IOP Publishing Mach. Learn.: Sci. Technol. 6 (2025) 015023 https://doi.org/10.1088/2632-2153/ad9431

MACHINE LEARNING
Science and Technology

OPEN ACCESS

RECEIVED
19 July 2024

REVISED
24 October 2024

ACCEPTED FOR PUBLICATION
18 November 2024

PUBLISHED
30 January 2025

PAPER

Density matrix emulation of quantum recurrent neural networks for multivariate time series prediction

J D Viqueira[1,2,*], D Faílde[1], M M Juane[1], A Gómez[1] and D Mera[2]

[1] Galicia Supercomputing Center (CESGA), 15705 Santiago de Compostela, Spain
[2] Computer Graphics and Data Engineering (COGRADE), Departamento de Electrónica e Computación, Universidade de Santiago de Compostela, 15782 Santiago de Compostela, Spain
* Author to whom any correspondence should be addressed.

E-mail: josedaniel.viqueira@rai.usc.es

Keywords: quantum computing, machine learning, time series prediction

Abstract

Quantum recurrent neural networks (QRNNs) are robust candidates for modelling and predicting future values in multivariate time series. However, the effective implementation of some QRNN models is limited by the need for mid-circuit measurements. Those increase the requirements for quantum hardware, which in the current noisy intermediate-scale quantum era does not allow reliable computations. Emulation arises as the main near-term alternative to explore the potential of QRNNs, but existing quantum emulators are not dedicated to circuits with multiple intermediate measurements. In this context, we design a specific emulation method that relies on density matrix formalism. Using a compact tensor notation, we provide the mathematical formulation of the operator-sum representation involved. This allows us to show how the present and past information from a time series is transmitted through the circuit, and how to reduce the computational cost in every time step of the emulated network. In addition, we derive the analytical gradient and the Hessian of the network outputs with respect to its trainable parameters, which are needed when the outputs have stochastic noise due to hardware errors and a finite number of circuit shots (sampling). We finally test the presented methods using a hardware-efficient ansatz and four diverse datasets that include univariate and multivariate time series, with and without sampling noise. In addition, we compare the model with other existing quantum and classical approaches. Our results show how QRNNs can be trained with numerical and analytical gradients to make accurate predictions of future values by capturing non-trivial patterns of input series with different complexities.

1. Introduction

Processing and analysing multivariate time series, generally involve sophisticated algorithms that load high-dimensional datasets evolving in time, wherein temporal correlations are not trivial. In classical computation, machine learning is a consolidated approach for prediction and anomaly detection tasks [1–7]. Recurrent neural networks (RNNs) are a powerful tool for learning sequential data [8], and, over decades, several variations over the first model have improved their performance, like the well-known *long short-term memory* (LSTM) or the *gated recurrent unit* (GRU) cells [1, 4, 9]. Transformers are a recent alternative [10] that seems to outperform the previous neural network models. However, there is not enough experience when using this algorithm for numerical multivariate time series.

The RNN models feature great results for sequential learning, but they are not problem-free. Some models cannot store information from first inputs in long time series. The LSTM cell arose to address this problem [1]. Moreover, because of the temporal correlations between different variables in multivariate time series, a high-dimensional space appears for computing data with non-linear patterns. Thus, it is necessary to build neural networks with more neurons, layers and parameters, which are computationally expensive and more challenging to train. Rapidly, we have to tackle a Deep Learning task, which requires computational

resources and algorithms that ease the parameter optimisation during the neural network training, such as the backpropagation algorithm for estimating the gradients [11].

Quantum machine learning (QML) leverages the power of quantum computing to produce complex patterns that are probably not straightforwardly reproducible in a classical computer [12, 13]. However, in the current *noisy intermediate-scale quantum* (NISQ) era of quantum computers, there is a need for algorithms requiring shallow circuits, since only a limited number of operations are meaningful. Consequently, a substantial part of the research in quantum algorithms is focusing on the variational quantum algorithms (VQAs) [14]. A VQA lies on hybrid classical-quantum routines to classically minimise a cost function computed with a back-end parameterised quantum circuit (PQC).

In this context, a VQA can be used as a Neural Network by feeding a quantum circuit with information from our dataset and then tuning the circuit parameters to minimise the cost function [15]. The first proposals of quantum RNNs (QRNNs) consist of a temporal loop that feeds a quantum state into a quantum circuit and applies the Schrödinger equation to evolve the state. Their applications were simulating stochastic processes [16] and stochastic filtering of signals with noise [17, 18]. In the middle of the NISQ era and with the explosion of Machine Learning, several proposals have arisen to enhance the power of the current Machine Learning models by adding PQCs as part of the algorithm [19–23].

The actual QRNN model seeks to maximise the use and the power of quantum computing by a PQC that computes the whole sequence, and the only classical part is data pre-processing, data post-processing and the optimisation of circuit parameters [24–29]. The most common circuit architecture repeatedly encodes classical data into the quantum circuit, applies a unitary operator and measures a subset of qubits, involving multiple intermediate measurements before the end of the circuit. However, other proposals avoid intermediate measurements at the expense of increasing the number of qubits with the time-series length [27] or truncating the circuit [29].

The QRNN model is expected to provide advantages compared to classical neural networks, which extend to QML models in general. The encoding of classical data into quantum states allows us to compute a number of functions that increases exponentially with the number of qubits [15]. This feature enhances non-linearities that are required for learning complex patterns. In classical computing, the process requires exponential resources. Besides that, circuits with intermediate measurements, known as dynamic circuits, are not widespread in the QML literature, but they can lead to a new realm of algorithms on real quantum hardware [30], and they are being introduced in quantum computing platforms [31, 32].

The main aim of this article is to provide the mathematical tools to emulate this type of circuit, which is the core of QRNN algorithms. This is very interesting for two reasons: with the mathematical formulation, we can learn about the propagation of information through the quantum circuit, and the emulation makes the algorithm compatible with classical devices, before sending it to a quantum computer. Within this context, we test the theoretical procedure in several use cases, showing their potential for multivariate time series prediction.

The manuscript is structured as follows. In section 2 we provide the method for emulating a quantum circuit with intermediate measurements, which has the property of returning values that depend on past data encoded in the circuit, like RNNs. As backpropagation is used in classical machine learning, in section 3 we provide a method to analytically compute first- and second-order partial derivatives of the circuit outputs, by an expansion of the parameter shift rule (PSR) [15, 33]. In section 4 we show the results after applying the former methods to an emulation of a QRNN for multivariate time series prediction, in order to demonstrate that the algorithm works by emulating the quantum circuit with the presented method. Finally, discussions are included in section 5.

2. Formulation of the QRNN states with density matrices

The QRNN model is based on the classical RNN model, as proposed by [25]. Following this approach, we explicitly provide a tensor representation of the internal states propagated through the QRNN and its outputs, based on the density matrix formalism. This representation allows the implementation of a classical emulator based on tensor operations and even simple matrix operations, which are very fast in current computational devices.

