npj | quantum information

Article

Published in partnership with The University of New South Wales

https://doi.org/10.1038/s41534-025-01099-6

Training-efficient density quantum machine learning

Brian Coyle [1,2], Snehal Raj [1,2,3], Natansh Mathur [1,2,4] ✉, El Amine Cherrat [1,2], Nishant Jain [1,2,5], Skander Kazdaghli [1,2] & Iordanis Kerenidis [1,2,4]

Quantum machine learning (QML) requires powerful, flexible and efficiently trainable models to be successful in solving challenging problems. We introduce density quantum neural networks, a model family that prepares mixtures of trainable unitaries, with a distributional constraint over coefficients. This framework balances expressivity and efficient trainability, especially on quantum hardware. For expressivity, the Hastings-Campbell Mixing lemma converts benefits from linear combination of unitaries into density models with similar performance guarantees but shallower circuits. For trainability, commuting-generator circuits enable density model construction with efficiently extractable gradients. The framework connects to various facets of QML including post-variational and measurement-based learning. In classical settings, density models naturally integrate the mixture of experts formalism and offer natural overfitting mitigation. The framework is versatile—we uplift several quantum models into density versions to improve model performance, or trainability, or both. These include Hamming weight-preserving and equivariant models, among others. Extensive numerical experiments validate our findings.

Modern deep learning owes much of its success to the existence to three factors. The first is increasing resource availability, meaning data and compute power. Secondly, families of powerful and expressive models, such as multilayer perceptrons[1] convolutional and recurrent neural networks[2] and multi-head attention mechanisms[3,4]. With these model primitives, one may build composites such as transformer, diffusion and mixer models which ultimately lead to specific state of the art models such as AlphaZero[5], GPT-3[6] or Dalle[7]. Particularly for these vast models containing billions of parameters, and the enormous quantity of data required, an efficient training algorithm is essential. A cornerstone of many such protocols is the *backpropagation* algorithm[8], and its variants, which enables the computation of gradients throughout the entirety of the network, with minimal overhead beyond the network evaluation itself.

It is reasonable to assume a similar trajectory for quantum machine learning. While rapid progress in quantum error correction[9–11] is increasing the number and quality of effective qubits (compute power), quantum processing units will still likely remain significantly depth-limited for the foreseeable future. On the model side, *quantum* neural networks (QNNs) are typically (but not exclusively) constructed from parametrised quantum circuits (PQCs)[12–15]. However, in many cases these lack task-specific features and unfortunately do not *generally* possess an efficient scaling for training in line with classical backpropagation[16]. Therefore, it is essentially to develop quantum models which can be associated to specific interpretations (for example the convolutional operation on images), and which are efficient to train. The popular *parameter-shift* rule for QNNs[17–22], extracts analytic gradients (i.e., not relying on approximate finite differences), but even the simplest instance of the rule, requires $\mathcal{O}(N)$ gradient circuits to be evaluated for N parameters. To put this in perspective, it was estimated in ref. 16 that, if one is allowed only a single day of computation, the parameter-shift rule can only compute gradients on $n \sim 100$ qubit trainable circuits with *only* ~9000 parameters, assuming reasonable quantum clock speeds. This also does not account for various other problem specific scalings, such as the data which needs to be iterated over for *each* training iteration, or other obstacles such as barren plateaus[23]. Scaling current quantum training approaches towards the size of billion or trillion-parameter deep neural networks, which have been so successful in the modern era, clearly will not be feasible with such methods. Additionally, since we are arguably in the boundary between the NISQ and ISQ eras, models should use circuits which are as compact, yet expressive as possible. On the other hand, they should also be complex enough to avoid classical simulation, surrogation or dequantisation[24–26] but not so complex to admit barren plateaus[27]. Clearly, satisfying all of these constraints is challenging task.

To partially address some of these challenges, in this work we introduce a framework of models dubbed *density* QNNs. Our primary aim is to showcase how these models add another dimension to the landscape of quantum learning models, giving practitioners a new toolkit to experiment

[1]QC Ware, Palo Alto, CA, USA. [2]QC Ware, Paris, France. [3]LIP6, CNRS, Sorbonne Université, Paris, France. [4]IRIF, CNRS—University of Paris, Paris, France. [5]Indian Institute of Technology, Roorkee, India. ✉e-mail: nashmathur@gmail.com

with when tackling the above questions. Through the text, we demonstrate how one may construct density QNNs may be constructed which are more trainable, or more expressive than their pure-state counterparts. Our results are laid out as follows. First, we introduce the general form of the density framework, before discussing comparisons and relationships to other QML model families/frameworks in the literature. Then, we propose two methods of preparing such models on a quantum computer. Next, we prove our primary theoretical results—firstly relating to the gradient query complexity of such models, and secondly discussing the connection to non-unitary quantum machine learning via the Mixing lemma from randomised compiling. We then discuss two proposed connections between density networks and mechanisms in the classical machine learning literature. First, there has been suggestion in the literature that the density networks as we propose them may be a quantum-native analogue of the *dropout* mechanism. We propose separate training and inference phases for density QNNs to bring this comparison closer to reality, but find it still lacking as a valid comparison. Secondly, we demonstrate a strong realisation of density networks within the *mixture of experts* (MoE) framework from classical machine learning—density QNNs can be viewed as a 'quantum mixture of experts'. Finally, we provide numerical results to demonstrate the flexibility of the model to improve performance, or improve trainability (or both). We test several QNN architectures on synthetic translation-invariant data, and Hamming weight preserving architectures on the MNIST image classification task. Finally, we show numerically how, in some capacity, density QNNs may prevent data overfitting using data reuploading as an example, despite not functioning as a true dropout mechanism.

Results

Density quantum neural networks

To begin, we explicitly define the framework (see Supplementary Material A for a discussion) of *density* quantum neural networks (density QNNs) as follows:

$$\rho(\boldsymbol{\theta},\boldsymbol{\alpha},\boldsymbol{x}) := \sum_{k=1}^{K} \alpha_k U_k(\boldsymbol{\theta}_k)\rho(\boldsymbol{x})U_k^\dagger(\boldsymbol{\theta}_k) \tag{1}$$

$\rho(\boldsymbol{x})$ is a data encoded initial state, which is usually assumed to be prepared via a 'data-loader' unitary, $\rho(\boldsymbol{x}) = |\boldsymbol{x}\rangle\langle\boldsymbol{x}|, |\boldsymbol{x}\rangle := V(\boldsymbol{x})|0\rangle^{\otimes n}$, a collection of sub-unitaries $\{U_k\}_{k=1}^K$, and a distribution, $\{\alpha_k\}_{k=1}^K$, which may depend on $\boldsymbol{x}$.

For now, we treat the density state above as an abstraction and later in the text we will discuss methods to prepare the state practically and actually use the model. The preparation method will have relevance for the different applications and connections to other paradigms. Once we have chosen a state preparation method for equation (1), we must choose particular specifications for the sub-unitaries. In some cases, we may recast efficiently trainable models/frameworks within the density formalism to increase their expressibility. In others, we use the framework to improve the overall inference speed of models. In this work, we assume that the sub-unitary circuit structures, once chosen, are fixed, and the only trainability arises from the parameters, $\{\boldsymbol{\theta}_k\}_{k=1}^K$ therein, as well as the coefficients, $\{\alpha_k\}_{k=1}^K$. In other words, we do not incorporate variable structure circuits learned for example via quantum architecture search.

As a generalisation, one may consider adding a data dependence into the sub-unitary *coefficients*, $\boldsymbol{\alpha} \to \boldsymbol{\alpha}(\boldsymbol{x})$, while retaining the distributional requirement for all $\boldsymbol{x}$, $\sum_k \alpha(\boldsymbol{x})_k = 1$. This gives us the more general family of density QNN states:

$$\rho_{\mathsf{D}}(\boldsymbol{\theta},\boldsymbol{\alpha},\boldsymbol{x}) = \sum_{k=1}^{K} \alpha_k(\boldsymbol{x})U_k(\boldsymbol{\theta}_k)\rho(\boldsymbol{x})U_k^\dagger(\boldsymbol{\theta}_k) \tag{2}$$

In the QML world, overly dense or expressive single unitary models are known to have problems related to trainability via barren plateaus[28]. Density QNNs and related frameworks may be a useful direction to retain highly parameterised models but via a combination of smaller, trainable models. We will demonstrate this through several examples in the remainder of the text. Before doing so, in the next section, we want to appropriately cast density QNNs within the current spectrum of quantum machine learning models.

Connection to other QML frameworks

Before proceeding, we first discuss the connection to other popular QML frameworks. For supervised learning purposes, each term in the density state, $U_k(\boldsymbol{\theta}_k)\rho(\boldsymbol{x})U_k^\dagger(\boldsymbol{\theta}_k)$ is expressive enough by itself to capture most basic models in the literature. This is due to the common model definition as $f(\boldsymbol{\theta},\boldsymbol{x}) := \mathrm{Tr}(\mathcal{O}U(\boldsymbol{\theta})\rho(\boldsymbol{x})U^\dagger(\boldsymbol{\theta})) = \mathrm{Tr}(\mathcal{O}(\boldsymbol{\theta})\rho(\boldsymbol{x}))$ for some observable, $\mathcal{O}$, i.e. the overlap between a parameterised Hermitian observable and a data-dependent state. This unifies many paradigms in quantum machine learning literature such as kernel methods[29] and data reuploading models via gate teleportation[30]. Due to the linearity of the quantum mechanics, we can write it also in this form by inserting equation (2) into the function evaluation:

$$\begin{aligned} f_{\mathsf{D}}(\{\boldsymbol{\theta},\boldsymbol{\alpha}\},\boldsymbol{x}) &= \mathrm{Tr}\big(\mathcal{O}(\boldsymbol{\theta},\boldsymbol{\alpha},\boldsymbol{x})\rho(\boldsymbol{x})\big), \\ \mathcal{O}(\boldsymbol{\theta},\boldsymbol{\alpha},\boldsymbol{x}) &:= \sum_{k=1}^{K} \alpha_k(\boldsymbol{x})U_k^\dagger(\boldsymbol{\theta}_k)\mathcal{O}U_k(\boldsymbol{\theta}_k) \end{aligned} \tag{3}$$

Removing this data-dependence from the coefficients simply removes the data-dependence from the observable, $\mathcal{O}(\boldsymbol{\theta},\boldsymbol{\alpha},\boldsymbol{x}) \to \mathcal{O}(\boldsymbol{\theta},\boldsymbol{\alpha})$. Finally, selecting the sub-unitaries to be identical and equal to the data loading unitary, with K equal to the size of the training data leads to observable $\mathcal{O}(\{\boldsymbol{x}_k\}_{k=1}^M,\boldsymbol{\alpha}) := \sum_{k=1}^M \alpha_k U^\dagger(\boldsymbol{x}_k)\mathcal{O}U(\boldsymbol{x}_k) = \sum_{k=1}^M \alpha_k\rho(\boldsymbol{x}_k)$. This is an optimal family of models in a kernel method via the representer theorem[29].

Next, returning to equation (3) and replacing the data dependence from the *state* with a parameter dependence, $\rho(\boldsymbol{x}) \to \rho(\boldsymbol{\theta})$, we fall within the family of *flipped* quantum models[31]. These are a useful model family where the role of data and parameters in the model have been flipped. This insight enables the incorporation of classical shadows[32] for, e.g. quantum training and classical deployment of QML models.

