) + f(-x)) + (f(x) - f(-x))\right], \tag{37}$$

approximate each component by a truncated Taylor series of degree d, and combine them via an equal-superposition LCU using one additional ancilla qubit, as in Theorem 9. To satisfy the amplitude constraints $|P_d(x)| \le \frac{1}{2}$ needed for Theorem 9, we rescale the target function to $\frac{1}{2}e^{-\beta(x+1)}$; this introduces only a constant-factor overhead in the final amplitude-amplification step. Finally, to implement the resulting polynomials using QSVT, we compute the required phase angles numerically using the `pyqsp` Python package[25].

Figure 10(a) shows the resulting degree-3 polynomial $P_3(x)$ approximating $\frac{1}{2}e^{-\widetilde{\beta}(x+1)}$. The approximation is accurate on the interval $[-1, 1]$, as required for QKAN, and deteriorates outside this range. Applying the polynomial to the output of the first layer yields a diagonal operator whose entries approximate the target Gaussian values. Figure 10(b) displays the absolute error in the normalized two-dimensional Gaussian state prepared using this degree-3 polynomial. For each grid point (i, j) we plot $|\psi_{exp}(i,j) - \psi_{QKAN}(i,j)|$, where ψ_{QKAN} is the normalized output state and ψ_{exp} is the ideal Gaussian. The error is largest near the center of the distribution. This matches the behavior in Fig. 10 (a): the center corresponds to the region where x is close to -1 and the polynomial approximation error is maximal.

Finally, Fig. 10c shows the ℓ_2 error, $\|\psi_{exp} - \psi_{QKAN}\|_2$, as a function of the polynomial degree d, together with the theoretical bound from Supplementary Note 4.B. The empirical error decays exponentially with d, in agreement with the lemma, and plateaus at $d = 20$ when the error reaches machine precision of 10^{-14}–10^{-15}.

Generalized state preparation via CHEB-QKAN

When discussing multivariate state preparation in "Application II: Multivariate state preparation" we assumed the input register $\vec{x}$ encoded a regular D-dimensional grid to prepare multivariate distributions on a grid. We now generalize this to an arbitrary input by showing that a CHEB-QKAN layer implements a general multivariate state-preparation routine on any block-

encoded input vector $\vec{x}$. Theorem 12 guarantees that we can prepare a quantum state with amplitudes matching the entries of an arbitrary K-dimensional CHEB-QKAN layer, as long as the input and weights can be block-encoded efficiently. The proof is deferred to Supplementary Note 5 and proceeds by applying the block-encoding to the uniform superposition, followed by amplitude amplification.

Theorem 12. (Multivariate state preparation via CHEB-QKAN). Let $\varepsilon \in (0, \frac{1}{2})$. We are given access to a controlled diagonal $(1, a_x, \varepsilon_x)$-block-encoding U_x of an input vector $\vec{x} \in [-1, 1]^N$, and access to $d+1$ controlled diagonal $(1, a_w, \varepsilon_w)$-block-encodings $U_{w^{(r)}}$ of weight vectors $\vec{w}^{(r)} \in [-1, 1]^{NK}$. Let $\mathcal{N}^2 := \sum_{q=1}^{K} \left(\frac{1}{N}\sum_{p=1}^{N} \phi_{pq}(x_p)\right)^2$ and d be the maximal degree of Chebyshev polynomials used in the parameterization of activation functions ϕ_{pq}. If $\varepsilon_x \leq \frac{\mathcal{N}^2}{144Kd^2}\varepsilon^2$ and $\varepsilon_w \leq \frac{\mathcal{N}}{3\sqrt{K}}\varepsilon$, then we can prepare a ℓ_2 normalized quantum state $|\psi\rangle$ with amplitudes corresponding to a $CHEB-QKAN$ layer such that

$$\left\| |\psi\rangle - \frac{1}{\mathcal{N}} \sum_{q=1}^{K} \left(\frac{1}{N} \sum_{p=1}^{N} \phi_{pq}(x_p) \right) |q\rangle_k \right\|_2 \leq \varepsilon, \tag{38}$$

The procedure succeeds with arbitrarily high probability by using $\mathcal{O}(\sqrt{K}d^2/\mathcal{N})$ applications of controlled-U_x and controlled-$U_{w^{(r)}}$ and their adjoint versions.