2.1. The classical RNN

Consider a set of data which is time-ordered,

$$\left\{\boldsymbol{x}_{(0)}, \boldsymbol{x}_{(1)}, \cdots, \boldsymbol{x}_{(t)}, \cdots, \boldsymbol{x}_{(T)}\right\}, \tag{1}$$

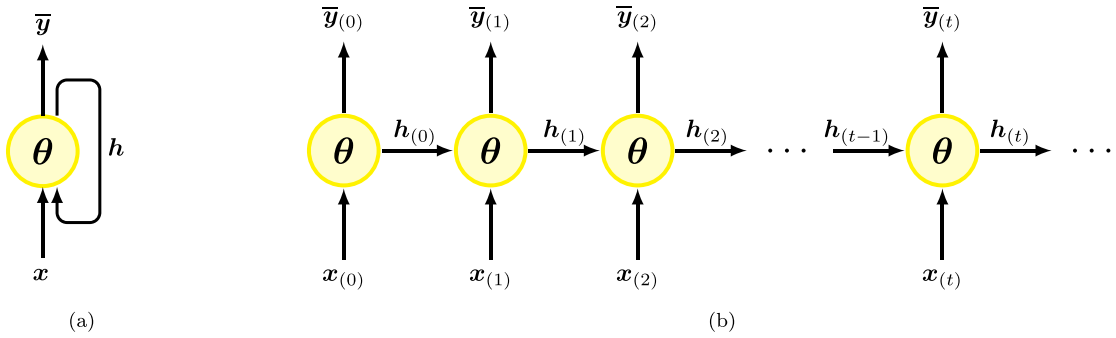

Figure 1. Classical recurrent neural network representations. (a) Basic RNN scheme. (b) Basic RNN unrolled through time.

that is the input multivariate time series, since each item is a vector containing the information of n_v variables. The output series is

$$\left\{\boldsymbol{y}_{(0)}, \boldsymbol{y}_{(1)}, \cdots, \boldsymbol{y}_{(t)}, \cdots, \boldsymbol{y}_{(T)}\right\}, \tag{2}$$

which is the target in the machine learning task.

The output of a RNN at time t depends on the inputs from the previous time steps since it preserves past information with a form of memory [34]. The easiest way to imagine a network with memory is thinking in a box that continuously receives and returns data. To start, the box is supplied with an input, $\boldsymbol{x}_{(0)}$, and it returns two objects: an output $\overline{\boldsymbol{y}}_{(0)}$ which is read, and a *hidden state* $\boldsymbol{h}_{(0)}$ which is re-introduced in the box next time. From the second time onwards, the box receives the previous hidden state $\boldsymbol{h}_{(t-1)}$ and the input $\boldsymbol{x}_{(t)}$. Then, it computes them and generates the output $\overline{\boldsymbol{y}}_{(t)}$ and the hidden state $\boldsymbol{h}_{(t)}$ to be re-introduced. The model is represented in figure 1, where we can see the recurrence. The behaviour is dynamic, in contrast to feedforward neural networks, for which the information is never transmitted backwards.

The information flux in a RNN through time splits into three different lines, represented in figure 1, that stand for *input data* $\boldsymbol{x}_{(t)}$, *output data* $\overline{\boldsymbol{y}}_{(t)}$ and *hidden state* $\boldsymbol{h}_{(t)}$. Both $\overline{\boldsymbol{y}}_{(t)}$ and $\boldsymbol{h}_{(t)}$ are functions that depend on $\boldsymbol{x}_{(t)}$ and $\boldsymbol{h}_{(t-1)}$,

$$\begin{cases} \overline{\boldsymbol{y}}_{(t)} = \mathcal{Y}\left(\boldsymbol{x}_{(t)}, \boldsymbol{h}_{(t-1)}\right) \\ \boldsymbol{h}_{(t)} = \mathcal{S}\left(\boldsymbol{x}_{(t)}, \boldsymbol{h}_{(t-1)}\right), \end{cases} \tag{3}$$

but at the same time, $\boldsymbol{h}_{(t-1)}$ also depends on previous inputs and a hidden state. Functions $\mathcal{Y}$ and $\mathcal{S}$ are the part which is expensive to compute. They consist of matrix calculations that take into account the tunable parameters (weights), structure and connections between the layers of neurons that form the network.

2.2. The QRNN

The QRNN leverages the power of PQCs implicitly performing matrix calculus. Like in the proposed classical model, wherein a network returns output data every time step, we measure the quantum circuit to obtain data in our classical devices. Quantum networks also contain a *hidden-state* flux $\boldsymbol{h}$. As its name suggests, a hidden state propagates internally and carries information not required to be known. That is why we can think of a quantum state as a hidden state. This quantum state is actually a mixed state that arises from the measurement of a subsystem A, and a subsystem B, which is not measured, provided that A and B are entangled.

Hence, we can now construct a general circuit that resembles the classical RNN described above, as it was proposed by [25]. Since a quantum circuit diagram represents the series of operations applied over several qubits (vertical direction) during the time (horizontal direction), the circuit in figure 2 can be interpreted as the unrolled representation through time of the QRNN. The circuit consists of two quantum registers: an *exchange register* (E) with n_E qubits, and a *memory register* (M) with n_M qubits. The former is used to exchange information between the quantum and the classical interface, by applying encoding and measurement operations, while the latter is never measured. The total number of qubits is n. We then have a circuit *block* for every time step: an estimation of an output from the input at time t. The instructions are (see figure 2):

1. Initialise register M to the $|0\rangle$ state.
2. Reset of register E. All register E qubits start at $|0\rangle$ every time step.

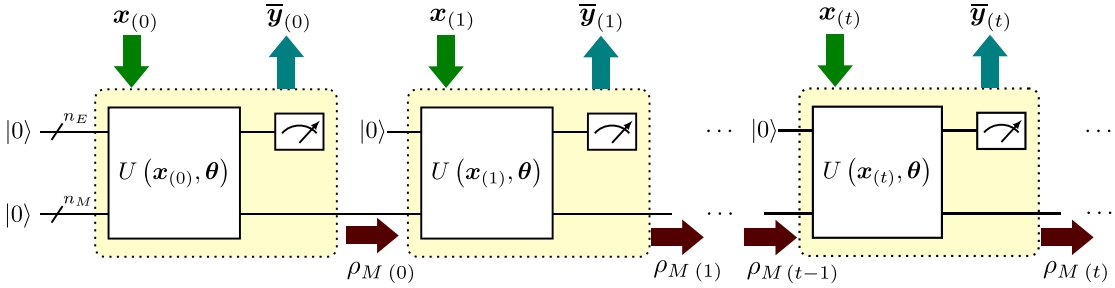

Figure 2. General form of the QRNN circuit. Arrows show the information flux.

3. Apply a parameterised unitary (ansatz) $U(\boldsymbol{x}_{(t)}, \boldsymbol{\theta})$ that evolves the state of each qubit and entangles some (or all) qubits from both registers E and M, correlating the information from both. $\boldsymbol{\theta}$ is the set of variable parameters (weights) of the network.
4. Measure qubits from register E.
5. Repeat steps (2), (3), and (4) iteratively from $t = 1$ to $t = T$.