Finally, we have the framework of *post-variational* quantum models[33], originating from the classical combinations of quantum states ansatz[34]. In motivation, these models are perhaps more similar to the '*implicit*'[30] models such as quantum kernel methods, where the quantum computer is used only for specific *fixed*, non-trainable, operations (e.g. evaluating inner products for kernels), rather than '*explicit*'[30] models where trainable parameters reside within unitaries, $U(\boldsymbol{\theta})$. Post-variational models involve optimising coefficients $\alpha_{k,q}$, which are injected into the model via a linear or non-linear combination of observables, $\{\mathcal{O}_q\}_{q=1}^Q$, applied to (non-trainable) unitary transformed states, $\{U_k\rho(\boldsymbol{x})U_k^\dagger\}_{k=1}^K$. In the linear case, the output of the model is:

$$\begin{aligned} f_{\mathsf{PV}}(\boldsymbol{\alpha},\boldsymbol{x}) &= \sum_{kq} \alpha_{kq}\mathrm{Tr}(\mathcal{O}_q U_k\rho(\boldsymbol{x})U_k^\dagger) \\ &= \sum_{kq} \alpha_{kq}\mathrm{Tr}(\mathcal{O}_{kq}\rho(\boldsymbol{x})), \mathcal{O}_{kq} := U_k\mathcal{O}_q U_k^\dagger \end{aligned} \tag{4}$$

The major benefit of post-variational models is that, similar to quantum kernel methods, the optimisation over a convex combination of parameters *outside* the circuit is in principle significantly easier than the non-convex optimisation of parameters *within* the unitaries. However, just like kernel methods, this comes at the limitation of an expensive forward pass through the model, which requires $\mathcal{O}(KQ)$ circuits to be evaluated. In the worst case, this should also be exponential in the number of qubits in to enable arbitrary quantum transformations on $\rho(\boldsymbol{x})$, $KQ \le 4^n$ [33]. To avoid evaluating an exponential number of quantum circuits, it is clearly necessary to employ heuristic strategies or impose symmetries to choose a sufficiently large yet expressive pool of operators $\mathcal{O}_{kq}$. In light of this, ref. 33 proposes ansatz

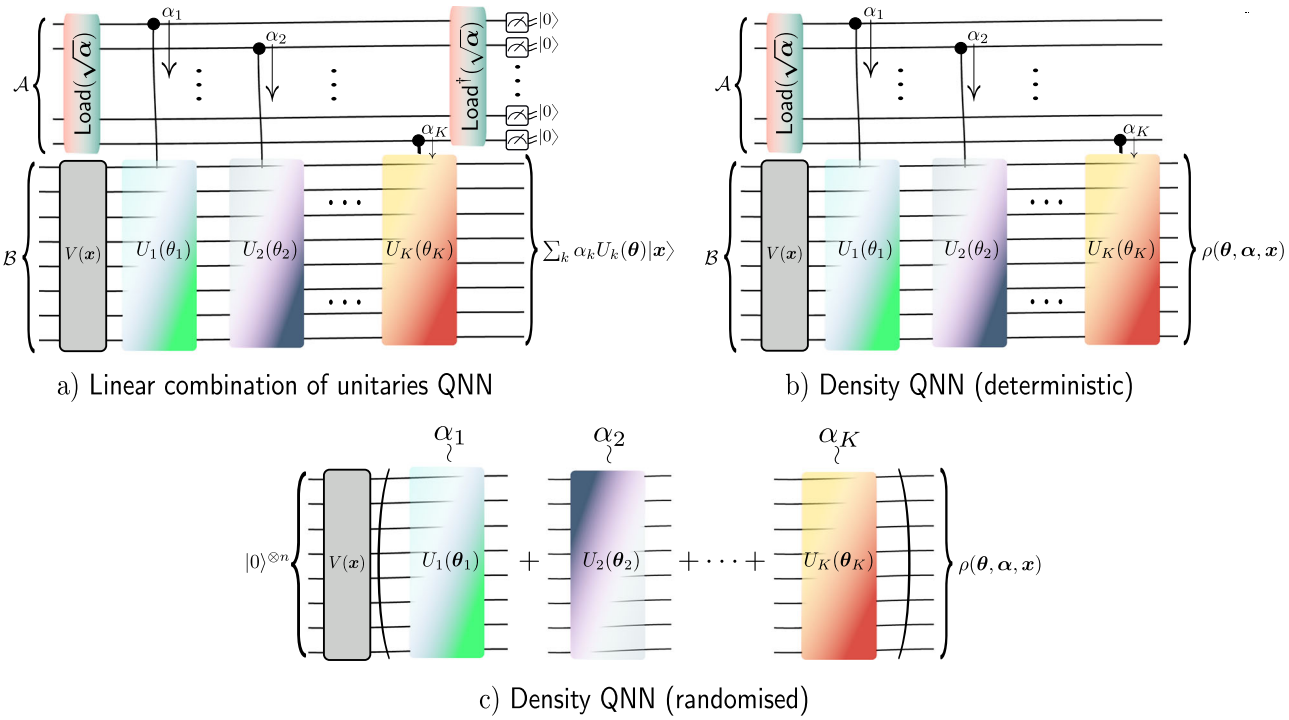

Fig. 1 | Density quantum neural networks. a Linear combination of unitaries quantum neural networks (LCU QNNs) preparing the state $\sum_k \alpha_k U_k(\boldsymbol{\theta}_k)|\boldsymbol{x}\rangle$ via postselection on an ancilla register $\mathcal{A}$ which prepares the distribution $\boldsymbol{\alpha}$. **b** shows corresponding density quantum neural network, implemented deterministically to prepare the state $\rho(\boldsymbol{\theta}, \boldsymbol{\alpha}, \boldsymbol{x})$. Finally, the instantiation of the density QNN state via randomisation is shown in (**c**) where sub-unitary, $U_k(\boldsymbol{\theta}_k)$ is only prepared with the probability α_k without the need for the multi controlled deep circuits and ancilla qubits. The deterministic density QNN, (**b**) is required if one wishes to make a true comparison of these networks to the dropout mechanism. From the Mixing lemma, the randomised version, (**c**), can distil the performance benefits of the more powerful LCU QNN, (**a**), into very short depth circuits. The probability loaders, $\mathsf{Load}(\sqrt{\boldsymbol{\alpha}})$ are assumed to be unary data loaders which act on K qubits within the register, $\mathcal{A}$ and have depth $\log(K)$[71]. One could also use binary `Prepare` and `Select` circuits acting on $\log(K)$ qubits as is more standard in LCU literature. The resulting functions from each network, $f(\boldsymbol{\theta}, \boldsymbol{\alpha}, \boldsymbol{x})$ result from the measurement of an observable, $\mathcal{O}$, via $f(\boldsymbol{\theta}, \boldsymbol{\alpha}, \boldsymbol{x}) = \mathrm{Tr}(\mathcal{O}\sigma(\boldsymbol{\theta}, \boldsymbol{\alpha}, \boldsymbol{x}))$, where σ is the output state from each circuit. $V(\boldsymbol{x})$ is the n-qubit data loader acting on register, $\mathcal{B}$.

expansion strategies[34] or gradient heuristics to grow the pool of quantum operations. Such techniques may be also incorporated into our proposal, but we leave such investigations to future work.

Preparing density quantum neural networks

As mentioned above, we have not yet described a method to prepare the density QNN state, equation (1). Figure 1 showcases two methods of doing so. For now, we do not assume any specific choice for the sub-unitaries. There are two methods to prepare the density state. The first is via a *deterministic* circuit which exactly prepares $\rho(\boldsymbol{\theta}, \boldsymbol{\alpha}, \boldsymbol{x})$, and shown in Fig. 1b. We prove the correctness of this circuit in Supplementary Material I.2. The structure of the circuit can be related directly to the corresponding *linear combination of unitaries* QNN[35] which prepares instead the pure state, $\sum_k \alpha_k U_k(\boldsymbol{\theta}_k)|\boldsymbol{x}\rangle$, seen in Fig. 1a. Notably, the deterministic density QNN removes the need for ancilla postselection on a specific state, ($(|0\rangle_{\mathcal{A}}^{\otimes n})$ in the figure). In other words, while a single forward pass through an LCU QNN will only succeed with some probability p, the deterministic density QNN state preparation succeeds with probability $p = 1$. While the circuits in Fig. 1a, b are conceptually simple, the controlled operation of the sub-unitaries may be very expensive in practice, which is a necessity without any further assumptions. In Supplementary Material I.2 we discuss certain assumptions on the structure of the sub-unitaries which may simplify the resource requirements of this preparation mechanism, specifically assuming a Hamming-weight preserving structure allows the removal of the generic controlled operation.

The second method uses the *distributional* property of $\boldsymbol{\alpha}$ to only prepare the density state $\rho(\boldsymbol{\theta}, \boldsymbol{\alpha}, \boldsymbol{x})$ *on average*, depicted in Fig. 1c. In this form, the forward pass completely removes the need for ancillary qubits, and complicated controlled unitaries.

The effect of this is threefold:

- A forward pass through the randomised density QNN (Fig. 1c) requires time which is upper bounded by the execution speed of *only* the most complex unitary, U_{k^*}. This is illustrated in Fig. 1c. In the language of post-variational models measuring Q observables on a randomised density QNN has complexity $\mathcal{O}(Q)$, a K-fold improvement.
- Secondly, we will show that the gain in efficiency in moving from the LCU to randomised density QNN does not come at a significant loss in model performance. We prove this, under certain assumptions, using the Hastings-Campbell Mixing lemma from randomised quantum circuit compiling.
- Thirdly, and related to the first two points, one can view the randomised density QNN as an *explicit* version of the post-variational (in the sense of ref. 30) framework. This may be an interesting direction to study given the series of hierarchies found by ref. 30 between implicit, explicit and reuploading models.

Gradient extraction for density QNNs

For density QNNs to be performant in practice, they must be efficiently trainable. In other words, it should not be exponentially more difficult to evaluate gradients from such models, compared to the component sub-unitaries. In the following, we describe general statements regarding the gradient extractability from density QNNs. By then choosing the sub-unitaries to themselves be efficiently trainable (in line with a so-called

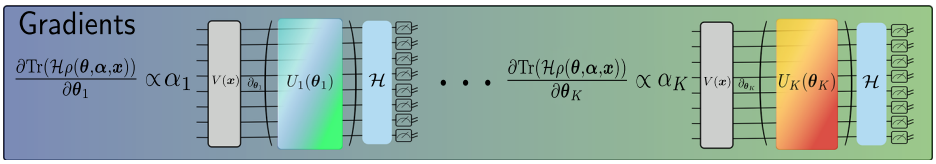

Fig. 2 | Illustration of Corollary 1. In the case where no parameters are shared across the sub-unitaries, the gradients of the density model in equation (1) when measured with an observable $\mathcal{H}$ simply involves computing gradients for each sub-unitary individually. As a result, the full model introduces an $\mathcal{O}(K)$ overhead for gradient extraction. If $K = \mathcal{O}(\log(N))$ and each sub-unitary admits a backpropagation scaling for gradient extraction, the density model will also admit a backpropagation scaling.

backpropagation scaling, which we will define), the entire model will also be. We formalise this as follows:

Proposition 1. (Gradient scaling for density quantum neural networks) Given a density QNN as in equation (1) composed of K sub-unitaries, $\mathcal{U} = \{U_k(\boldsymbol{\theta}_k)\}_{k=1}^K$, implemented with distribution, $\boldsymbol{\alpha} = \{\alpha_k\}$, an unbiased estimator of the gradients of a loss function, $\mathcal{L}$, defined by a Hermitian observable, $\mathcal{H}$:

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\alpha}, \boldsymbol{x}) = \mathrm{Tr}\big(\mathcal{H}\rho(\boldsymbol{\theta}, \boldsymbol{\alpha}, \boldsymbol{x})\big) \tag{5}$$

can be computed by classically post-processing $\sum_{l=1}^K \sum_{k=1}^K T_{\ell k}$ circuits, where $T_{\ell k}$ is the number of circuits required to compute the gradient of sub-unitary k, $U(\boldsymbol{\theta}_k)$ with respect to the parameters in sub-unitary ℓ, $\boldsymbol{\theta}_\ell$. Furthermore, these parameters may be shared across the unitaries, $\boldsymbol{\theta}_k = \boldsymbol{\theta}_{k'}$ for some k, k'.

The proof is given in Supplementary Material B.1, but it follows simply from the linearity of the model. Now, there are two sub-cases one can consider. First, if all parameters between sub-unitaries are independent, $\boldsymbol{\theta}_k \neq \boldsymbol{\theta}_\ell, \forall\, k, \ell$. This gives the following corollary, also in Supplementary Material B.1 and illustrated in Fig. 2.

Corollary 1. Given a density QNN as in equation (1) composed of K sub-unitaries, $\mathcal{U} = \{U_k(\boldsymbol{\theta}_k)\}_{k=1}^K$ where the parameters of sub-unitaries are independent, $\boldsymbol{\theta}_k \neq \boldsymbol{\theta}_\ell, \forall\, k, \ell$ an unbiased estimator of the gradients of a loss function, $\mathcal{L}$, equation (5) can be computed by classically post-processing $\sum_{k=1}^K T_k$ circuits, where T_k is the number of circuits required to compute the gradient of sub-unitary k, $U(\boldsymbol{\theta}_k)$ with respect to the parameters, $\boldsymbol{\theta}_k$.

The second case is where some (or all) parameters are shared across the sub-unitaries. Taking the extreme example, $\theta_l^j = \theta_k^j =: \theta^j\, \forall k, l$—i.e. all sub-unitaries from equation (1) have the same number of parameters, which are all identical. In this case, for each sub-unitary, l, we must evaluate all K terms in the sum so at most the number of circuits will increase by a factor of K^2—we need to compute every term in the matrix of partial derivatives.

Note that this is the number of *circuits* required, not the overall sample complexity of the estimate. For example, take the single layer commuting-block circuit (just a commuting-generator circuit) with C mutually commuting generators. Also assume a suitable measurement observable, $\mathcal{H}$, such that the resulting gradient observables, $\{\mathcal{O}_c | \mathcal{O}_c := [G_c, \mathcal{H}]\}_{k=1}^C$, can be simultaneously diagonalized. To estimate these C gradient observables each to a precision ε (meaning outputting an estimate $\tilde{o}_c$ such that $|\tilde{o}_k - \langle\psi|\mathcal{O}_c|\psi\rangle| \leq \varepsilon$ with confidence $1-\delta$) requires $\mathcal{O}\big(\varepsilon^{-2}\log\big(\frac{C}{\delta}\big)\big)$ copies of ψ (or equivalently calls to a unitary preparing ψ). It is also possible to incorporate strategies such as shadow tomography[32], amplitude estimation[36] or quantum gradient algorithms[37] to improve the C, δ or ε parameter scalings for more general scenarios, though inevitably at the cost of scaling in the others.