Such a strategy can be viewed as a multivariate extension of the nonlinear amplitude transformation procedure outlined by Guo et al.[97] because the amplitudes of the prepared state are multivariate functions of the input vector. In their approach, a nonlinear amplitude transformation unitary is applied to a uniform superposition state, followed by amplitude amplification. On the other hand, achieving a multivariate version of exponential improvement through importance sampling, as proposed by Rattew and Rebentrost[59], remains an open problem, due to the challenges associated with implementing importance sampling in the multivariate setting.

Data availability

No datasets were generated or analysed during the current study.

Code availability

This manuscript does not report any new code. All algorithmic details necessary for implementation are described explicitly in the manuscript.

Received: 21 October 2024; Accepted: 10 February 2026;
Published online: 11 March 2026

References

1. Kolmogorov, A. On the representation of continuous functions of several variables by superpositions of continuous functions of a smaller number of variables. *Dokl. Akad. Nauk SSSR* **108**, 179–182 (1956).
2. Kolmogorov, A. On the representation of continuous functions of many variables by superposition of continuous functions of one variable and addition. *Dokl. Akad. Nauk SSSR* **114**, 953–956 (1957).
3. Arnold, V. On functions of three variables. *Dokl. Akad. Nauk SSSR* **114**, 679–681 (1957).
4. Arnold, V. On the representation of continuous functions of three variables by superpositions of continuous functions of two variables. *Mat. Sb. (N. S.)* **48**, 3–74 (1959).
5. Liu, Z. et al. KAN: Kolmogorov-Arnold networks. In *The Thirteenth International Conference on Learning Representations* https://openreview.net/forum?id=Ozo7qJ5vZi (2025).
6. Krizhevsky, A., Sutskever, I. & Hinton, G. E. Imagenet classification with deep convolutional neural networks. In Pereira, F., Burges, C., Bottou, L. & Weinberger, K. (eds.) *Advances in Neural Information Processing Systems*, vol. 25 (Curran Associates, Inc., 2012). https://proceedings.neurips.cc/paper_files/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html.
7. Goodfellow, I., Bengio, Y. & Courville, A.*Deep Learning* (MIT Press, 2016). http://www.deeplearningbook.org.
8. LeCun, Y., Bengio, Y. & Hinton, G. Deep learning. *Nature* **521**, 436–444 (2015).
9. He, K., Zhang, X., Ren, S. & Sun, J. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770–778 https://doi.org/10.1109/CVPR.2016.90 (2016).
10. Liu, Z., Tegmark, M., Ma, P., Matusik, W. & Wang, Y. Kolmogorov-Arnold networks meet science. *Phys. Rev. X* **15**, 041051 (2025).