The unitary $U(\boldsymbol{x}_{(t)}, \boldsymbol{\theta})$ must encode the classical input $\boldsymbol{x}_{(t)}$ each time step, apply parameterised gates depending on the set of weights $\boldsymbol{\theta}$, and apply entanglement between registers E and M. $\boldsymbol{\theta}$ is always the same for the different repetitions of the unitary, following the classical approach (see figure 1).

Variations of this structure are available in the literature [28] when trying to create a circuit that sustains the coherence for a longer time. This is not the scope of this work, since we want to establish some bases for designing and emulating quantum RNNs. Solutions for NISQ-era limitations in these networks will require further research that may involve quantum hardware parameters, such as the readout features or the coherence itself [35].

2.3. Memory and output of a QRNN

The way the vanilla RNN [36] (from which subsequent RNN algorithms were originated) operates data is the ideal sample of how data is explicitly processed in a RNN:

$$\begin{aligned} \boldsymbol{h}_{(t)} &= \sigma_h\left(W_{hx}\boldsymbol{x}_{(t)} + W_{hh}\boldsymbol{h}_{(t-1)} + \boldsymbol{b}_h\right), \\ \overline{\boldsymbol{y}}_{(t)} &= \sigma_y\left(W_{yh}\boldsymbol{h}_{(t)} + \boldsymbol{b}_y\right), \end{aligned} \tag{4}$$

being $\{W_{hx}, W_{hh}, W_{yh}\}$ matrices with the weights, $\{\boldsymbol{b}_h, \boldsymbol{b}_y\}$ the bias vectors, and $\{\sigma_h, \sigma_y\}$ the activation functions.

A density matrix formalism is behind the classical emulation of the quantum circuit in figure 2. The density matrix represents the state of the quantum system as an ensemble of pure quantum states, each of which is described by a state vector. Typically, the state of a quantum circuit evolves unitarily through the application of quantum gates (unitary operators) and can be described by a state vector. However, a single measurement of register E makes the register M collapse to a state $|\psi_s\rangle$, which depends on the measurement result, s, occurring with a probability p_s. From a probabilistic perspective, the measurement process gives place to an ensemble of pure states, $\{p_s, |\psi_s\rangle\}$. The density matrix is described as

$$\rho = \sum_s p_s |\psi_s\rangle\langle\psi_s|, \tag{5}$$

where $\langle\psi_s|$ is the hermitian conjugate of $|\psi_s\rangle$. The density matrix encodes all the information about the ensemble, including the post-measurement state of register M, which is referred to as a *mixed* state.

This formalism provides a powerful method for calculating the exact probability distribution in every measurement of register E qubits. At the same time, it accounts for non-unitary operations, such as the measurement itself and the reset operation. An alternative is a state vector emulation by sampling outputs, i.e. measurement probabilities are computed from multiple circuit executions, like in a real quantum device. However, exact probabilities are often preferable when emulating circuits in VQAs, especially when the optimisation method is based on gradients.

Following the instructions to build the quantum circuits, we formulate the expressions to compute two objects for every time step, as in equation (4): one is the reduced density matrix $\rho_{M\,(t)}$, which transmits information from t to $t + 1$, while the other one is the expected value of some observable $\langle O\rangle_{(t)}$ (the output

before classical post-processing). Following the density matrix formalism, we get

$$\begin{aligned} \rho_{M\,(t)} &= \mathrm{Tr}_E\left[U\,\rho_{(t)}\,U^\dagger\right], \\ \langle O\rangle_{(t)} &= \mathrm{Tr}\,\left[U\,\rho_{(t)}\,U^\dagger\,O\otimes I^{\otimes n_M}\right], \end{aligned} \tag{6}$$

where $\rho_{(t)}$ is the initial density matrix of the circuit at time t after the reset on register E and before applying the operator $U = U(\boldsymbol{x}_{(t)}, \boldsymbol{\theta})$. Here, Tr_E denotes the partial trace over subsystem E and I is the identity operator. O is our observable, which will be, without loss of generality, diagonal. If the observable were not diagonal, we could get its spectral decomposition and apply the necessary transformations to the circuit(s) before measurement. The output (prediction) at time t is

$$\bar{y}_{(t)} = f\left(\langle O\rangle_{(t)}\right), \tag{7}$$

being f an arbitrary function. We are restricting to a single-variable output. However, the generalisation for multiple variables is straightforward by considering a set of observables instead of a single one.

This section aims to derive a tensor representation of both $\rho_{M\,(t)}$ and $\langle O\rangle_{(t)}$ to (i) provide an explicit formula that can be implemented in a classical computer as matrix products, and (ii) show how the information is transmitted through the quantum circuit.

For the derivation of the formulas, a symbol and a group of r indices (lowercase letters) represent every mathematical object that is a r-rank tensor. The ordering of the indices follows the next criteria. A subindex in parenthesis refers to the current time step, and we can sometimes neglect it. The rest of the indices identify the coefficients for every projector or vector inside the Hilbert space. Indices above (below) correspond to the base vectors in the Hilbert space (dual Hilbert space). For quantum operators over the full circuit and density matrices representing the n qubits, indices are in pairs. The first index from the pair corresponds with register E, while the second one, with register M. We use the Einstein summation convention, i.e. terms are summed when upper and lower indices are repeated. Note that operations do not commute in general. See appendix A for a more comprehensive explanation of the tensor notation used and the operations involved.

The Hilbert space $\mathcal{H}_R$ for each register $R \in \{E, M\}$ has a dimension of $N_R = 2^{n_R}$ and the Hilbert space $\mathcal{H} = \mathcal{H}_E \otimes \mathcal{H}_M$ for the entire system of $n = n_E + n_M$ qubits has a dimension of $N = 2^n$.

Tensors describing states at time t and operators are defined in such a way that the following equalities hold.

- *Density matrix* describing the whole system:

$$\rho_{(t)} = \sum_{a,b,c,d} \rho^{ab}_{(t)cd} |a\otimes b\rangle\langle c\otimes d| \in L(\mathcal{H}). \tag{8}$$

- *Reduced density matrix* describing register M state:

$$\rho_{M\,(t)} = \sum_{b,d} \rho^{b}_{(t)d} |b\rangle\langle d| \in L(\mathcal{H}_M). \tag{9}$$

- *State vector* describing register E pure state after reset and V:

$$|\psi_E\rangle_{(t)} = \sum_i \left(\psi_E\right)^i_{(t)} |i\rangle \in \mathcal{H}_E. \tag{10}$$

- *Unitary operator* acting over the whole system:

$$U_{(t)} = \sum_{a,b,c,d} U^{ab}_{(t)cd} |a\otimes b\rangle\langle c\otimes d| \in L(\mathcal{H}). \tag{11}$$

- *Kraus operator* acting over the reduced density matrix:

$$B^e_{(t)} = \sum_{e,b,d} B^{eb}_{(t)d} |b\rangle\langle d| \in L(\mathcal{H}_E, \mathcal{H}_M). \tag{12}$$

- *Partial trace* over register E:

$$\begin{array}{lll} \mathrm{Tr}_E & : L(\mathcal{H}_E\otimes\mathcal{H}_M) & \to L(\mathcal{H}_M) \\ & \sum_{a,b,c,d} \rho^{ab}_{(t)cd} |a\otimes b\rangle\langle c\otimes d| & \to \sum_{a,b,c,d} \rho^{ab}_{(t)cd} |b\rangle\langle d| \delta^c_a \end{array}, \tag{13}$$

where δ^c_a is the Kronecker delta and restricts the sum to the terms having $a = c$.