Efficiently trainable density networks. The results of the previous section state that moving to the density framework does not result in an exponential increase in gradient extraction difficulty, unless the number of sub-unitaries is exponential. However, what we really care for is that the models are end-to-end *efficiently* trainable, meaning that overall their gradients can be computed with a *backpropagation* scaling. This is the resource scaling which the (classical) backpropagation algorithm obeys, and which we ideally would strive for in quantum models. In the following, we can specialise the derived results to the cases where the component sub-unitaries have efficient gradient extraction protocols. This will render the entire density model also efficiently trainable in this regime.

This 'backpropagation' scaling can be defined as follows. Specifically:

Definition 1. (Backpropagation scaling[16,38]) Given a parameterised function, $f(\boldsymbol{\theta})$, $\boldsymbol{\theta} \in \mathbb{R}^N$, with $f'(\boldsymbol{\theta})$ being an estimate of the gradient of f with respect to $\boldsymbol{\theta}$ up to some accuracy ε. The total computational cost to estimate $f'(\boldsymbol{\theta})$ with backpropagation is bounded with:

$$\mathcal{T}(f'(\boldsymbol{\theta})) \leq c_t \mathcal{T}(f(\boldsymbol{\theta})) \tag{6}$$

and

$$\mathcal{M}(f'(\boldsymbol{\theta})) \leq c_m \mathcal{M}(f(\boldsymbol{\theta})) \tag{7}$$

where $c_t, c_m = \mathcal{O}(\log(N))$ and $\mathcal{T}(g)/\mathcal{M}(g)$ is the time/amount of memory required to compute g.

In plain terms, a model which achieves a backpropagation scaling according to Definition 1, particularly for quantum models, implies that it does not take significantly more effort, (in terms of number of qubits, circuit size, or number of circuits) to compute gradients of the model with respect to all parameters, than it does to evaluate the model itself.

One family of circuits which does obey such a scaling are the so-called *commuting-block* QNNs, defined in ref. 38, and which contain B blocks of unitaries generated by operators which all mutually commute within a block. We discuss the specific circuits in 'Methods' but for now we specialise Proposition 1 to these commuting-block unitaries as follows:

Corollary 2. (Gradient scaling for density commuting-block quantum neural networks) Given a density QNN containing k sub-unitaries, each acting on n qubits. Each sub-unitary, k, has a commuting-block structure with B_k blocks. Assume each sub-unitary has different parameters, $\boldsymbol{\theta}_k \neq \boldsymbol{\theta}_\ell, \forall\, k, \ell$. Then an unbiased estimate of the gradient can be estimated by classical post-processing $\mathcal{O}(2\sum_k B_k - K)$ circuits on $n+1$ qubits.

Proof. This follows immediately from Proposition 1 and Theorem 5 from ref. 38. Here, the gradients of a single B-block commuting block circuit can be computed by post-processing $2B-1$ circuits, where 2 circuits are required per block, with the exception of the final block, which can be treated as a commuting generator circuit and evaluated with a single circuit.

At this stage, we showcase two possibilities when constructing density networks. It should be noted that these in some sense represent extreme cases, and should not be taken as the exclusive possibilities. Ultimately, the successful models will likely exist in the middle group. The first path allows us to increase the *trainability* of certain QML models in the literature. In Table 1, we show some results if we were to do so for some popular

Table 1 | Summary of gradient scalings for training density quantum neural networks

QNN ansatz	N_{params} Original	N_{grad} Original	N_{params} Density	N_{grad} Density
D layer hardware efficient[39]	$\mathcal{O}(nD)$	$\mathcal{O}(nD)$	$\mathcal{O}(nD)$	$\mathcal{O}(D)$
Equivariant XX[38]	$\mathcal{O}(G)$	$\mathcal{O}(1)$	$\mathcal{O}(KG)$	$\mathcal{O}(K)$
HW pres.[65]—pyramid	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$	$\mathcal{O}(1)$
HW pres.[55]—butterfly	$\mathcal{O}(n\log(n))$	$\mathcal{O}(n\log(n))$	$\mathcal{O}(n\log(n))$	$\mathcal{O}(\log(n))$
HW pres.[67]—round-robin	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$

Number of gradient circuits (N_{grad}) required to estimate full gradient vector for original quantum neural networks versus their density QNN counterparts each with N_{params} parameters acting on n qubits. The equivariant XX ansatz[38], an example of a commuting-generator circuit contains G commuting unitaries (G depends on the maximum locality chosen), K versions of which can be combined to give a density version. For the HW preserving, we refer to the versions created exclusively from commuting-block unitaries, not those which take K versions of the original circuit—see Fig. 6 for the distinction with the round-robin circuit. We suppress precision factors of $\mathcal{O}(\varepsilon^{-2})$ and $\mathcal{O}(\log(\delta^{-1}))$, and we assume a direct sampling method to evaluate gradients.

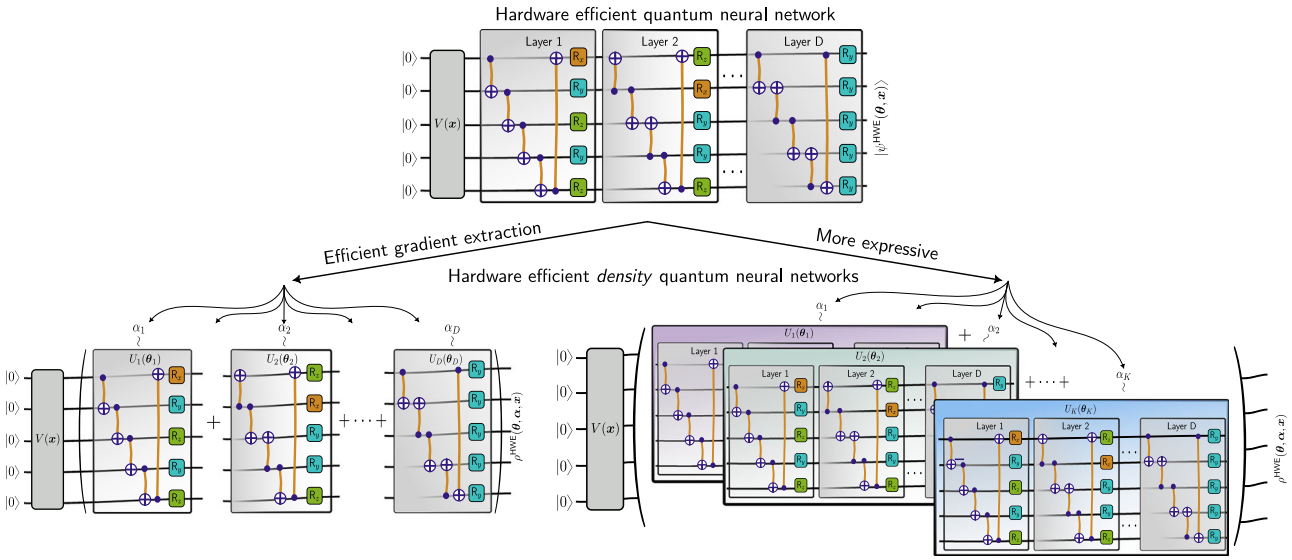

Fig. 3 | Decomposing a hardware efficient ansatz for a density QNN. D layers of a hardware efficient (HWE) ansatz with entanglement generated by CNOT ladders and trainable parameters in single qubits R_x, R_y, R_z gates. (bottom left) D layers extracted into D sub-unitaries with probabilities, $\{\alpha_d\}_{d=1}^{D}$ for a density QNN version. Applying the commuting-generator framework to the density version, $\rho^{\text{HWE}}(\boldsymbol{\theta}, \boldsymbol{\alpha}, \boldsymbol{x})$, enables parallel gradient evaluation in $2D$ circuits versus $2nD$ as required by the pure state version, $|\psi^{\text{HWE}}(\boldsymbol{\theta}, \boldsymbol{\alpha}, \boldsymbol{x})\rangle$. TO illustrate potential differences between sub-unitaries, we arbitrarily reverse CNOT directions in subsequent layers and partially accounting for low circuit depth. (bottom right) Alternatively, we can simply create a more expressive version of the hardware efficient QNN within the density framework by duplicating across K sub-unitaries with probabilities $\{\alpha_k\}_{k=1}^{K}$ retaining D layers each. In this case, the model requires $2nDK$ circuits for gradient extraction, but each sub-unitary can have independent parameters learning different features, especially if each contains different entanglement structures.

examples. The first step is to dissect commuting-block components from each 'layer' of the respective model, then treat these components as sub-unitaries within the density formalism and then apply Corollary 2. In the following sections, we describe this strategy for the models in the table, beginning with the hardware efficient ansatz.

Secondly, we may simply use it as a means to increase overall model expressibility—where the component sub-unitaries, $U_k(\boldsymbol{\theta}_k)$ are any generic trainable circuits (which are independent for simplicity). Further assume each $U_k(\boldsymbol{\theta}_k)$ has identical structure with N parameters acting on n qubits and requiring $T_{n,N}$ gradient circuits each. Then, from Corollary 1, a density model will require $KT_{n,N}$ parameter-shift circuits. In many cases, K will be a constant independent of N, or n, and furthermore this evaluation over the K sub-unitaries can be done in parallel.

In the next section, and in Fig. 3 we illustrate these paths using hardware efficient QNNs. For the examples in the following sections (the other models referenced in Table 1 and others), we demonstrate both of these directions.

Hardware efficient quantum neural networks. To illustrate the two possible paths for model construction, we use a toy example (shown in Fig. 3)—the common but much maligned *hardware efficient*[39] quantum neural network. These 'problem-independent' ansätze were proposed to keep quantum learning models as close as possible to the restrictions of physical quantum computers, by enforcing specific qubit connectivities and avoiding injecting trainable parameters into complex transformations. These circuits are extremely flexible, but this comes at the cost of being vulnerable to barren plateaus[23] and generally difficult to train.

A D layer hardware efficient ansatz on n qubits is usually defined to have 1 parameter per qubit (located in a single qubit Pauli rotation) per layer. The parameter-shift rule with such a model would require $2nD$ individual circuits to estimate the full gradient vector, each for M measurements shots. Given such a circuit, we can construct a density version with D sub-unitaries and reduce the gradient requirements from $2nD$ to $2D$ as the gradients for the single qubit unitaries in each sub-unitary can be evaluated in parallel, using the commuting-generator toolkit from Methods and Corollary 2. This example is relatively trivial as the resulting unitaries are shallow depth (which also likely increases the ease of classical simulability) and training each corresponds only to learning a restricted single qubit measurement basis. In the Fig. 3, we take a variation of the common CNOT-ladder layout—entanglement is generated in each layer by nearest-

neighbour CNOT gates. Typically, an identical structure is used in each layer, however in the figure we allow each sub-unitary extracted from each layer to have a varying CNOT control-target directionality and different single qubit rotations in each layer. This is to increase differences between each 'expert' (see below) as each sub-unitary can generate different levels of (dis)entanglement. Secondly, as illustrated in Fig. 3 one can also define a density version which is *not* more trainable than the original version—in this case, we have K depth-D hardware efficient circuits, which according to the parameter shift rule would now require $\mathcal{O}(2KDn)$ circuits. However, the density model contains more parameters (K-fold more) than the original single circuit version, and possibly is more expressive as a QML model.

LCU and the mixing lemma

The second feature of the density framework is the relationship to linear combination of unitaries (LCU) quantum machine learning. Above, we discussed two methods of preparing the density state, equation (1) and illustrated in Fig. 1. Now, we will demonstrate how one may translate performance guarantees from families of LCU QNNs (Fig. 1a) to the randomised version of the density QNN (Fig. 1c).

Specifically, we will show that, in at least one restricted learning scenario, we will show that if one can construct and train an LCU QNN (Fig. 1a) which has a better learning performance (in terms of e.g. classification accuracy) than any component unitary, this improved performance can be transferred to a density QNN without performance loss. This transference has an important consequence—due to the minimal requirements of implementing a randomised density QNN (Fig. 1c) on quantum hardware, relative to the LCU QNN, we can implement the more performant model much more cheaply. To do so, we will prove a result using the Hastings-Campbell mixing lemma[40,41] from the field compiling of complex unitaries onto sequences of simpler quantum operations.