11. Bodner, A. D., Tepsich, A. S., Spolski, J. N. & Pourteau, S. Convolutional Kolmogorov-Arnold networks. Preprint at https://doi.org/10.48550/arXiv:2406.13155 (2024).
12. Kiamari, M., Kiamari, M. & Krishnamachari, B. GKAN: Graph Kolmogorov-Arnold networks. Preprint at https://doi.org/10.48550/arXiv:2406.06470 (2024).
13. Carlo, G. D., Mastropietro, A. & Anagnostopoulos, A. Kolmogorov-Arnold graph neural networks. Preprint at https://doi.org/10.48550/arXiv:2406.18354 (2024).
14. Sidharth, S., Keerthana, A., Gokul, R. & Anas, K. Chebyshev polynomial-based Kolmogorov-Arnold networks: An efficient architecture for nonlinear function approximation. Preprint at https://doi.org/10.48550/arXiv:2405.07200 (2024).
15. Kundu, A., Sarkar, A. & Sadhu, A. KANQAS: Kolmogorov-Arnold network for quantum architecture search. *EPJ Quantum Tech.* **11**, 76 (2024).
16. Bozorgasl, Z. & Chen, H. Wav-KAN: Wavelet Kolmogorov-Arnold networks. Preprint at https://doi.org/10.48550/arXiv:2405.12832 (2024).
17. Genet, R. & Inzirillo, H. A temporal Kolmogorov-Arnold transformer for time series forecasting. Preprint at https://doi.org/10.48550/arXiv:2406.02486 (2024).
18. Wang, Y. et al. Kolmogorov Arnold Informed neural network: A physics-informed deep learning framework for solving forward and inverse problems based on Kolmogorov Arnold networks. *Comput. Methods Appl. Mech. Eng.* **433**, 117518 (2025).
19. Aghaei, A. A. rKAN: Rational Kolmogorov-Arnold networks. Preprint at https://doi.org/10.48550/arXiv:2406.14495 (2024).
20. Xu, J. et al. FourierKAN-GCF: Fourier Kolmogorov-Arnold network – an effective and efficient feature transformation for graph collaborative filtering. Preprint at https://doi.org/10.48550/arXiv:2406.01034 (2024).
21. Aghaei, A. A. fKAN: Fractional Kolmogorov-Arnold networks with trainable Jacobi basis functions. *Neurocomputing* **623,** 129414 (2025).
22. Zhang, F. & Zhang, X. GraphKAN: Enhancing feature extraction with graph Kolmogorov Arnold networks. Preprint at https://doi.org/10.48550/arXiv:2406.13597 (2024).
23. Gilyén, A., Su, Y., Low, G. H. & Wiebe, N. Quantum singular value transformation and beyond: exponential improvements for quantum matrix arithmetics. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2019, 193–204 (Association for Computing Machinery, New York, NY, USA, 2019). https://doi.org/10.1145/3313276.3316366.
24. Rall, P. & Fuller, B. Amplitude estimation from quantum signal processing. *Quantum* **7**, 937 (2023).
25. Martyn, J. M., Rossi, Z. M., Tan, A. K. & Chuang, I. L. Grand unification of quantum algorithms. *PRX Quantum* **2**, 040203 (2021).
26. Gilyén, A. & Poremba, A. Improved quantum algorithms for fidelity estimation. Preprint at https://doi.org/10.48550/arXiv.2203.15993 (2022).
27. Wang, Q. et al. Quantum algorithm for fidelity estimation. *IEEE Trans. Inf. Theor.* **69**, 273–282 (2023).