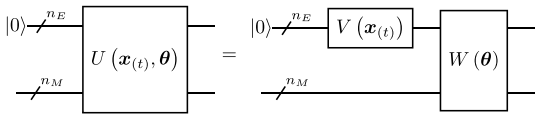

Figure 3. Decomposition of U operator into encoding (V) and evolution part (W). The latter is the same for all the circuit blocks since $\boldsymbol{\theta}$ does not change during a circuit evaluation.

- *Hermitian conjugate.* Represented by †, this operation permutes the above indexes with the below indexes (except (t)) and applies the complex conjugate.

In the preceding formulas, $L(\mathcal{X}, \mathcal{Y})$ denotes the space of bounded linear operators, and $L(\mathcal{X}) = L(\mathcal{X}, \mathcal{X})$ [37]. The symbols $|\cdot\rangle$ ($\langle\cdot|$) represent vectors in the space (dual space), such that

$$|a\rangle \in \mathcal{H}_E,\ |b\rangle \in \mathcal{H}_M,\ \langle c| \in \mathcal{H}_E^*,\ \langle d| \in \mathcal{H}_M^* \tag{14}$$

are basis vectors. The symbols above (below) can also be interpreted as row (column) indices in matrix form.

The reduced density matrix inmediately before measurement at a time t is given by

$$\left(\rho_M\right)^m_{(t)n} = U^{im}_{(t)kq}\, \rho^{kq}_{(t)lr}\, \left(U^\dagger\right)^{lr}_{(t)jn}\, \delta^j_i = \sum_{\boldsymbol{i}=0}^{N_E-1} U^{\boldsymbol{i}m}_{(t)kq}\, \rho^{kq}_{(t)lr}\, \left(U^\dagger\right)^{lr}_{(t)\boldsymbol{i}n}. \tag{15}$$

By decomposing the operator into an encoding operator $V(\boldsymbol{x}_{(t)})$ that acts only over register E and an operator $W(\boldsymbol{\theta})$ that entangles both registers (see figure 3), register E is in a pure state $\left(\psi_E\right)^i_{(t)}$ after the application of V. The resulting reduced density matrix is computed through an operator-sum representation

$$\begin{aligned}\left(\rho_M\right)^m_{(t)n} &= \sum_{\boldsymbol{i}=0}^{N_E-1} W^{\boldsymbol{i}m}_{(t)kq} \left(\left(\rho_M\right)^k_{(t-1)l} \left(\psi^\dagger_E\right)_{(t)\,r} \left(\psi_E\right)^q_{(t)}\right) \left(W^\dagger\right)^{lr}_{(t)\boldsymbol{i}n} \\ &= \sum_{\boldsymbol{i}=0}^{N_E-1} B^{\boldsymbol{i}m}_{(t)k}\, \left(\rho_M\right)^k_{(t-1)l}\, \left(B^\dagger\right)^l_{(t)\boldsymbol{i}n},\end{aligned} \tag{16}$$

where

$$B^{im}_{(t)k} \equiv W^{im}_{kq}\, \left(\psi_E\right)^q_{(t)} \tag{17}$$

are Kraus operators.

Following equation (6), the expectation value of some diagonal observable $O^i_m = d^i\, \delta^i_m$ (no summing over i since the symbol is above in both terms) at time t is

$$\langle O\rangle_{(t)} = \left(\rho'^{kl}_{ij} O^i_m \delta^j_n\right) \delta^{mn}_{kl} = \left(\rho'^{kl}_{ij} d^i\, \delta^i_m \delta^j_n\right) \delta^{mn}_{kl} = \left(\rho'^{kl}_{ij} d^i\, \delta^j_n\right) \delta^{in}_{kl} \tag{18}$$

where ρ' is the density matrix after applying the U operator.

By decomposing this density matrix, we have

$$\langle O\rangle_{(t)} = \sum_{\boldsymbol{i}=0}^{N_E-1} d^{\boldsymbol{i}} \sum_{\boldsymbol{n}=0}^{N_M-1} B^{\boldsymbol{n}}_{\boldsymbol{i}k}\, \left(\rho_M\right)^k_{(t-1)l}\, \left(B^\dagger\right)^{\boldsymbol{i}l}_{\boldsymbol{n}}, \tag{19}$$

which, again, proves the dependency of this observable with respect to both the inputs $\boldsymbol{x}_{(t)}$ and the reduced density matrix from the previous step $\rho_{M\,(t-1)}$, provided that W is an operator entangling E and M. In equation (4), it depends on the current hidden state, not the previous one. However, the recursion makes it dependent on $\boldsymbol{h}_{(t-1)}$ too. We want to remark that this output value is always a real number, due to the general properties of the density matrices.

2.4. Emulation computational cost

In most QML problems, the W operator is represented by a highly dense matrix (without null elements) and is difficult to decompose since in general it should be formed by several entanglement layers. During emulation, we build this operator once as a 4-rank tensor. After that, each time step t, N_E different $(B^i)^m_k$ operators are computed with equation (17). Therefore, we operate with these $N_M \times N_M$ matrices, avoiding the $N \times N$ matrix size in case we were using the full U operator. Moreover, equations (16) and (19) are implemented in a single routine, because the terms in (19) sum are selected from those of equation (16) with $n = m$.

Let us assume that the computational cost scales as the product of open and contracted dimensions of tensors, that is, the amount of values that the tensor indices can acquire in aggregate.

The cost of creating N_E different Kraus operators, B^e, after T time steps scales $\mathcal{O}(N_E^2 N_M^2 T)$, and the cost of generating all the items in equation (16) after T time steps scales $\mathcal{O}(N_E N_M^3 T)$. In comparison, the cost of evolving the state of the complete system directly with the operator U would scale $\mathcal{O}(N_E^3 N_M^3 T)$.

The cost of a state vector emulation under unitary evolution scales $N = 2^n$. However, each measurement process gives raise to an ensemble $\{p_s, |\psi_s\rangle\}$ of up to N_E elements. Each state vector $|\psi_s\rangle$ is evolved unitarily before the next measurement, that generates again an ensemble of N_E elements. Then, we could only recover the exact expectation values after computing $(N_E)^T$ different quantum trajectories, which arise from collapsing the state to every possibility after measurement. Another option is the state vector emulation of N_{shots} random quantum trajectories. This method consists of randomly selecting a quantum state $|\psi_s\rangle$ from the ensemble with probability p_s, which is the probability of measuring result s. Each possible measurement result corresponds to a binary string of n_E bits. The emulation needs to be repeated N_{shots} times. In this case, the outputs are not exact, but they have sampling noise, due to the emulation of a finite number of random trajectories. This sampling noise is fatal for gradients computation when finite differences methods are used.