In this context, we will adapt the Mixing lemma as follows. Assume one trains K sub-unitaries $\{U_k(\boldsymbol{\theta}_k)\}$ each to be 'good' models, in that they each achieve a low prediction error, δ_1, to some ground truth function. Next, with the trained sub-unitaries fixed, one learns a linear combination, $\sum_k \alpha_k U_k$, with (distributional) coefficients, $\{\alpha_k\}$, by training *only* the coefficients. Assume this more powerful QNN model (LCU QNN) achieves a 'better' prediction error, $\delta_2 < \delta_1$. However, despite better performance, the LCU QNN is far more expensive to implement than any individual U_k (as can be seen in Fig. 1a). The logic of the Mixing lemma implies that instead of this deep circuit, we may randomise over the unitaries—create a randomised density QNN—and achieve the same error as the LCU QNN but with the same overhead as the most complex U_k. We formalise this as the following:

Lemma 1. (Mixing lemma for supervised learning) Let $h(\boldsymbol{x})$ be a target ground truth function, prepared via the application of a fixed unitary, $V, h(\boldsymbol{x}) := \mathrm{Tr}(\mathcal{O}V\rho(\boldsymbol{x})V^\dagger)$ on a data encoded state, $\rho(\boldsymbol{x})$ and measured with a fixed observable, $\mathcal{O}$. Suppose there exists K unitaries $\{U_k(\boldsymbol{\theta})\}_{k=1}^K$ such that these each are δ_1 good predictive models of $h(\boldsymbol{x})$:

$$\mathbb{E}_{\boldsymbol{x}}|h(\boldsymbol{x}) - f_k(\boldsymbol{\theta}, \boldsymbol{x})| \leq \delta_1, \forall k \tag{8}$$

and a distribution $\{\alpha_k\}_{k=1}^K$ such that predictions according to the LCU model $f_{\mathrm{LCU}}(\boldsymbol{\theta}, \boldsymbol{\alpha}, \boldsymbol{x}) := \mathrm{Tr}(\mathcal{O}\left(\sum_k \alpha_k U_k(\boldsymbol{\theta})\right)\rho(\boldsymbol{x})\left(\sum_k \alpha_k U_k^\dagger(\boldsymbol{\theta})\right))$ have error bounded as:

$$\mathbb{E}_{\boldsymbol{x}}|h(\boldsymbol{x}) - f_{\mathrm{LCU}}(\boldsymbol{\theta}, \boldsymbol{\alpha}, \boldsymbol{x})| \leq \delta_2 \tag{9}$$

for some δ_1, $\delta_2 > 0$. Then, the corresponding density QNN, $f(\boldsymbol{\theta}, \boldsymbol{\alpha}, \boldsymbol{x}) = \mathrm{Tr}(\mathcal{O}\rho(\boldsymbol{\theta}, \boldsymbol{\alpha}, \boldsymbol{x}))$, $\rho(\boldsymbol{\theta}, \boldsymbol{\alpha}, \boldsymbol{x}) = \sum_{k=1}^K \alpha_k U_k^\dagger(\boldsymbol{\theta})\rho(\boldsymbol{x})U_k(\boldsymbol{\theta})$ can generate predictions for $h(\boldsymbol{x})$ with error:

$$\mathbb{E}_{\boldsymbol{x}}|h(\boldsymbol{x}) - f(\boldsymbol{\theta}, \boldsymbol{\alpha}, \boldsymbol{x})| \leq \frac{\delta_1^2}{4\,\|\,\mathcal{O}\|_\infty} + 2\delta_2 \tag{10}$$

We prove this lemma in Supplementary Material B.2. In the above, $\mathbb{E}_{\boldsymbol{x}}|h(\boldsymbol{x}) - g_{\boldsymbol{\theta}}(\boldsymbol{x})|$ is the expected prediction error admitted by the model, $g_{\boldsymbol{\theta}}(\boldsymbol{x})$, where the expectation is taken over the distribution the data is drawn from. Take the common predictor observable to be $\mathcal{O} = Z, \|\,\mathcal{O}\|_\infty = 1$ and set $\delta_1 = \delta$. Assume we find a distribution for the LCU QNN which quadratically reduces this error $\delta_2 = \delta^2$. Then according to Lemma 1, the density QNN will also be an $\mathcal{O}(\delta^2)$ good predictor, but at the same implementation cost as a single sub-unitary QNN.

In the rest of this work, we do not primarily take the QNN → LCU QNN → density QNN route as implied by Lemma 1, instead directly train the parameters of the density QNN indicating the viability of bypassing the expensive LCU directly. Though, for the sake of completeness, in Supplementary Material C we do provide one example where the LCU QNN outperforms the single-circuit QNN numerically. It may be that non-unitary (pure LCU) quantum machine learning[35] and the density QNNs we present here have different difficulties *in practice* to achieve good model performance. Also, the above result is clearly restrictive to the setting where the model to be learned is itself a QNN, and we also know the correct observable to measure. However, it may be generalised to stronger statements where the data is generated by an arbitrary classical function. We leave both of these investigations to future work.

In the quantum compiling problem, one endeavours to produce a sequence of 'simple' operations which approximate the behaviour of a target unitary or quantum channel on input states. Randomised compiling (informally via the Hastings-Campbell mixing lemma[40,41]) allows one to carry a quadratic suppression in compiling error from a channel composed of a LCU, into the corresponding *randomised* channel which has the same execution overhead as an individual unitary in the linear combination. If we treat the compilation task as a learning problem, as has been done in several works, spawning the subfield of *variational* quantum compiling[42,43], one can intuitively see how the Mixing lemma may be directly applied.

However, to argue for the benefit of density networks over individual QML circuits, we must go further and generalise to other learning tasks, for example supervised data classification/regression (which is the primary task we use density QNNs for in this work).

Connection to classical mechanisms

We have discussed features of density QNNs, and elucidated their position within the spectrum of existing quantum models. In this section, we discuss relationships or analogies between density QNNs to *purely classical* mechanisms and models. We summarise these relationships in the following three observations:

- **Observation 1:** The randomised state preparation method may be used as the training mode, and the deterministic state preparation method may be used as the inference mode for density QNNs, if analogies are to be made with the classical dropout mechanism.
- **Observation 2:** Unlike classical dropout, density QNNs do not combine an exponential number of sub-networks at inference time.
- **Observation 3:** Density QNNs are a quantum analogue of Mixture of Experts (MoE) models, which are subtly different from ensemble methods.

We expand on these observations in the following.

Quantum dropout. In the previous section, we have demonstrated how density QNNs have the capacity to mitigate overfitting via more strategic parameter allocation. In light of this, one may make an analogy with the randomised density QNN and the *dropout*[44,45] mechanism in classical neural networks. Dropout is an effective method of combining the predictions of exponentially many classical neural networks, and also mitigates overfitting. This analogy has indeed been remarked in recent works[46] due to the random 'removal' of $K-1$ sub-unitaries in each forward pass, which, on the surface, appears similar to the randomised removal of neurons in a neural network. it has therefore been conjectured that randomised density QNNs as we have defined them may be less

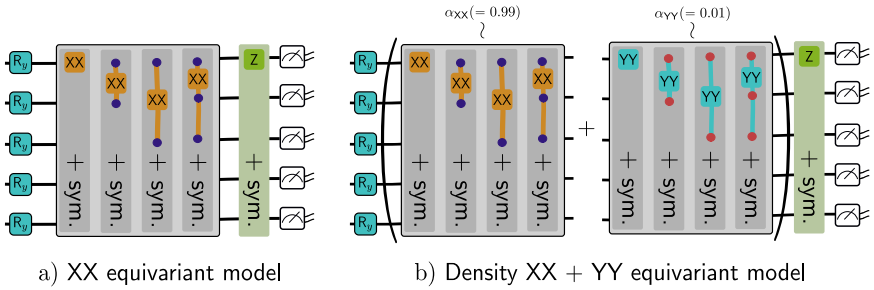

Fig. 4 | Equivariant density QNNs. a *The commuting-generator XX model* and **a**, **b** *XX+YY density QNN model*. The former contains up to three-body Pauli-X generated operations with twirling applied to enforce equivariance. The latter contains two sub-unitaries U_{XX} (circuit (**a**)) and U_{YY} which has the same structure but replacing Pauli-X operations with Pauli-Y. U_{XX}/U_{YY} are applied with probabilities $\alpha_{XX/YY}$. Each sub-unitary in **b** are commuting-generator circuits, so each has efficiently extractable gradients.

prone to overfitting because of this analogy. However, in the following, we will argue that this is an incorrect, or at least an incomplete comparison.

This incompleteness of the comparison arises from (at least) two sources. Firstly, there is a distinct difference (and important) between the *training* and *evaluation* modes in classical dropout. In the training mode, a dropout mechanism applies a random zeroing of the nodes of a neural network, which removes outgoing weights from those nodes. This means, on any forward pass through the loss function, only a sub-network is actually activated. However, in the *inference* mode, all sub-networks are effectively present, where the outgoing weights of a dropped node are re-weighted by the probability of dropping that node.

Regarding different training and inference operations, we propose the randomised implementation of the density QNN (Fig. 1c) as the state preparation method for the training phase of the model. To align with classical dropout, we then propose the *deterministic* density QNN (Fig. 1b) as the inference mode. Due to the linearity of the model, this has the effect of weighting the contributions of each sub-unitary QNN, $U_k(\boldsymbol{\theta}_k)$ by the corresponding coefficients, α_k, exactly as in classical dropout.

However, dropout also has the key feature of efficiently combining an *exponential* number of effective sub-networks at inference, which is a critical feature in boosting model performance. However, in order to maintain training efficiency (within Corollary 2), a density QNN may have at most $K = \mathcal{O}(\log(N))$ 'sub-networks', where N is the total number of circuit parameters. As such, it is unclear if at a fundamental level a density QNN can function fully as a quantum analogue for dropout. Nevertheless, we will demonstrate that the density QNN still can, in some capacity, mitigate overfitting, and perform the effective *action* of dropout.

Density quantum neural networks as a mixture of experts. The next, and final, comparison to classical methods we discuss is an interpretation of the density QNN framework as a MoE model[47,48]. MoE models have achieved success even at the level of large scale machine learning models such as Mixtral[49], particularly where sparsity is required. The MoE framework is flexible and powerful as it allows an increase in the capacity of a model with minimal increase in computational effort. This same methodology drives the density QNN framework of this work. A MoE contains a set of '*experts*', $\{f_1, \ldots, f_K\}$, each of which is 'responsible' for a different training case. A *gating* network, $\mathcal{G}$, decides which expert should be used for a given input. In the simplest form, the MoE output, $F(\boldsymbol{x})$, is a weighted sum of the experts, according to the gating network output $F(\boldsymbol{x}) = \sum_k \mathcal{G}_k(\boldsymbol{x}) f_k(\boldsymbol{x})$ where $\mathcal{G}_k(\boldsymbol{x})$ is the k^{th} output of the gating network, given input $\boldsymbol{x}$. A typical difference between MoE models and ensemble models is that, for the latter every element of the ensemble is evaluated for every input, while for an MoE, only a subset are activated (e.g. the top-k experts for a given input). We elaborate on this in Supplementary Material H.

We can interpret the density QNN exactly as a form of MoE as follows. By allowing the sub-unitary coefficients to be *data-dependent* ($\alpha_k \to \alpha_k(\boldsymbol{x})$), as in equation (3), we can predict them using a gating network in an MoE. Now, the sub-unitaries become the 'experts' and the overall model has increased capacity to learn which sub-unitary (expert) is more relevant for the task at hand (the given input $\boldsymbol{x}$). If the data were quantum states, we could use another quantum neural network as a gating network, or perhaps classical shadows[32] with a classical neural network. We choose a simple yet traditional model for the data-dependent gating mechanism we describe in 'Methods'. There has been extensive investigation into variations of the MoE models both for deep[50,51] and shallow models[52,53] though we leave thorough investigation of different approaches in the quantum world to future work. For example, one could employ techniques regulating the dependence on the overall mixture on any individual subsets of unitaries[50] which can occur when the distribution tends to become quite peaked, meaning the gating network chooses to rely only on a small fraction of experts for a given input.

Numerical results

Now, we demonstrate that density networks can be successful *in practice*, through three examples. The first is an demonstration where an efficiently trainable model can be made more expressive (see Fig. 3, right), and in the second we choose an example of an interpretable model family for which we construct *both* efficiently trainable versions (Fig. 3, left) and more expressive (Fig. 3, right). The third, and final, main example demonstrates the ability of density QNNs to mitigate overfitting compared to standard single unitary QML models. We summarise the takeaways of each set of numerics as follows:

- **Model**: Equivariant quantum neural networks.
  Takeaways: Backpropagation scaling models can be made more expressive and performant. Model initialisations via sub-unitary pre-training is possible and successful.
- **Model**: Hamming-weight preserving (orthogonal) quantum neural networks.
  Takeaways: Non-backpropagation scaling models can be made trainable with minimal or no performance loss. Increasing sub-unitary expressivity monotonically improves overall performance.
- **Model**: Data reuploading quantum neural networks.
  Takeaways: Distributing parameters over shallower sub-unitaries mitigates overfitting to training data.

Equivariant quantum neural networks. For the first example, we build a density version of a commuting-generator (a special case of commuting-block circuit) ansatz on the simplified classification problem posed by ref. 38. Here, the challenge is classifying *bars* vs. *dots*, in a noisy setting, described in 'Methods'.