28. Wang, Q., Guan, J., Liu, J., Zhang, Z. & Ying, M. New quantum algorithms for computing quantum entropies and distances. *IEEE Trans. Inf. Theor.* **70**, 5653–5680 (2024).
29. Li, J. & Tong, Y. Exponential quantum advantage for pathfinding in regular sunflower graphs. Preprint at https://doi.org/10.48550/arXiv.2407.14398 (2024).
30. Qiu, Y., Koor, K. & Rebentrost, P. The quantum Esscher transform. Preprint at https://doi.org/10.48550/arXiv.2401.07561 (2024).
31. Schuld, M. & Petruccione, F.*Machine Learning with Quantum Computers* (Springer Cham, 2021). https://doi.org/10.1007/978-3-030-83098-4.
32. Biamonte, J. et al. Quantum machine learning. *Nature* **549**, 195–202 (2016).
33. Beer, K. et al. Training deep quantum neural networks. *Nat. Commun.* **11**, 808 (2020).
34. Cerezo, M. et al. Variational quantum algorithms. *Nat. Rev. Phys.* **3**, 625–644 (2021).
35. Havlíček, V. et al. Supervised learning with quantum-enhanced feature spaces. *Nature* **567**, 209–212 (2019).
36. Mitarai, K., Negoro, M., Kitagawa, M. & Fujii, K. Quantum circuit learning. *Phys. Rev. A* **98**, 032309 (2018).
37. Farhi, E. & Neven, H. Classification with quantum neural networks on near term processors. Preprint at https://doi.org/10.48550/arXiv.1802.06002 (2018).
38. Benedetti, M., Lloyd, E., Sack, S. & Fiorentini, M. Parameterized quantum circuits as machine learning models. *Quantum Sci. Technol.* **4**, 043001 (2019).
39. Schuld, M. & Killoran, N. Quantum machine learning in feature Hilbert spaces. *Phys. Rev. Lett.* **122**, 040504 (2019).
40. Du, Y., Hsieh, M.-H., Liu, T. & Tao, D. Expressive power of parametrized quantum circuits. *Phys. Rev. Res.* **2**, 033125 (2020).
41. Abbas, A. et al. The power of quantum neural networks. *Nat. Comput. Sci.* **1**, 403–409 (2021).
42. Schuld, M., Sweke, R. & Meyer, J. J. Effect of data encoding on the expressive power of variational quantum-machine-learning models. *Phys. Rev. A* **103**, 032430 (2021).
43. Banchi, L., Pereira, J. & Pirandola, S. Generalization in quantum machine learning: A quantum information standpoint. *PRX Quantum* **2**, 040321 (2021).
44. Holmes, Z., Sharma, K., Cerezo, M. & Coles, P. J. Connecting Ansatz expressibility to gradient magnitudes and barren plateaus. *PRX Quantum* **3**, 010313 (2022).
45. Caro, M. C. et al. Generalization in quantum machine learning from few training data. *Nat. Commun.* **13**, 4919 (2022).
46. Pira, L. & Ferrie, C. On the interpretability of quantum neural networks. *Quantum Mach. Intell.* **6**, 52 (2024).
47. Rebentrost, P., Mohseni, M. & Lloyd, S. Quantum support vector machine for big data classification. *Phys. Rev. Lett.* **113**, 130503 (2014).
48. Kerenidis, I., Landman, J. & Prakash, A. Quantum algorithms for deep convolutional neural networks. In *International Conference on Learning Representations* https://openreview.net/forum?id=Hygab1rKDS (2020).
49. Guo, N. et al. Quantum transformer: Accelerating model inference via quantum linear algebra. Preprint at https://doi.org/10.48550/arXiv.2402.16714 (2024).
50. Allcock, J., Hsieh, C.-Y., Kerenidis, I. & Zhang, S. Quantum algorithms for feedforward neural networks. *ACM Trans. Quantum Comput.* **1**, 1–24 (2020).
51. Rebentrost, P., Bromley, T. R., Weedbrook, C. & Lloyd, S. Quantum Hopfield neural network. *Phys. Rev. A* **98**, 042308 (2018).
52. Amin, M. H., Andriyash, E., Rolfe, J., Kulchytskyy, B. & Melko, R. Quantum Boltzmann machine. *Phys. Rev. X* **8**, 021050 (2018).