As time series often involve several time steps and Hilbert spaces grow exponentially with the number of qubits, the most efficient emulation method is the one that uses the operator-sum representation, for most QRNN configurations when computing exact outputs. Even for cases of interest where sampling noise is acceptable, but the number of shots is never fewer than 100, the cost of the operator-sum representation will be preferable.

3. Analytical derivatives

In Machine Learning, most parameter optimisation algorithms are based on gradients, which require some method to be computed. One is numerical differentiation, which is inaccurate, and another one is symbolic differentiation, which is more computationally expensive [38]. Automatic Differentiation, with *backpropagation* as the most-known algorithm, seems to have many advantages.

In QML, as current circuits are prone to errors and the outputs are obtained after running the circuit multiple times (shots), systematic and stochastic errors seriously affect the estimation of the cost functions. Errors are amplified when computing numerical gradients, which makes it much more difficult to compute them unless we run circuits with a great number of shots and apply sophisticated error mitigation techniques. Automatic Differentiation as is done in classical computing is not possible, because we need to store intermediate results through the network and intermediate quantum states cannot be accessed in real quantum devices without affecting them. To access them, we need to measure them, and they collapse.

Despite these restrictions, the PSR [15, 33] provides a method to compute analytical gradients in quantum hardware. Hereafter, we consider circuits with parameters encoded into rotation gates $R_i(\alpha) = \mathrm{e}^{-\mathrm{i}\alpha\sigma_i}$ generated by a Pauli matrix σ_i, because they are a set of fundamental gates used for making PQCs in gate-based quantum devices.

3.1. First-order partial derivatives

We define the shift expectation value at time t, $\langle O\rangle^{\chi}_{(t)}\Big|_{ri}$, as the expectation value at time t after shifting the parameter θ_i a value χ at block $t_r \leqslant t$ in the quantum circuit. We recall that if $t_r > t$, the shift does not affect the output at time t.

For the set of gates that we consider, the shifts are $\pm\frac{\pi}{2}$ [33]. Then, in the remaining, we write only the sign.

The partial derivative of the observable at time t, $\langle O\rangle_{(t)}$, is then

$$\partial_i \langle O\rangle_{(t)} \equiv \frac{\partial\langle O\rangle_{(t)}}{\partial\theta_i} = \sum_{r=0}^{t} \frac{1}{2}\left(\langle O\rangle^{+}_{(t)} - \langle O\rangle^{-}_{(t)}\right)\Big|_{ri}, \tag{20}$$

derived in appendix B.

In classical RNNs, the chain rule propagates backwards in time, giving rise to *backpropagation through time* [39]. This method is necessary for computing gradients in RNNs, as backpropagation is used in deep learning models, but it has some caveats. Firstly, it requires computing plenty of terms because we need to repeat the network the same number of time steps we have. When contributions from the beginning of the series are negligible, a possible solution is truncation methods [40]. Secondly, the propagation backwards in time requires several weights-matrix products. Instability manifests as *vanishing* (*exploding*) gradients if the eigenvalues of the weights matrices are lower (higher) than 1 [41]. Truncation can partially address this problem [40, 42].

From equation (20), computing the partial derivative of the observable at time t requires $2t$ function evaluations with this method. Therefore, computing the gradient requires $2tN_\theta$ function evaluations, being N_θ the number of parameters of our ansatz. The loss function for training depends on the T time steps. We can evaluate the outputs in a single circuit evaluation, i.e. running the circuit a given number of times (shots) but fixing all the parameters. Thus, in general, the number of function evaluations needed to compute the exact gradient of the loss function is $2TN_\theta$. This is worse than the case of circuits without mid-circuit measurements. Previous proposals implement techniques to reduce the number of circuit evaluations [33, 43, 44], but not for this type of intermediate-measurement-based circuits.

Instability problems cannot appear in quantum neural networks in the form of *exploding gradients* because quantum circuits naturally implement unitary operations [45] before measuring. Unitary operations are a restriction of some classical ML models, precisely to avoid exploding gradients, and there are ways to implement them without losing too much expressivity [46]. In the case of a QRNN, it is straightforward to see that exploding gradients cannot appear, by looking at equation (20) since the subtraction of two observables cannot blow up, and the partial derivative only involves the summation of these terms.

The *vanishing gradient* problem does appear in VQAs, where parameterised circuits can suffer from barren plateaus in the optimisation landscape [47, 48]. In the case of RNNs, vanishing gradients cause loss of memory: the network stops depending on past inputs at some moment during the training. In the QRNN model, dependencies of past inputs gradually vanish too, because, at time t, the network starts in a mixed state, $\rho_{E\,(t)} \otimes \rho_{M\,(t-1)}$, evolved by a unitary operator. The terms from $\rho_{M\,(t-1)}$ are attenuated after these operations and many circuit blocks.

3.2. Second-order partial derivatives

Some optimisers require the computation of the Hessian matrix, apart from the Jacobian (gradient). Some authors use the PSR to compute the Hessian [49, 50] in PQCs. Here, we provide the method for computing second-order partial derivatives in the QRNN, by applying the PSR.

We define the double-shift expectation value at the time t as $\langle O\rangle_{(t)}^{\chi\lambda}\Big|_{sj}^{ri}$, which is the expectation value at time t after shifting the parameter θ_i in block t_r a value χ and the parameter θ_j a value λ in the block t_s of the quantum circuit.

The second-order partial derivatives are

$$\partial_i\partial_j\langle O\rangle_{(t)} \equiv \frac{\partial^2\langle O\rangle_{(t)}}{\partial\theta_i\,\partial\theta_j} = \frac{1}{4}\sum_r^t\sum_s^t\left(\langle O\rangle_{(t)}^{++} + \langle O\rangle_{(t)}^{--}\right)\Big|_{sj}^{ri} - -\frac{1}{4}\sum_r^t\sum_s^t\left(\langle O\rangle_{(t)}^{+-} + \langle O\rangle_{(t)}^{-+}\right)\Big|_{sj}^{ri}, \tag{21}$$

for $i \neq j$.

Moreover,

$$\partial_i^2\langle O\rangle_{(t)} \equiv \frac{\partial^2\langle O\rangle_{(t)}}{\partial\theta_i^2} = \frac{1}{2}\sum_r^t\sum_s^r\left(\langle O\rangle_{(t)}^{++} + \langle O\rangle_{(t)}^{--}\right)\Big|_{si}^{ri} - -\frac{1}{2}\sum_r^t\sum_s^{r-1}\left(\langle O\rangle_{(t)}^{+-} + \langle O\rangle_{(t)}^{-+}\right)\Big|_{si}^{ri} - \frac{1}{2}t\langle O\rangle_{(t)} \tag{22}$$

for $i = j$. These are derived in appendix B.