For the base QNN, we use the original 'XX' ansatz (denoted $U_{XX}(\boldsymbol{\theta})$) of ref. 38, see Fig. 4a) and compare against a density QNN version with two sub-unitaries, $\{U_1, U_2\}$ and coefficients $\{\alpha_1, \alpha_2\}$. We define $U_1 := U_{XX}(\boldsymbol{\theta})$. The second sub-unitary, $U_2 := U_{YY}(\boldsymbol{\theta})$, has the exact same structure as $U_{XX}(\boldsymbol{\theta})$, but with the XX generators replaced by Pauli-Y generators. The measurement operator is chosen to be the translation invariant operator $\sum_i Z_i$ to suit the translation symmetry in the synthetic problem. The output function of the model is then:

$$f(\boldsymbol{\theta}, \boldsymbol{\alpha}, \boldsymbol{x}) := \sum_{i=1}^{d} \sum_{k=1}^{K} \mathrm{Tr}(Z_i \rho(\boldsymbol{\theta}, \boldsymbol{\alpha}, \boldsymbol{x})) \tag{11}$$

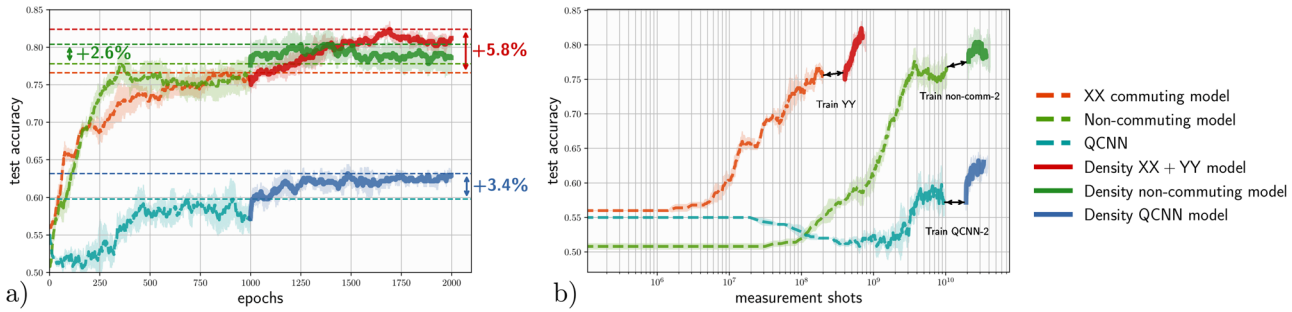

Fig. 5 | Numerics on noisy bars and dots dataset. We create density QNN versions of the non-trivial models from ref. 38; (1) the commuting-generator circuit in Fig. 4, (2) a `non-commuting' equivariant QNN and (3) the quantum convolutional neural network[54], all on 10 qubits. In all cases the density QNN is initialised from (separately) pretrained (for 1000 epochs) base versions, and training continues for another 1000. For the non-commuting and QCNN density models the sub-unitaries have identical structure but trained independently. We show mean and standard deviation in test accuracy vs. **a** *training epochs* and **b** *number of overall shots* over 5 independent training runs, from the same initialisation. In all cases, after base model performance saturation, the density QNN improves the final result. The gaps in (**b**) are to account for the extra measurement overhead to initialise and train the second sub-unitaries. This is U_{YY} for the density model and U_2 (non-comm-2/QCNN-2) for the other two models. In all cases, the density version is initialised with $\alpha_1/\alpha_{XX} = 0.99$ $\alpha_2/\alpha_{YY} = 0.01$ which are also trainable.

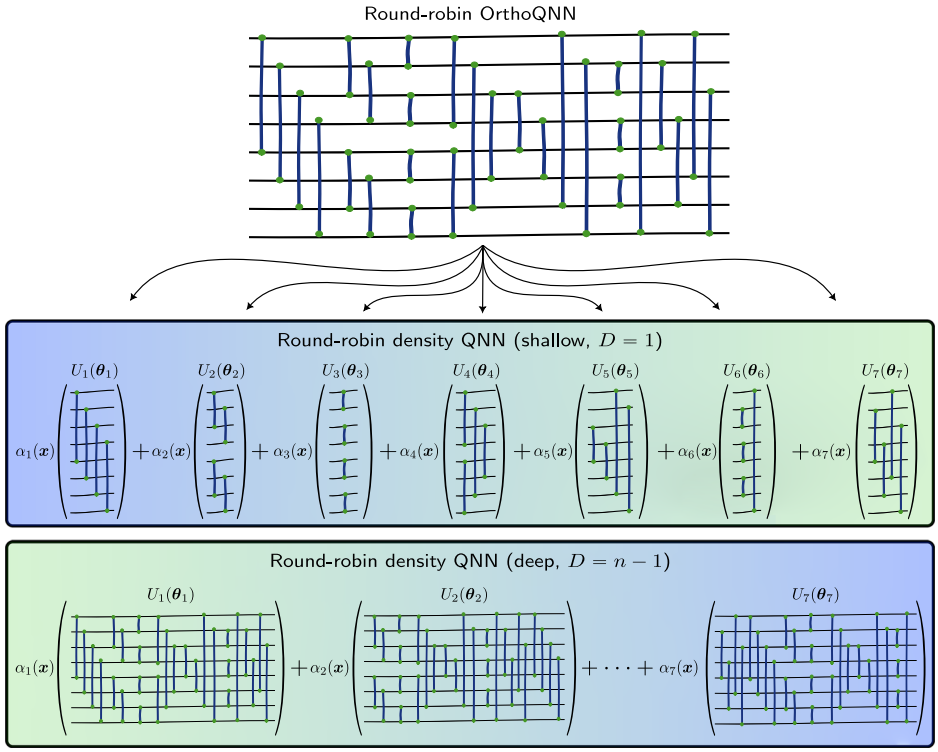

Fig. 6 | Hamming weight preserving density QNN with round-robin structure. Illustration of extraction possibilities from an orthogonal QNN with round-robin connectivity, on 8 qubits. We can extract a minimally expressive `shallow' (depth $D = 1$) or a maximally expressive `deep' (depth $D = \frac{n(n-1)}{2}$) sub-unitary for each 'expert'. Both cases have $K = n - 1$ sub-unitaries, which have $\frac{n}{2}$ (shallow) or $\frac{n(n-1)}{2}$ (deep) parameters. The shallow version has gradients which can be evaluated in parallel using $n - 1$ circuits with gradient complexity stated in Table 1. The deep version assumes a cycled connectivity as seen in the figure with gradient extraction complexity $\mathcal{O}(n^3)$. We can interpolate between the two regimes for different sub-unitary depths, D, between $1 \le D \le n - 1$, but efficient gradient extraction is lost for $D \ge 2$. An MoE gating network can be used to predict the probability coefficients, $\{\alpha_k(\boldsymbol{x})\}$.

with

$$\rho(\boldsymbol{\theta}, \boldsymbol{\alpha}, \boldsymbol{x}) := \alpha_1 U_{XX}(\boldsymbol{\theta}) \left[\bigotimes_{j=1}^{d} |\boldsymbol{x}\rangle\langle\boldsymbol{x}|_j^{ry} \right] U_{XX}^{\dagger}(\boldsymbol{\theta}) + \alpha_2 U_{YY}(\boldsymbol{\theta}) \left[\bigotimes_{j=1}^{d} |\boldsymbol{x}\rangle\langle\boldsymbol{x}|_j^{ry} \right] U_{YY}^{\dagger}(\boldsymbol{\theta}) \tag{12}$$

Each generator that appears in $U_{XX/YY}(\boldsymbol{\theta})$ is of the form $\mathsf{sym}(X_1)$ or $\mathsf{sym}(X_1 \ldots X_k)$ ($\mathsf{sym}(Y_1)$ and $\mathsf{sym}(Y_1 \ldots Y_k)$ respectively) for some $k \le K$. The operation sym is the twirling operation used to generate equivariant quantum circuits (in this case, equivariance with respect to translation symmetry). For example, $\mathsf{sym}(X_1X_2)$ is a sum of all pairs of X operators on the state with no intermediate trivial qubit, $\mathsf{sym}(X_1X_3)$ is a sum of all pairs on the state separated by exactly *one* trivial qubit (in either direction, visualising the qubits as a 1D chain with closed boundary conditions) and so on. However, note that in this case since the YY ansatz ($U_{XX/YY}(\boldsymbol{\theta})$) is in the same basis as the data encoding (R_y), we have classical simulatability for computing expectation values since the initial circuit is effectively Clifford[38]. The density circuits are visualised in Fig. 4b which we adapt from ref. 38, and the results of the experiment can be seen in Fig. 5a, b. We also construct density versions of the two other non trivial models considered in ref. 38, the quantum convolutional neural network (QCNN)[54] and a 'non-commuting` equivariant circuit (see ref. 38, Fig. 5c). In both these latter cases, we again consider two sub-unitaries U_1 and U_2 which have identical structure but different learned parameterisations, $U_1 = U_2$, $\boldsymbol{\theta}_1^* \neq \boldsymbol{\theta}_2^*$. In initialising the density QNN, we bias the model towards the 'first' sub-unitary with probability 99% (as in Fig. 4b). The density QNN outperforms its base counterpart in all cases, and the best tradeoff between efficiency and model

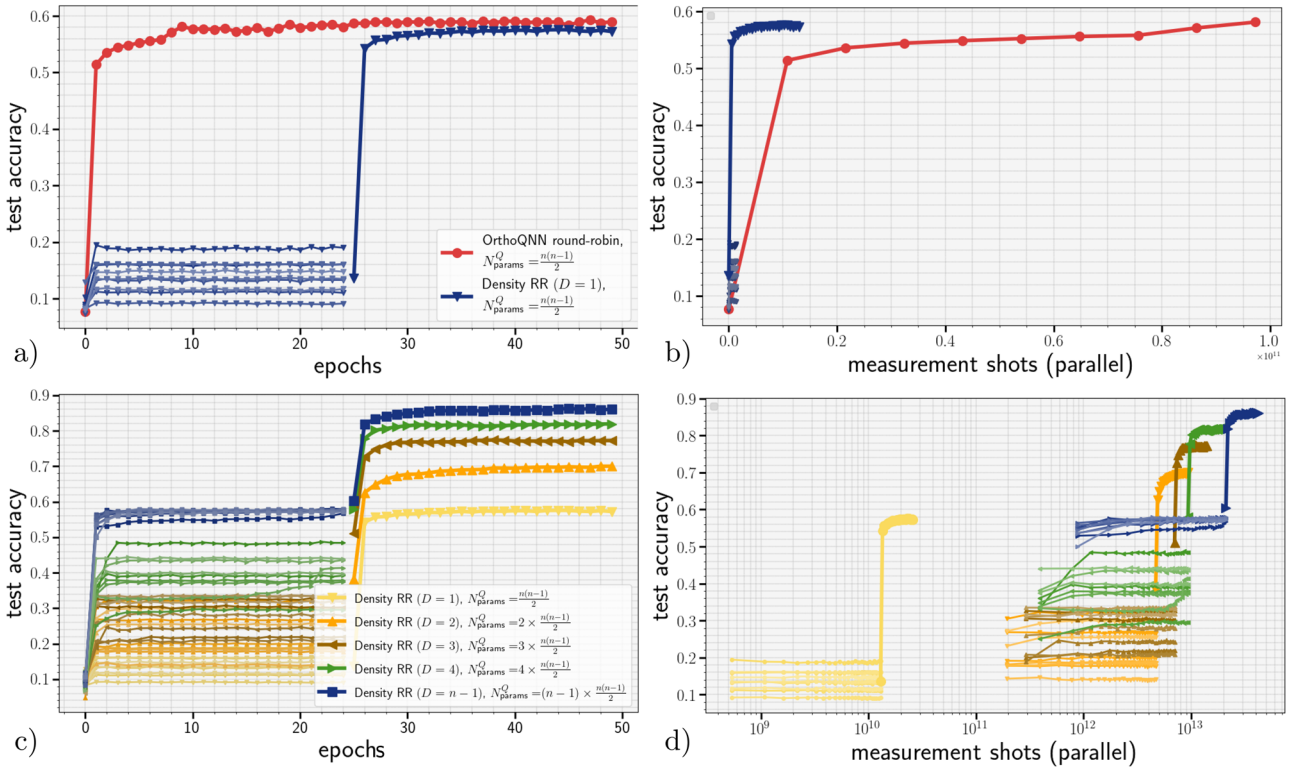

Fig. 7 | Round-robin Hamming weight preserving density QNN results. *Top row* Round-robin OrthoQNN versus shallow ($D = 1$) round-robin density QNN, as a function of training epochs (**a**) and measurement shots (**b**) on 10 qubits. For $D = 1$ the density QNN is a commuting generator circuit and so requires only $n - 1$ circuit evaluations for each iteration whereas the OrthoQNN requires $\frac{4n(n-1)}{2}$ circuits as per the parameter-shift rule for gradient evaluation. *Bottom row* increasing complexity of density QNN by increasing round robin depth, D, from $D \in \{1, 2, 3, 4, (n-1)\}$ as a function of epochs (**c**) and measurement shots (**b**, log scale). In all cases for the density QNN, $n - 1 = 9$ sub-unitaries are trained *in parallel* for 25 epochs (9 thin lines) before initialising the density QNN, and training continues for a further 25. Performance of density QNN increases monotonically with sub-unitary depth, D.

performance is found for the density $XX + YY$ model. We describe the experiment details in 'Methods'.