53. Kapoor, A., Wiebe, N. & Svore, K. Quantum perceptron models. In Lee, D., Sugiyama, M., Luxburg, U., Guyon, I. & Garnett, R. (eds.) *Advances in Neural Information Processing Systems*, vol. 29 (Curran Associates, Inc., 2016). https://proceedings.neurips.cc/paper/2016/hash/d47268e9db2e9aa3827bba3afb7ff94a-Abstract.html.
54. Grover, L. K. Synthesis of quantum superpositions by quantum computation. *Phys. Rev. Lett.* **85**, 1334–1337 (2000).
55. Plesch, M. & Brukner, Č Quantum-state preparation with universal gate decompositions. *Phys. Rev. A* **83**, 032302 (2011).
56. Sanders, Y. R., Low, G. H., Scherer, A. & Berry, D. W. Black-box quantum state preparation without arithmetic. *Phys. Rev. Lett.* **122**, 020502 (2019).
57. Zhang, X.-M., Li, T. & Yuan, X. Quantum state preparation with optimal circuit depth: Implementations and applications. *Phys. Rev. Lett.* **129**, 230504 (2022).
58. McArdle, S., Gilyén, A. & Berta, M. Quantum state preparation without coherent arithmetic. Preprint at https://doi.org/10.48550/arXiv.2210.14892 (2022).
59. Rattew, A. G. & Rebentrost, P. Non-linear transformations of quantum amplitudes: Exponential improvement, generalization, and applications. Preprint at https://doi.org/10.48550/arXiv:2309.09839 (2023).
60. Gonzalez-Conde, J., Watts, T. W., Rodriguez-Grasa, P. & Sanz, M. Efficient quantum amplitude encoding of polynomial functions. *Quantum* **8**, 1297 (2024).
61. Rosenkranz, M. et al. Quantum state preparation for multivariate functions. *Quantum* **9**, 1703 (2025).
62. Bauer, C. W., Deliyannis, P., Freytsis, M. & Nachman, B. Practical considerations for the preparation of multivariate Gaussian states on quantum computers. Preprint at https://doi.org/10.48550/arXiv:2109.10918 (2021).
63. Manabe, H. & Sano, Y. The state preparation of multivariate normal distributions using tree tensor network. *Quantum* **9**, 1755 (2025).
64. Braun, J. & Griebel, M. On a constructive proof of Kolmogorov's superposition theorem. *Constr. Approx.* **30**, 653–675 (2009).
65. Girosi, F. & Poggio, T. Representation properties of networks: Kolmogorov's theorem is irrelevant. *Neural Comput.* **1**, 465–469 (1989).
66. Akashi, S. Application of ϵ-entropy theory to Kolmogorov-Arnold representation theorem. *Rep. Math. Phys.* **48**, 19–26 (2001).
67. Poggio, T., Banburski, A. & Liao, Q. Theoretical issues in deep networks. *Proc. Natl. Acad. Sci. USA* **117**, 30039–30045 (2020).
68. Wang, Y., Siegel, J. W., Liu, Z. & Hou, T. Y. On the expressiveness and spectral bias of KANs. In The *Thirteenth International Conference on Learning Representations*. https://openreview.net/forum?id=ydlDRUuGm9 (2025).
69. Chao, R., Ding, D., Gilyen, A., Huang, C. & Szegedy, M. Finding angles for quantum signal processing with machine precision. Preprint at https://doi.org/10.48550/arXiv:2003.02831 (2020).
70. Dong, Y., Meng, X., Whaley, K. B. & Lin, L. Efficient phase-factor evaluation in quantum signal processing. *Phys. Rev. A* **103**, 042419 (2021).
71. Berry, D. W., Childs, A. M., Cleve, R., Kothari, R. & Somma, R. D. Simulating Hamiltonian dynamics with a truncated Taylor series. *Phys. Rev. Lett.* **114**, 090502 (2015).
72. Low, G. H. & Chuang, I. L. Hamiltonian simulation by uniform spectral amplification. Preprint at https://doi.org/10.48550/arXiv:1707.05391 (2017).
73. Cleve, R., Ekert, A., Macchiavello, C. & Mosca, M. Quantum algorithms revisited. *Proc. R. Soc. A: Math. Phys. Eng. Sci.* **454**, 339–354 (1998).
74. Brassard, G., Høyer, P., Mosca, M. & Tapp, A. Quantum amplitude amplification and estimation. In *Quantum computation and information*, vol. 305 of *Contemporary Mathematics*, 53–74 (American Mathematical Society, Providence, RI, USA, 2002). https://doi.org/10.1090/conm/305/05215.