Considering these expressions, the symmetry $\partial_i\partial_j = \partial_j\partial_i$ and that we can recycle the intermediate calculations for the final estimation of the Hessian of the loss function, the total number of function evaluations is $2T^2N_\theta^2 + 1$ (proof in appendix B). The symmetries allow us to reduce the computational cost. However, the scaling with time steps and number of parameters is quadratic. Some approximations would be needed to improve this scaling.

Both gradient and Hessian calculations are easy to parallelise. Then, it would be plausible to simultaneously use multiple quantum processing units (QPUs) to perform several circuit evaluations. The problem is the high noise level of the current quantum hardware and different noise models for different

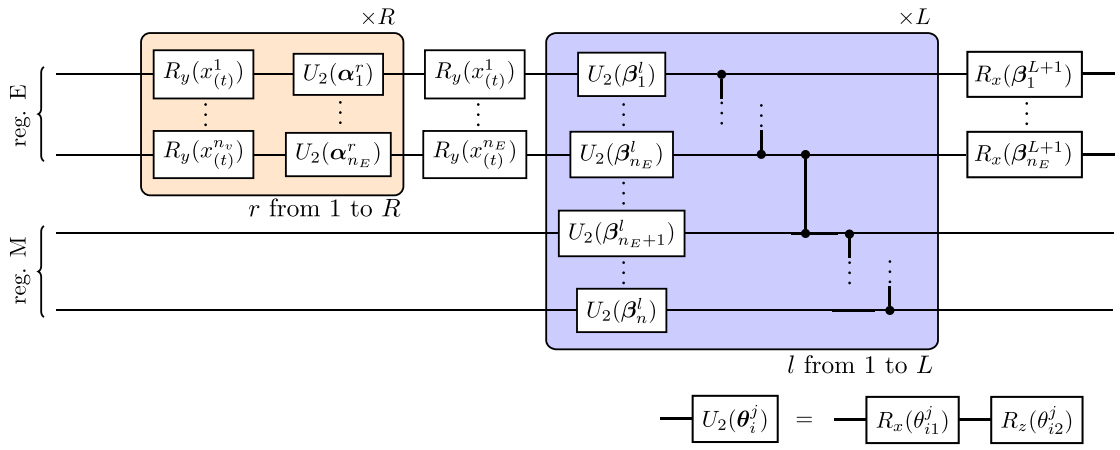

Figure 4. QRNN ansatz, $U(\boldsymbol{x}_{(t)}, \boldsymbol{\theta})$, consisting of two parts. The first one is the data encoding, and gates inside the orange box are repeated with different parameters, that are a subset of trainable parameters, $\boldsymbol{\alpha}_i^r \in \{\boldsymbol{\theta}\}$. We use one qubit per input variable. The second one is the evolution and entanglement part, where the blue box is repeated L times (layers). Each layer is a column of $R_z R_x$ rotations parameterised by a pair of parameters, $\boldsymbol{\beta}_i^l \in \{\boldsymbol{\theta}\}$, and a ladder of CZ gates. A final column of R_x gates is applied over register E before measurement, because the R_z does not affect the Z-expectation value.

QPUs that could complicate the optimisation process. Further investigation is required to see the scope of these architectures.

4. Results

We have implemented an algorithm that emulates a QRNN, by using the definitions and methods described in the previous sections. The model is based on a quantum circuit that uses a hardware-efficient ansatz. This ansatz uses layers of rotation gates for the encoding of classical inputs into the quantum circuit states, and alternates layers of single-qubit rotation gates, parameterised by the set $\boldsymbol{\theta}$, plus ladders of CZ gates.

The circuit ansatz $U(\boldsymbol{x}_{(t)}, \boldsymbol{\theta})$, used to test the performance of the QRNN model, is represented in figure 4. The encoding part $V(\boldsymbol{x}_{(t)}, \boldsymbol{\theta})$ acts only over register E qubits and repeats the encoding for each input value $R+1$ times in order to have a better expressivity [51–53]. The evolution and entanglement operator $W(\boldsymbol{\theta})$ does not vary during the circuit since $\boldsymbol{\theta}$ is fixed inside a circuit evaluation. We aim to maximise its expressibility by repeating the entangling layers several times with different parameters.

In order to test the ideal QRNN emulation and its learning capabilities, we use four datasets: (a) a dimmed triangular signal, (b) a non-linear damping signal with a sinusoidal perturbation, a set (c) consisting of two non-linear damping signals as input and a linear combination of them as output, and (d) the Santa Fe laser series [54, 55]. (a), (b) and (c) are series of 1000 points, while (d) contains 1980 points. The target is a series with temporal delays with respect to the inputs. These four examples are meaningful for two reasons: the signals are nonlinear, a feature that hinders the training, and are different enough to test several quantum circuit configurations. Moreover, example (c) demonstrates the applicability of the model to, at least, two-variable series, while (d) showcases its effectiveness in predicting a chaotic signal obtained from laboratory, and has been extensively used as a benchmark for time series prediction algorithms. All the datasets were normalized to ensure that values fall within the interval $[-1, 1]$. Details about data generation are included in appendix C.

To evaluate the performance of our model against existing algorithms, we have repeated the training with the classical RNN, LSTM and GRU models. Dataset (d) is employed to assess a Quantum Reservoir computing model described in [56]. This model consists of a non-variational quantum circuit that also incorporates intermediate measurements. We compare their results with ours in the context of predicting the Santa Fe laser series. Three tasks were made with this dataset: prediction with $t_\mathrm{d} = \{1, 5, 10\}$, with t_d defined such that $y_{(t)} = x_{(t+t_\mathrm{d})}$, i.e. t_d represents the delay between the input and the target series.

The QRNN is trained by optimising the set of parameters $\boldsymbol{\theta}$, using the Adam algorithm implemented in the open-source software framework *Pennylane* [57]. The *stepsize* parameter is set to 0.001, which is the default in the original algorithm [58]. We divide the series into windows of $T = 20$ points and the task is predicting 5 points for the output variable. The windows are divided into three sets: 20 % for testing, 20 % of the remaining for validation, and the rest for training. The distribution of validation samples is randomly generated for the three datasets. The loss function for training is the mean squared error (MSE) between the

Table 1. Quantum circuit configuration for each case analysed. In case (b), we re-upload input data in two qubits. We add the estimated number of circuit evaluations required per epoch for both numerical and analytical gradient-based parameter optimisations, considering the training size of the datasets and the number of parameters.