Orthogonal quantum neural networks. For the second example, we turn to the Hamming weight (HW) preserving quantum neural network ($U(1)$ equivariant), and specifically their orthogonal QNNs variants. As mentioned above, these models have desirable properties from a machine learning point of view—they are interpretable and can stabilise training. While we focus on the easiest to simulate version of HW preserving models here, all of the below is applicable to the more difficult to classically simulate compound QNNs[55] (Supplementary Material E.2) and general Hamming weight preserving unitaries.

The data encoded state for orthogonal QNNs on n qubits, $\rho(\boldsymbol{x}) := |\boldsymbol{x}\rangle\langle\boldsymbol{x}|, |\boldsymbol{x}\rangle := \sum_j x_j |\mathbf{e}_j\rangle$, is a unary amplitude encoding of the vector $\boldsymbol{x}$ ($\mathbf{e}_j$ is a basis vector with a single 1 in position j and zeros otherwise). The unitary that acts on this state consists of two qubit gates known as reconfigurable beam splitter (RBS) gates (see 'Methods'). There are various configurations for these gates which parameterise some subset of $SO(n)$ matrices. To begin, we choose the *round-robin* architecture, shown in Fig. 6 (top), composed of $\mathcal{O}(n)$ layers of where in each layer qubit is connected to only one other, alternating so eventually each qubit is connected to every other. Evaluating gradients for this QNN specification via the parameter shift rule requires $\mathcal{O}(n^2)$ circuits. We construct the 'shallow' version of a round-robin density QNN by simply extracting each layer into the sub-unitaries, with $K = \mathcal{O}(n)$. Then each U_k is a depth $D = 1$ commuting generator circuit whose gradients can be evaluated in parallel with $\mathcal{O}(n)$ circuits via Corollary 2. In the spirit of Fig. 3 (right), we also take $K = \mathcal{O}(n)$ copies of the *full* round robin circuit to construct a more expressive model. This model will require $\mathcal{O}(Kn^2) = \mathcal{O}(n^3)$ circuits for gradient evaluation, but also has $\mathcal{O}(n)$ more parameters than the 'vanilla' round-robin model. We can also interpolate between these two regimes by scaling the complexity of each sub-unitary with a round-robin layer depth between $1 \le D \le n - 1$. In all cases, we demonstrate the MoE formalism by introducing a gating network to predict the coefficients, $\{\alpha(\boldsymbol{x})\}$, details given in 'Methods'.

In all cases, since the unitaries are Hamming weight preserving, the output states, $|\boldsymbol{y}^k\rangle$ from each sub-unitary are of the form $|\boldsymbol{y}\rangle = \sum_j y_j |\mathbf{e}_j\rangle$ for some vector $\boldsymbol{y}$. This output is related to the input vector via some orthogonal matrix transformation O^U, $\boldsymbol{y} = O^U\boldsymbol{x}$ where the elements of O^U can be computed via the angles of the RBS gates in the circuit (see 'Methods'). The typical output of such a layer is the vector $\boldsymbol{y}$ itself, for further processing in a deep learning pipeline. As a result the output of a 'orthogonal' density QNN is a linear combination of orthogonal transformations, $\boldsymbol{y} = \sum_k \alpha_k \boldsymbol{y}_k = \sum_k \alpha_k O^{U_k}\boldsymbol{x}$, benefitting from both stable orthogonal training within each sub-unitary, and the generality afforded by the linear combination. In contrast, adding more (e.g. RBS) gates directly to the pure state version circuit (i.e. the 'vanilla' round-robin QNN) would not increase the expressivity as the output would only correspond to another (different) orthogonal matrix. The results of these experiments are shown in Fig. 7.

Mitigating overfitting with density QNNs. We have argued above that the density QNN framework possesses similarities and differences to the dropout mechanism. Nevertheless, we can demonstrate that indeed, density QNNs have the capacity to do what dropout does—which is to mitigate overfitting. To demonstrate this, we use *data reuploading*[56] QNNs. Data reuploading is a poweful QML technique which allows the construction of universal quantum classifiers, even with single qubits. We compare a single, deep ('vanilla') data reuploading QNN to a density version with approximately the same number of parameters, and test

which model is more likely to overfit training data, generated by Chebyshev polynomials—see 'Methods' for details. In summary, we find that given a budget of N parameters, distributing these across a density QNN with shallow sub-unitaries will lead to better results (from the perspective of overfitting) than increasing the circuit depth of a single quantum classifier. See 'Methods' for a definition of data reuploading and Supplementary Material G.3 for a discussion of data reuploading with density QNNs.

For both density and vanilla reuploading models, we use arbitrary single qubit rotations with 3 parameters as the trainable operations. For the vanilla version, these are repeated for L 'reuploads' so the model has $3L$ parameters. For the density QNN, we choose $K = 5$ sub-unitaries, the depth of each (D) is chosen such that $5 \times 3D \approx 3L$. Both models are illustrated in Fig. 8a, and we also shown the truncated Fourier series learned by each of the 5 sub-unitaries in the density case. The results can be seen in Fig. 8b which shows the test error, as measured by mean squared error (MSE) and the generalisation gap (difference between train and test errors).

We see as the vanilla reuploading model becomes more expressive (with increasing L), it overfits and test error grows. In contrast, the density model (with approximately the same number of parameters) does not overfit and the test error remains small. This means that if, without any prior, one would choose an overparameterised model (see[57,58] for further discussions on overparameterisation which may be incorporated in future work), the density model would perform better. In Supplementary Material J we provide further detail on this, and supplementary numerics.

Drawbacks and limitations

To conclude our results, we discuss some potential drawbacks and limitations with the density framework, specifically those which it does not solve relative to single-unitary variational learning models. A major problem with the latter is the existence of barren plateaus, or regions of problematic gradients more generally. As discussed in ref. 38 the general relationship between commuting generator circuits, and barren plateaus is still an open question—particularly in finding a circuit architecture which has a suitable relationship between input states, circuit generators and measurement observable to render the dynamical Lie algebra polynomial, the circuit non-classically simulatable and gradients non-vanishing. This is not a question resolved via the density framework either, although if such an architecture was to be found, it could similarly be uplifted as other specific models studied in this work. However, an interesting research direction provided by the density framework is to allow a combination of different sub-unitaries, which may have different and independent gradient behaviour. However, in this case it is possible that the sub-unitary with most favourable gradients would dominate the training (a feature known as representation or expert collapse in the mixture of expert literature[59–61]). However, it has also been demonstrated how barren plateau issues can be mitigated or eliminated via clever initialisation strategies.

Besides the barren plateau issue, there are still a number of open questions with variational quantum learning models, which are also potentially inherited by non-unitary (LCU, dissipative or density) learning models including; the effect of quantum or measurement noise, lack of data or problem-dependent circuit architectures and training dynamics related to hyperparameter tuning and loss function choices. All of these are fruitful areas for future study.

Discussion

Efficiently trainable quantum machine learning models, particularly those operable on quantum hardware, are crucial for the field's advancement. Here, we introduce density quantum neural networks (density QNNs), a generalisation of traditional pure parameterised quantum circuit models, incorporating classical randomisation. We demonstrate that the gradient complexity of these models hinges on the sub-unitaries' gradient evaluation. By employing commuting-block circuits for these sub-unitaries, the model requires a constant number of gradient circuits while potentially enhancing expressivity. Additionally, investigating the density formalism's impact on the efficient classical simulability of the model in specific scenarios is essential. We also highlight the connections between density QNN formalism and other primitives in quantum and classical machine learning, such as quantum-native dropout, the MoE formalism, and measurement-based and post-variational quantum machine learning. Exploring the intersections among these distinct learning mechanisms presents promising research opportunities. Most significantly, we provide a direct connection between LCU QML and density QNNs via the Hastings-Campbell Mixing lemma from randomised quantum compiling. This indicates that performance benefits from the LCU can be directly translated into the density framework, for a dramatically reduced cost.

Inspired by interpretable models such as orthogonal QNNs, and efficiently trainable circuits such as commuting generator QNNs, we developed density versions of these families for numerical experimentation. The density versions demonstrate increased learning performance, or more efficient trainability, or possibly both, over their pure state counterparts. Future research should explore the expressivity of density QNNs with

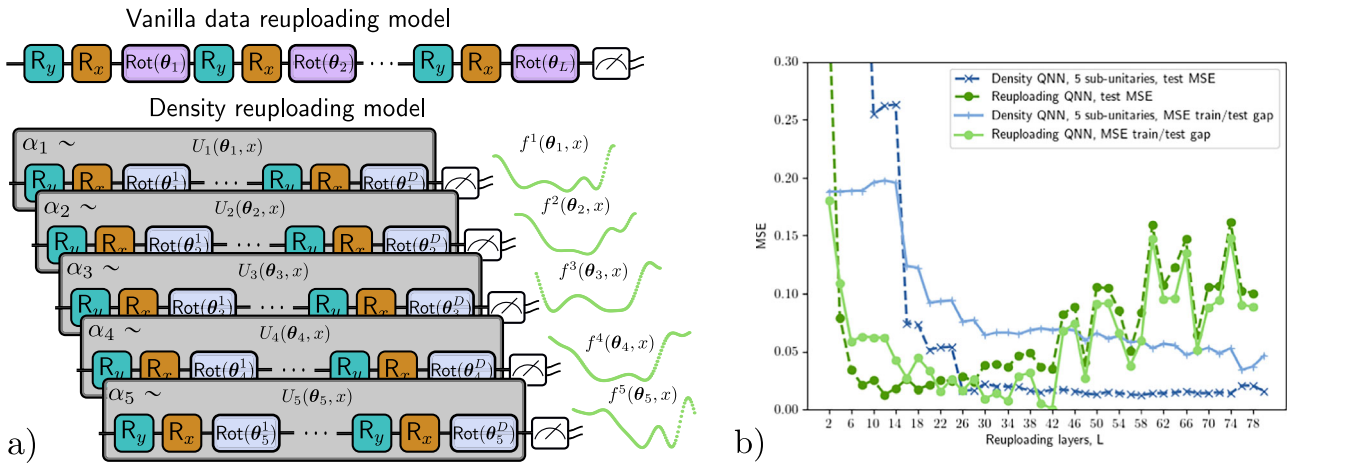

Fig. 8 | Overfitting mitigation with density QNNs—data reuploading. a A 'vanilla' single qubit data reuploading QNN with L reuploading layers versus a density QNN with 5 sub-unitaries, each with D data reuploads. Also shown is the partial Fourier series representations of each sub-unitary, $\{f^k(\boldsymbol{\theta}_k, x)\}_{k=1}^5$ when learning the Chebyshev polynomial $T_2(x)$. **b** Generalisation gap between train and test mean square errors (MSE) in a regression problem for learning the Chebyshev polynomial $T_2(x)$. The number of parameters between both models is kept approximately the same for each L, so that $5 \times 3D \approx 3L$. For a fixed number of parameters, the test error and train/test gap (generalisation error) for the vanilla data reuploading model diverges, while the density QNN test error remains small.

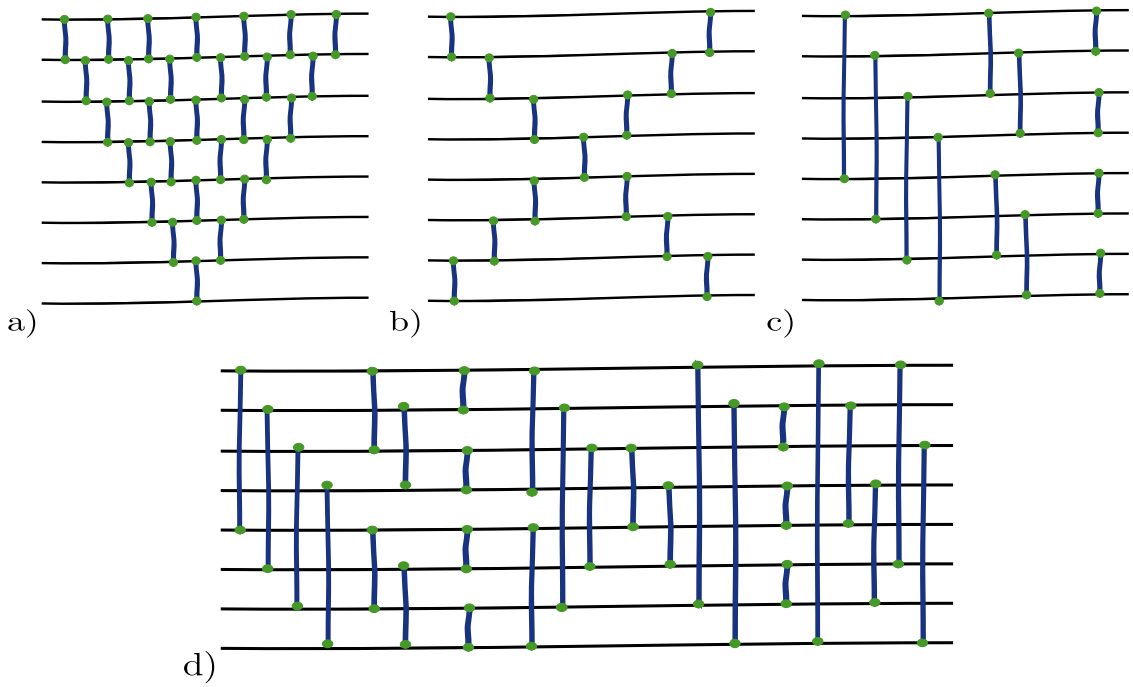

Fig. 9 | Ansätze for orthogonal quantum neural networks[55,65]. **a** Pyramid circuit, **b** X circuit, **c** Butterfly circuit and **d** Round-robin circuit. Each gate corresponds to an *RBS* gate with (potentially different) parameter θ. With respect to the number of qubits, n, the depths of each of these layers is $2n-1, n-1, \log(n)$ and $n-1$, respectively.

various sub-unitary families and seek other efficiently trainable sub-unitary types. Successful examples can be promptly applied to the density model, as illustrated with hardware-efficient QNNs.