75. Yoder, T. J., Low, G. H. & Chuang, I. L. Fixed-point quantum search with an optimal number of queries. *Phys. Rev. Lett.* **113**, 210501 (2014).
76. Grover, L. & Rudolph, T. Creating superpositions that correspond to efficiently integrable probability distributions. Preprint at https://doi.org/10.48550/arXiv.quant-ph/0208112 (2002).
77. Kitaev, A. & Webb, W. A. Wavefunction preparation and resampling using a quantum computer. Preprint at https://doi.org/10.48550/arXiv.0801.0342 (2008).
78. Rattew, A. G., Sun, Y., Minssen, P. & Pistoia, M. The efficient preparation of normal distributions in quantum registers. *Quantum* **5**, 609 (2021).
79. Kane, C. F., Gomes, N. & Kreshchuk, M. Nearly optimal state preparation for quantum simulations of lattice gauge theories. *Phys. Rev. A* **110**, 012455 (2024).
80. Kuklinski, P., Rempfer, B., Obenland, K. & Elenewski, J. A simpler Gaussian state-preparation. Preprint at https://doi.org/10.48550/arXiv.2508.03987 (2025).
81. Carrasquilla, J. & Melko, R. G. Machine learning phases of matter. *Nat. Phys.* **13**, 431–434 (2017).
82. Low, G. H. & Chuang, I. L. Hamiltonian simulation by qubitization. *Quantum* **3**, 163 (2019).
83. Chakraborty, S., Gilyén, A. & Jeffery, S. The power of block-encoded matrix powers: Improved regression techniques via faster Hamiltonian simulation. In Baier, C., Chatzigiannakis, I., Flocchini, P. & Leonardi, S. (eds.) *46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)*, vol. 132 of *Leibniz International Proceedings in Informatics (LIPIcs)*, 33:1–33:14 (Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 2019). https://doi.org/10.4230/LIPIcs.ICALP.2019.33.
84. Low, G. H. & Chuang, I. L. Optimal Hamiltonian simulation by quantum signal processing. *Phys. Rev. Lett.* **118**, 010501 (2017).
85. Dalzell, A. M. et al. Quantum algorithms: A survey of applications and end-to-end complexities https://doi.org/10.1017/9781009639651 (2025).
86. Camps, D., Lin, L., Van Beeumen, R. & Yang, C. Explicit quantum circuits for block encodings of certain sparse matrices. *SIAM J. Matrix Anal. Appl.* **45**, 801–827 (2024).
87. Krogh, A. & Hertz, J. A simple weight decay can improve generalization. In Moody, J., Hanson, S. & Lippmann, R. (eds.) *Advances in Neural Information Processing Systems*, vol. 4 (Morgan-Kaufmann, 1991). https://proceedings.neurips.cc/paper_files/paper/1991/hash/8eefcfdf5990e441f0fb6f3fad709e21-Abstract.html.
88. Ng, A. Y. Feature selection, L_1 vs. L_2 regularization, and rotational invariance. In *Proceedings of the Twenty-First International Conference on Machine Learning*, ICML '04, 78 (Association for Computing Machinery, New York, NY, USA, 2004). https://doi.org/10.1145/1015330.1015435.
89. Bartlett, P. L. & Mendelson, S. Rademacher and Gaussian complexities: risk bounds and structural results. *J. Mach. Learn. Res.* **3**, 463–482 (2003).
90. Schuld, M., Bergholm, V., Gogolin, C., Izaac, J. & Killoran, N. Evaluating analytic gradients on quantum hardware. *Phys. Rev. A* **99**, 032331 (2019).
91. Schuld, M., Bocharov, A., Svore, K. M. & Wiebe, N. Circuit-centric quantum classifiers. *Phys. Rev. A* **101**, 032308 (2020).
92. Spall, J. Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. *IEEE Trans. Autom. Control* **37**, 332–341 (1992).
93. Kingma, D. P. & Ba, J. Adam: A method for stochastic optimization. In *International Conference on Learning Representations* https://arxiv.org/abs/1412.6980 (2015).
94. Stokes, J., Izaac, J., Killoran, N. & Carleo, G. Quantum natural gradient. *Quantum* **4**, 269 (2020).
95. Koczor, B. & Benjamin, S. C. Quantum natural gradient generalized to noisy and nonunitary circuits. *Phys. Rev. A* **106**, 062416 (2022).
96. Gacon, J., Zoufal, C., Carleo, G. & Woerner, S. Simultaneous perturbation stochastic approximation of the quantum Fisher information. *Quantum* **5**, 567 (2021).
97. Guo, N., Mitarai, K. & Fujii, K. Nonlinear transformation of complex amplitudes via quantum singular value transformation. *Phys. Rev. Res.* **6**, 043227 (2024).
98. Lin, L. Lecture notes on quantum algorithms for scientific computation. Preprint at https://doi.org/10.48550/arXiv:2201.08309 (2022).

Acknowledgements

We thank Juan Carrasquilla, Naixu Guo, and Xiufan Li for discussions and feedback. P.-W.H. is partially supported by the Engineering and Physical Sciences Research Council (EPSRC) Doctoral Training Partnership (DTP) under grant EP/W524311/1, with a CASE Conversion Studentship in collaboration with Quantum Motion. This work is supported by the National Research Foundation, Singapore, and A*STAR under its CQT Bridging Grant and its Quantum Engineering Programme under grant NRF2021-QEP2-02-P05. Portions of this manuscript were drafted or edited with the assistance of ChatGPT to improve clarity and style.

Author contributions

All authors contributed to the development of the QKAN construction and to writing the final manuscript. P.I. and P.-W.H. worked out the proofs of theorems. L.P. and P.R. conceived the project. P.R. supervised the project.

Competing interests

The authors declare no competing interests.

Additional information

Supplementary information The online version contains supplementary material available at https://doi.org/10.1038/s41534-026-01202-5.

Correspondence and requests for materials should be addressed to Petr Ivashkov, Lirandë. Pira or Patrick Rebentrost.