Case	n_E	n_M	L	R	N_θ	n_{cev} num.	n_{cev} ana.
(a)	1	2	3	3	26	864	33 312
(b)	2	2	4	1	39	1280	49 952
(c)	2	3	5	3	65	2112	83 232
(d)	1	2	5	3	38	2496	97 344

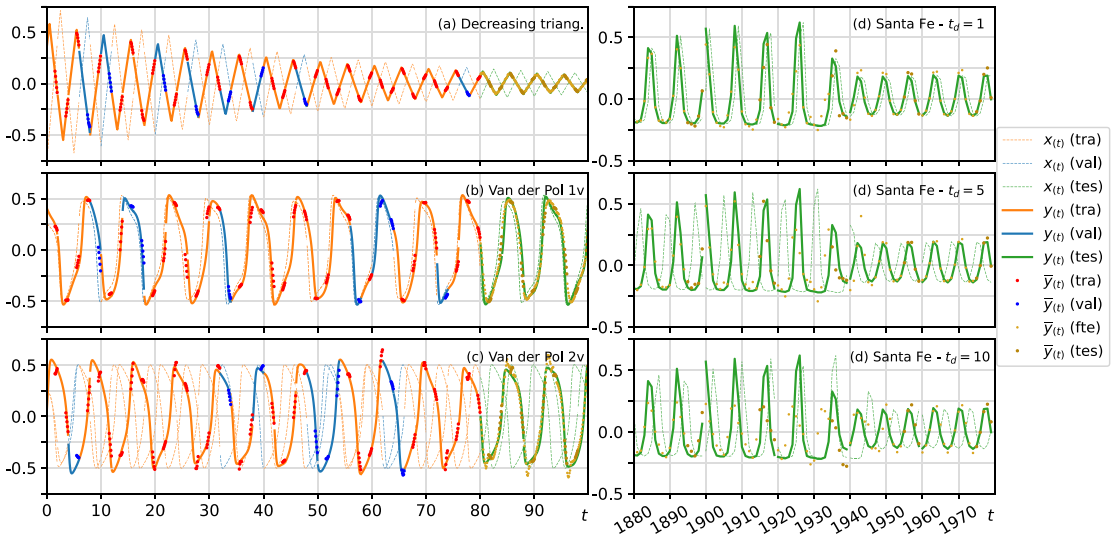

Figure 5. Results of the learning task with the QRNN model. Numerical gradients were used for the training, without sampling noise. Each time series was divided into windows of 20 points, each one being a sample. The neural network must predict the value of the last 5 outputs for each window, by reducing the RMSE with respect to the last 5 points of the target. Dashed lines are the inputs $\boldsymbol{x}_{(t)}$, from which we make the prediction. Solid lines represent the targets. Points are the predictions. Windows are represented in orange (red), blue (dark blue) or green (gold, dark gold) depending on whether the window is used for training (tra), validation (val) or testing (tes), respectively. In the test region, we add a window to make the prediction for the full test sequence (fte). For the Santa Fe series, a representative portion of the test region is displayed.

output $\bar{y}$ and the target series y of each training window (sample). The prediction for every time step is

$$\bar{y}_{(t)} = \langle Z^{\otimes n_E} \rangle_{(t)} + b, \tag{23}$$

where Z is the σ_z Pauli matrix and b is a bias, which is an extra trainable parameter.

The network hyperparameters used for each case are indicated in table 1. They were not optimised during this analysis, since the scope of this section is mainly testing the algorithm. However, we systematically increase the number of qubits and layers to have a network with sufficient complexity to make adequate predictions of non-trivial series. Indeed, in (b) we reintroduce data in the second qubit to improve the expressivity. We also want to emphasise that the patterns in (b) and (c) are not periodic. Further work would assess different ansatz architectures, hyperparameters and optimisation techniques.

We have executed 10 different parameter optimisations for each dataset, randomly initialising the rotation gates parameters in the interval $[0, 2\pi)$. For each epoch, the optimiser runs all the samples in the training dataset, in a random order (shuffle). The result of an optimisation is the set of parameters that return the lowest RMSE in the validation dataset, to avoid overfitting. The result strongly depends on the initialisation. That has consequences in the training, leading to differences in the accuracy of the predictions. Nonetheless, results never show a relative root MSE (RMSE) greater than 8.5 % of the signal amplitudes (from -0.75 to 0.75). The optimisation returning the best validation RMSE is selected as the solution among the 10 different optimisation results. After that, we run one optimisation with analytical gradients and no sampling noise, and two groups of 8 optimisations with analytical gradients and sampling noise of 10^3 and 10^4 shots, respectively. In order to make a correct comparison, the random seed for parameters initialisation and shuffle are the same as the one from the best numerical optimisation.

The Adam optimiser uses the gradient of the loss function to find its minimum. It is based on adaptive estimations of lower-order moments to change the parameters, allowing stochastic loss functions [58]. This