This work underscores the transformative potential of density QNNs in advancing quantum machine learning, laying a robust foundation for future innovations.

Methods

Commuting-block quantum neural networks

Commuting-block QNNs are circuits decomposed into *Bblocks* or layers, so the ansatz has the following form:

$$\mathcal{U}(\boldsymbol{\theta}) = \prod_{b=1}^{B}\prod_{j=1}^{N_b} U_b(\boldsymbol{\theta}_j^b) = \prod_{b=1}^{B}\prod_{j=1}^{N_b} e^{i\theta_j^b G_j^b} \tag{13}$$

Each block, $b \in [1, \ldots B]$, with $\prod_{j=1}^{N_b} e^{i\theta_j^b G_j^b}$ contains generators, G_j^b such that $[G_i^b, G_j^b] = 0\ \forall i, j$. Increasing the number of blocks was shown to increase the dimension of the dynamical Lie algebra (DLA)[62], which is a concept useful in probing barren plateaus and expressibility in QNNs. Assume $b = 1$ for simplicity. Then, given a measurement operator, $\mathcal{H}$, each mutually commuting generator defines a gradient observable, $\mathcal{O}_k := [G_k, \mathcal{H}]$, such that $i\langle\psi(\boldsymbol{\theta}, \boldsymbol{x})|[G_k, \mathcal{H}]|\psi(\boldsymbol{\theta}, \boldsymbol{x})\rangle = \partial_{\theta_k}\mathcal{L}$. If all generators commute or anticommute with $\mathcal{H}$, i.e. $[G_k, \mathcal{H}] = 0$ or $\{G_k, \mathcal{H}\} = 0$ for all k, one can show that the gradient operators, $\mathcal{O}_k$, all mutually commute[38]. As a result, $\mathcal{O}_k$ are simultaneously diagonalisable and their statistics can be extracted in parallel, using a single circuit query, which is diagonalized by appending a 'diagonalizing' unitary to it. One can also extract higher order gradient information (second derivatives etc.) but with a decreasing precision[38]. If G_i $\forall\ i$ is a tensor product of Pauli operators, implemented in depth $T(n)$, the gradient circuit has depth $T(n) + \mathcal{O}\left(\frac{n}{\log(n)}\right)$ in general, but the overhead can be constant if G_i is supported on a constant number of qubits, admitting a backpropagation scaling.

Hamming weight preserving quantum neural networks

Hamming weight preserving unitaries, or $U(1)$ equivariant circuits[27] are a useful family of QNNs which admit favourable properties such as interpretability. Such circuits can act on states which have a fixed (k) Hamming weight (k is equal to the number of 1's in a computational basis state) input or on superpositions of different Hamming weight states. In the latter case, the unitary can be written in a block diagonal form with the kth block, acting independently on the Hamming weight k subspace[55,63,64]. The dimension of the Hamming weight k subspace is $\binom{n}{k}$. However, the dimension of the DLA (and hence the behaviour of barren plateaus etc.) depends on the type of operation performed within this space[64]. However, for small k even brute force subspace simulation is sufficient for efficient classical simulation. Though we note that DLA or brute force simulation are not the only possible classical simulation methods for these or generic circuits. For example, one could consider Clifford proximity or entanglement content in the resulting states and tailor simulation appropriately. Focusing on the $k = 1$ HW space, we get *orthogonal* quantum neural networks (OrthoQNNs)[55,63,65], which have been useful in mitigating unstable gradient dynamics for classical models and may offer a cubic to quadratic speedup in parameterising and training (classical) orthogonal neural networks[65].

Hamming weight preserving unitaries (OrthoQNNs as a special case)[65–67] can be built from *reconfigurable beam splitter* (RBS) gates, or Givens' rotations, which have the following (two qubit) form:

$$\begin{aligned} RBS(\theta) &:= e^{-i\frac{\theta}{2}G_{RBS}} := e^{-i\frac{\theta}{2}(Y\otimes X - X\otimes Y)} \\ &= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) & 0 \\ 0 & \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \end{aligned} \tag{14}$$

Different forms of OrthoQNNs can be defined, each parameterising a (potentially restricted) orthogonal matrix shown in Fig. 9. These are the explicit 'pyramid', 'X' and 'butterfly' ansätze used above and shown in Fig. 9. Replacing the *RBS* gates in these circuits with a so-called *fermionic* beam splitter (*FBS*) gate gives *compound* QNNs, which we discuss in Supplementary Material E.2. These compound QNNs can be represented

by compound matrices acting on higher Hamming weight ($k > 1$) or superpositions thereof.

Data reuploading

Data reuploading[56] QNNs incorporate unitaries which inject the *same* data into the parameterised state multiple times. This allows the learned functions to be highly non-linear in the data, and can produce universal classifiers. Further, it is well known that such models can be expressed as *partial Fourier series* in the data,[24,68,69]. A single reuploading layer, ℓ, can be written as follows:

$$\mathcal{E}^{\ell}_{\boldsymbol{\theta}^{\ell},\boldsymbol{x}}(\rho) := U(\boldsymbol{\theta}^{\ell})V_{\ell}(\boldsymbol{x})\rho V^{\dagger}_{\ell}(\boldsymbol{x})U^{\dagger}(\boldsymbol{\theta}^{\ell}) \tag{15}$$

where ρ is some data-independent initial state, usually $\rho = |0\rangle\langle 0|^{\otimes n}$. Then, the function output from L such uploads is:

$$\mathcal{E}^{L}_{\boldsymbol{\theta},\boldsymbol{x}}(\rho) := \mathcal{E}^{L}_{\boldsymbol{\theta}^{L},\boldsymbol{x}}\Big(\cdots \mathcal{E}^{2}_{\boldsymbol{\theta}^{2},\boldsymbol{x}}\Big(\mathcal{E}^{1}_{\boldsymbol{\theta}^{1},\boldsymbol{x}}(\rho)\Big)\Big) \tag{16}$$

$$f(\boldsymbol{\theta},\boldsymbol{x}) = \mathrm{Tr}\Big(\mathcal{O}\mathcal{E}^{L}_{\boldsymbol{\theta},\boldsymbol{x}}(\rho)\Big) = \sum_{\boldsymbol{\omega}\in\Omega} c_{\boldsymbol{\omega}}(\boldsymbol{\theta})e^{i\boldsymbol{\omega}^{\top}\boldsymbol{x}} \tag{17}$$

which is a sum of Fourier coefficients with frequencies, $\Omega := \{\boldsymbol{\omega}\}$. Here, it is assumed that the encoding unitaries are *Hamiltonian encodings* of the data, $V_{\ell}(\boldsymbol{x}) = \prod_{d=1}^{D}\exp\left(i\boldsymbol{x}_d H^{\ell}_d\right)$ for some (potentially highly non-local) Hermitian operators, $\{H^{\ell}_d\}_{d,\ell}$.

Now, it is known that the expressivity of such models are determined by the eigenspectrum of the Hamiltonians used in the encoding unitaries. These generate the Fourier frequencies available to the model, Ω, while the trainable parameters in the unitaries, $\boldsymbol{\theta}$, determine the coefficients $c_{\boldsymbol{\omega}}(\boldsymbol{\theta})$ which dictate how these frequencies are combined. It is straightforward to generalise this model into a density counterpart, but we explicitly do so in Supplementary Material G.3 for completeness.

Experiment details

Equivariant density QNN. The bars and dots dataset is as follows. Each datapoint is a d-dimensional vector (bar or dot) with either alternating $+1$ or -1 values (dot) or sequential periods of $+1$ or -1 of length $\lfloor\frac{d}{2}\rfloor$ (bar). Gaussian noise with mean = 0 and variance σ^2 is added to each vector. In ref. 38, the translation invariance of the data enables the application of a commuting-generator equivariant ansatz, where each generator consists of a symmetrised Pauli-X string containing up to K-body terms. The measurement observable is also a symmeterised Pauli Z string with $K = 1$, meaning $\mathcal{H} = \sum_{i=1}^{d} Z_i$. Each bar/dot, $\boldsymbol{x} := [x_1, \ldots, x_d]^{\top} \in \mathbb{R}^d$, is encoded as a Pauli Y rotation per qubit, $x_i \mapsto R_y(\frac{\pi x_i}{2})|0\rangle = \cos(\frac{\pi x_i}{4})|0\rangle + \sin(\frac{\pi x_i}{4})|1\rangle =: |\boldsymbol{x}\rangle_i^{ry}$. This angle encoding uses the same number of qubits as the unary amplitude encoding in the OrthoQNN above.

For the numerics in Fig. 5, we use 10 qubits in simulation. We increase the noise to $\sigma = 1.8$ to raise the problem difficulty. We train the model with train and test data of sizes of 1000, 100 respectively and a batch size of 20. The Adam optimiser with a learning rate of 0.001 is used in all cases. We initialise the weighting parameters, $\{\alpha_1, \alpha_2\} = \{\alpha_{XX}, \alpha_{YY}\} = \{0.99, 0.01\}$ to bias the model towards the (pre-trained) equivariant XX model, and $\boldsymbol{\alpha}$ are also trainable. The density model is compared to its commuting XX counterpart, plus a non-commuting model (a model which does not obey the commuting generator properties) and the QCNN[54] with the same structures as described in ref. 38. The formulae to compute the number of shots required by each model (commuting XX, non-commuting, QCNN) is given in ref. 38. The number of shots for the density QNN $XX + YY$ model being $2N_{XX} + 2N_{YY}$ where $N_{XX/YY}$ is the number of shots to train the XX, YY models separately (over 500 epochs for each model). We adapt the code of[38] to generate the results for the commuting, non-commuting and QCNN models which is built using Pennylane[70].

Orthogonal density QNN. For all the density orthogonal QNN results, we take 60,000 train and 10,000 test 28×28 pixel images from the MNIST dataset. For the round-robin decomposition (Fig. 6) each MNIST image is downscaled to 10 qubits using principal component analysis.

For simplicity in simulation, we compute the action of each sub-unitary on the data state, $|\boldsymbol{x}\rangle$ (which is the data-loaded feature vector using the unitary $V(\boldsymbol{x})$), and classically combine them. This bypasses the need to explicitly reconstruct the density matrix, $\rho(\boldsymbol{\theta}, \boldsymbol{\alpha}, \boldsymbol{x})$, though we stress in a real quantum implementation one would apply the randomisation over sub-unitaries, or create the density state explicitly on the quantum computer as in Fig. 1. There are various choices one could make to create the data state, $|\boldsymbol{x}\rangle$, for all the above results we assume a *parallel* vector loader[55,71] but the choice of loader will depend on the available quantum hardware.

With the addition of the data dependent MoE gating network to predict the distribution of sub-unitary coefficients in Fig. 6, the following parameterisation is chosen. Specifically, 'MoE' refers to coefficients defined as a gating network with the following form $\alpha_k(\boldsymbol{x}) = \mathtt{softmax}_k(\mathtt{Linear}(\boldsymbol{x}))$. We investigate the addition of a non-linearity in (Supplementary Material H.2). Also, we could replace the input to the gating network to be a more general feature vector, $\boldsymbol{x} \to \sigma(\boldsymbol{x})$, i.e. the output of another neural network.

For all numerical results involving orthogonal quantum neural networks, we used QCLearn, a custom in-house quantum machine learning software package with model tailored classical simulation, primarily built in JAX[72].

Data reuploading density QNN. For the numerical results demonstrating overfitting mitigation in Fig. 8 with data reuploading, we generate training and test data from the family of Chebyshev polynomials of the first kind, $\{T_n(x)\}$, and compare each models ability to learn the underlying function, defined as:

$$\begin{aligned} T_0(x) &= 1, T_1(x) = x, \\ T_{n+1}(x) &= 2xT_n(x) - T_{n-1}(x), \end{aligned} \tag{18}$$

We choose the ground truth function to be $h(x) := T_4(x)$ and add Gaussian noise with mean 0 and variance 0.25 to the training data to nudge both models towards visible overfitting. The arbitrary single qubit operation we use for reupload ℓ in both reuploading models (density/vanilla) is $U_{\ell}(\boldsymbol{\theta}_{\ell}) = R_z(\theta^1_{\ell})R_y(\theta^2_{\ell})R_z(\theta^3_{\ell})$. We also again use Pennylane[70] for these results.

Data availability

The data corresponding to this manuscript is available upon reasonable request.

Code availability

The code corresponding to this manuscript is available upon reasonable request.

Received: 18 July 2025; Accepted: 7 August 2025;
Published online: 07 November 2025

References

1. Rosenblatt, F. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychol. Rev.* **65**, 386–408 (1958).
2. LeCun, Y., Bengio, Y. & Hinton, G. Deep learning. *Nature* **521**, 436–444 (2015).
3. Bahdanau, D., Cho, K. & Bengio, Y. Neural Machine Translation by Jointly Learning to Align and Translate. *In Proc. 3rd International Conference on Learning Representations (ICLR)* 2015 (eds Yoshua Bengio and Yann LeCun) (San Diego, CA, USA 2015).
4. Vaswani, A. et al. Attention is All you Need. In *Advances in Neural Information Processing Systems* Vol. 30 (eds Guyon, I. et al.) (Curran Associates, Inc., 2017).

5. Silver, D. et al. Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm https://doi.org/10.48550/arXiv.1712.01815 (2017).
6. Brown, T. B. et al. Language models are few-shot learners. In *Proc. 34th International Conference on Neural Information Processing Systems*, NIPS '20, 1877–1901 (Curran Associates Inc., 2020).
7. Ramesh, A. et al. Zero-shot text-to-image generation. In *Proc. 38th International Conference on Machine Learning* 8821–8831 (PMLR, 2021).
8. Rumelhart, D. E., Hinton, G. E. & Williams, R. J. Learning representations by back-propagating errors. *Nature* **323**, 533–536 (1986).
9. Sivak, V. V. et al. Real-time quantum error correction beyond break-even. *Nature* **616**, 50–55 (2023).
10. Acharya, R. et al. Quantum error correction below the surface code threshold. *Nature* **638**, 920–926 (2025).
11. Silva, M. P. d. et al. Demonstration of logical qubits and repeated error correction with better-than-physical error rates. https://doi.org/10.48550/arXiv.2404.02280 (2024).
12. Benedetti, M., Lloyd, E., Sack, S. & Fiorentini, M. Parameterized quantum circuits as machine learning models. *Quantum Sci. Technol.* **4**, 043001 (2019).
13. Bharti, K. et al. Noisy intermediate-scale quantum algorithms. *Rev. Mod. Phys.* **94**, 015004 (2022).
14. Cerezo, M. et al. Variational quantum algorithms. *Nat. Rev. Phys.* **3**, 625–644 (2021).
15. Cerezo, M., Verdon, G., Huang, H.-Y., Cincio, L. & Coles, P. J. Challenges and opportunities in quantum machine learning. *Nat. Comput Sci.* **2**, 567–576 (2022).
16. Abbas, A. et al. On quantum backpropagation, information reuse, and cheating measurement collapse. In *Thirty-seventh Conference on Neural Information Processing Systems*. https://openreview.net/forum?id=HF6bnhfSqH (2023).
17. Mitarai, K., Negoro, M., Kitagawa, M. & Fujii, K. Quantum circuit learning. *Phys. Rev. A* **98**, 032309 (2018).
18. Crooks, G. E. Gradients of parameterized quantum gates using the parameter-shift rule and gate decomposition. https://doi.org/10.48550/arXiv.1905.13311 (2019).
19. Vidal, J. G. & Theis, D. O. Calculus on parameterized quantum circuits. https://doi.org/10.48550/arXiv.1812.06323 (2018).
20. Schuld, M., Bergholm, V., Gogolin, C., Izaac, J. & Killoran, N. Evaluating analytic gradients on quantum hardware. *Phys. Rev. A* **99**, 032331 (2019).
21. Sweke, R. et al. Stochastic gradient descent for hybrid quantum-classical optimization. *Quantum* **4**, 314 (2020).
22. Kyriienko, O. & Elfving, V. E. Generalized quantum circuit differentiation rules. *Phys. Rev. A* **104**, 052417 (2021).
23. McClean, J. R., Boixo, S., Smelyanskiy, V. N., Babbush, R. & Neven, H. Barren plateaus in quantum neural network training landscapes. *Nat. Commun.* **9**, 4812 (2018).
24. Landman, J., Thabet, S., Dalyac, C., Mhiri, H. & Kashefi, E. Classically approximating variational quantum machine learning with random fourier features. In *The Eleventh International Conference on Learning Representations.* https://openreview.net/forum?id=ymFhZxw70uz (2023).
25. Rudolph, M. S., Fontana, E., Holmes, Z. & Cincio, L. Classical surrogate simulation of quantum systems with LOWESA. https://doi.org/10.48550/arXiv.2308.09109 (2023).
26. Bermejo, P. et al. Quantum Convolutional Neural Networks are (Effectively) Classically Simulable. https://doi.org/10.48550/arXiv.2408.12739 (2024).
27. Cerezo, M. et al. Does provable absence of barren plateaus imply classical simulability?. *Nat. Commun.* **16**, 7907 (2025).
28. Holmes, Z., Sharma, K., Cerezo, M. & Coles, P. J. Connecting ansatz expressibility to gradient magnitudes and barren plateaus. *PRX Quantum* **3**, 010313 (2022).
29. Schuld, M. Supervised quantum machine learning models are kernel methods. https://doi.org/10.48550/arXiv.2101.11020 (2021).
30. Jerbi, S. et al. Quantum machine learning beyond kernel methods. *Nat. Commun.* **14**, 517 (2023).
31. Jerbi, S., Gyurik, C., Marshall, S. C., Molteni, R. & Dunjko, V. Shadows of quantum machine learning. *Nat. Commun.* **15**, 5676 (2024).
32. Huang, H.-Y., Kueng, R. & Preskill, J. Predicting many properties of a quantum system from very few measurements. *Nat. Phys.* **16**, 1050–1057 (2020).
33. Huang, P.-W. & Rebentrost, P. Post-variational quantum neural networks. https://doi.org/10.48550/arXiv.2307.10560 (2023).
34. Huang, H.-Y., Bharti, K. & Rebentrost, P. Near-term quantum algorithms for linear systems of equations with regression loss functions. *N. J. Phys.* **23**, 113021 (2021).
35. Heredge, J., West, M., Hollenberg, L. & Sevior, M. Nonunitary quantum machine learning. *Phys. Rev. Appl.* **23**, 044046 (2025).
36. Brassard, G., Hoyer, P., Mosca, M. & Tapp, A. Quantum Amplitude Amplification and Estimation. https://doi.org/10.48550/arXiv.quant-ph/0005055 (2002).
37. Huggins, W. J. et al. Nearly optimal quantum algorithm for estimating multiple expectation values. *Phys. Rev. Lett.* **129**, 240501 (2022).
38. Bowles, J., Wierichs, D. & Park, C.-Y. Backpropagation scaling in parameterised quantum circuits. https://doi.org/10.48550/arXiv.2306.14962 (2023).
39. Kandala, A. et al. Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets. *Nature* **549**, 242–246 (2017).
40. Hastings, M. B. Turning gate synthesis errors into incoherent errors. *Quantum Info Comput.* **17**, 488–494 (2017).
41. Campbell, E. Shorter gate sequences for quantum computing by mixing unitaries. *Phys. Rev. A* **95**, 042306 (2017).
42. Khatri, S. et al. Quantum-assisted quantum compiling. *Quantum* **3**, 140 (2019).
43. Sharma, K., Khatri, S., Cerezo, M. & Coles, P. J. Noise resilience of variational quantum compiling. *N. J. Phys.* **22**, 043006 (2020).
44. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.* **15**, 1929–1958 (2014).
45. Baldi, P. & Sadowski, P. J. Understanding dropout. In *Advances in Neural Information Processing Systems* Vol. 26 (eds Burges, C. J., Bottou, L., Welling, M., Ghahramani, Z. & Weinberger, K. Q.) (Curran Associates, Inc., 2013).
46. Nguyen, Q. T. et al. Theory for Equivariant Quantum Neural Networks. *PRX Quantum* **5**, 020328 (2024).
47. Jacobs, R. A., Jordan, M. I., Nowlan, S. J. & Hinton, G. E. Adaptive mixtures of local experts. *Neural Comput.* **3**, 79–87 (1991).
48. Jordan, M. & Jacobs, R. Hierarchical mixtures of experts and the EM algorithm. In *Proc. 1993 International Conference on Neural Networks (IJCNN-93-Nagoya, Japan)* Vol. 2, 1339–1344 (IEEE, 1993).
49. Jiang, A. Q. et al. Mixtral of Experts.https://doi.org/10.48550/arXiv.2401.04088 (2024).
50. Eigen, D., Ranzato, M. & Sutskever, I. Learning Factored Representations in a Deep Mixture of Experts. https://doi.org/10.48550/arXiv.1312.4314 (2014).
51. Shazeer, N. et al. Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer. In *International Conference on Learning Representations*. https://openreview.net/forum?id=B1ckMDqlg (2017).
52. Cao, L. Support vector machines experts for time series forecasting. *Neurocomputing* **51**, 321–339 (2003).

53. Lima, C. A. M., Coelho, A. L. V. & Von Zuben, F. J. Hybridizing mixtures of experts with support vector machines: investigation into nonlinear dynamic systems identification. *Inf. Sci.* **177**, 2049–2074 (2007).
54. Cong, I., Choi, S. & Lukin, M. D. Quantum convolutional neural networks. *Nat. Phys.* **15**, 1273–1278 (2019).
55. Cherrat, E. A. et al. Quantum Vision Transformers. *Quantum* **8**, 1265 (2024).
56. Pérez-Salinas, A., Cervera-Lierta, A., Gil-Fuster, E. & Latorre, J. I. Data re-uploading for a universal quantum classifier. *Quantum* **4**, 226 (2020).
57. Larocca, M., Ju, N., García-Martín, D., Coles, P. J. & Cerezo, M. Theory of overparametrization in quantum neural networks. *Nat. Comput Sci.* **3**, 542–551 (2023).
58. Peters, E. & Schuld, M. Generalization despite overfitting in quantum machine learning models. *Quantum* **7**, 1210 (2023).
59. Lee-Thorp, J. & Ainslie, J. Sparse mixers: combining MoE and mixing to build a more efficient BERT. In *Findings of the Association for Computational Linguistics: EMNLP 2022* (eds Goldberg, Y., Kozareva, Z. & Zhang, Y.) 58–75 (Association for Computational Linguistics, 2022).
60. Riquelme, C. et al. Scaling Vision with Sparse Mixture of Experts. In *Advances in Neural Information Processing Systems* Vol. 34 (eds Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P. S. & Vaughan, J. W.) 8583–8595 (Curran Associates, Inc., 2021).
61. Chi, Z. et al. On the Representation Collapse of Sparse Mixture of Experts. In *Advances in Neural Information Processing Systems* (eds Oh, A. H., Agarwal, A., Belgrave, D. & Cho, K.) https://openreview.net/forum?id=mWaYC6CZf5 (2022).
62. Larocca, M. et al. Diagnosing barren plateaus with tools from quantum optimal control. *Quantum* **6**, 824 (2022).
63. Cherrat, E. A. et al. Quantum deep hedging. *Quantum* **7**, 1191 (2023).
64. Monbroussou, L. et al. Trainability and Expressivity of Hamming-Weight Preserving Quantum Circuits for Machine Learning. *Quantum* **9**, 1745 (2025).
65. Landman, J. et al. Quantum methods for neural networks and application to medical image classification. *Quantum* **6**, 881 (2022).
66. Kiani, B., Balestriero, R., LeCun, Y. & Lloyd, S. projUNN: efficient method for training deep networks with unitary matrices. https://doi.org/10.48550/arXiv.2203.05483 (2022).
67. Hamze, F. Parallelized Computation and Backpropagation Under Angle-Parametrized Orthogonal Matrices. https://doi.org/10.48550/arXiv.2106.00003 (2021).
68. 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).
69. Mhiri, H. et al. Constrained and Vanishing Expressivity of Quantum Fourier Models. *Quantum* **9**, 1847 (2025).
70. Bergholm, V. et al. PennyLane: Automatic differentiation of hybrid quantum-classical computations. https://doi.org/10.48550/arXiv.1811.04968 (2022).
71. Johri, S. et al. Nearest centroid classification on a trapped ion quantum computer. *npj Quantum Inf.* **7**, 1–11 (2021).
72. Bradbury, J. et al. JAX: composable transformations of Python +NumPy programs (2018).

Acknowledgements

This work was partially supported by the Plan France 2030 through the PEPR integrated project EPiQ (ANR-22-PETQ- 0007).

Author contributions

The idea was conceived by B.C. All authors contributed to the code for numerical results. Numerical experiments performed by B.C., N.M. and S.R. B.C., N.M., S.R. and I.K. contributed to the theoretical results. I.K. 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/s41534-025-01099-6.

Correspondence and requests for materials should be addressed to Natansh Mathur.