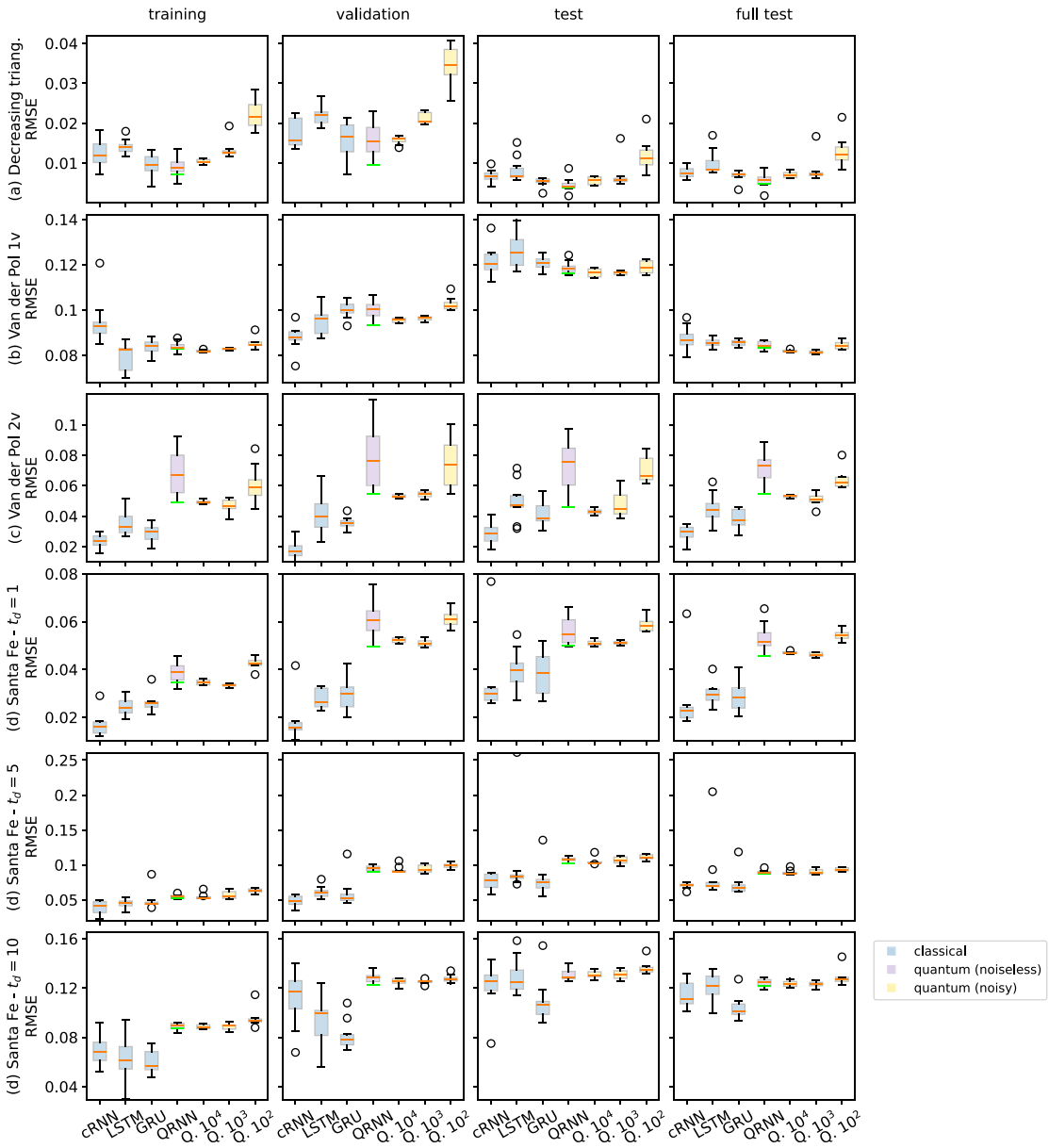

Figure 6. Comparison of RMSE results across different models used for predicting each dataset. Box plots illustrate the distributions of RMSE for the classical RNN, LSTM, GRU, and QRNN models, based on 10 realizations with varied random parameter initializations. The box plots for QRNN ('Q. N_{shots}') with finite number of shots represent 8 realisations, all initialised with the same parameter set as the noiseless optimisation giving the lowest validation RMSE. The RMSE values are calculated using outputs from a noiseless emulation of the QRNN, utilizing the parameters from the corresponding training, whether it is noiseless or noisy. The whiskers extend from the box to the farthest data point lying within 1.5 times the inter-quartile range from the box. Outliers, i.e. values past the end of the whiskers, are plotted as individual points. The green horizontal line indicates the RMSE value associated with the optimization that resulted in the lowest validation loss.

algorithm does not compute the Hessian. We use the forward difference method estimation with an absolute step size $\epsilon = 1 \times 10^{-7}$, following *Pennylane* defaults, and the analytical form of the gradient.

In ideal emulation without sampling (exact), the emulated quantum circuit returns an expectation value with a precision given by the classical machine variables (e.g. float-64). As, in general, an expectation value cannot be measured from a quantum circuit in a single shot, its evaluation by multiple repetitions (shots) of the circuit leads to stochastic variations. That makes numerical gradients very unstable and we must use analytical gradients. The PSR allows us to tackle this problem in real quantum circuits, but ideal exact emulation using numerical gradients requires fewer circuit evaluations. The density matrix emulation enables exact probabilities calculation, but also simulating stochastic noise to investigate its effect on the algorithm's performance. In the latter case, we evaluate the partial derivatives of the circuit outputs by the formulas provided in section 3, and then we apply the chain rule to calculate the gradient of the loss function.

For analytical optimisations, we have emulated the QRNN by the provided formulation, and artificially added sampling noise, corresponding to 10^2, 10^3 and 10^4 shots, which are common numbers when

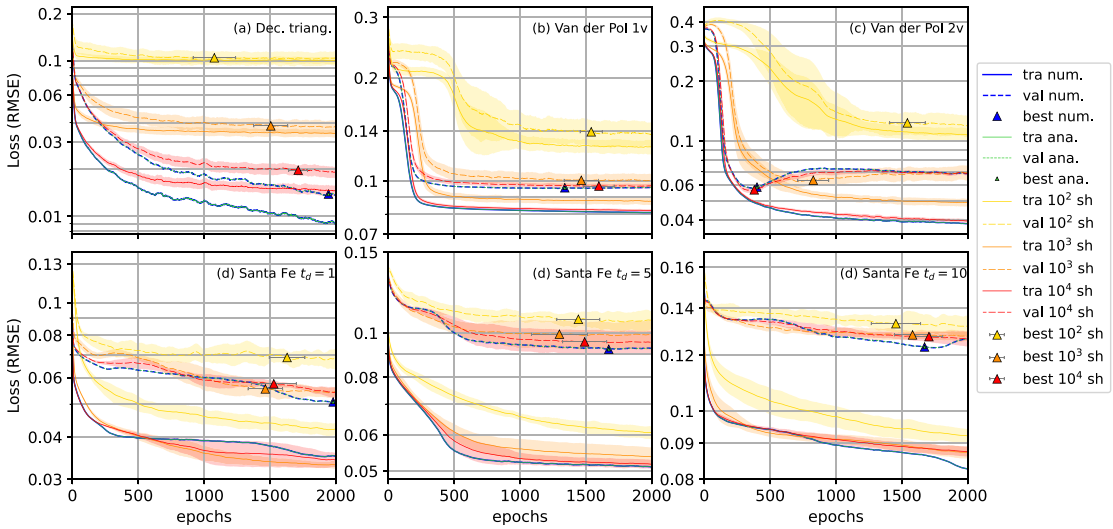

Figure 7. Training curves for each of the three datasets, showing how the value of the RMSE varies with the iterations of the parameter optimisation, for training and validation sets, and comparing results with both analytical and numerical gradients. The optimisations are those with lower final validation RMSE for numerical gradients. Optimisations with analytical gradients simulate sampling noise in every circuit evaluation with the number of shots specified in the legend. Lines of optimisations with sampling noise represent the mean of 8 different realisations and shaded regions represent the interval $\pm\sigma$ (standard deviation) around the mean. Triangles indicate the average epoch of minimum validation RMSE with the corresponding error bars representing the standard error of the mean. The curves are plotted as a moving average of 50 points to ease the visualisation.

evaluating expected values in VQAs. It is simulated with Gaussian noise of standard deviation

$$\sigma\left(\langle Z^{\otimes n_E}\rangle_{(t)}\right) = \sqrt{\frac{1-\langle Z^{\otimes n_E}\rangle^2_{(t)}}{N_{\text{shots}}}}, \tag{24}$$

according to the postulates of quantum mechanics.

The corresponding datasets and their predictions are shown in figure 5. In addition, figure 6 contains the resulting RMSE distributions of the estimated points with respect to the training, validation and test targets, always considering the last 5 points in each sample. The fourth group of RMSE destributions corresponds to the prediction of the whole test series, not only the last 5 points of each window, by shifting the input windows 5 by 5 points. Figure 7 plots the optimisation curves for the training of every dataset. Both figures 6 and 7 compare the training and the results when computing numerical gradients and analytical gradients with and without noise.

In terms of final losses, they are identical for numerical and analytical gradients without noise. The univariate output series estimations approximate to the output targets under a reasonable loss because the RMSE is, in the worst case, an order of magnitude lower than the range of output values (from -0.75 to 0.75). Sampling noise causes an effect on the final loss during the training. The loss value cannot be below a threshold that depends on the number of shots, as clearly seen in the optimisation curve of the case (a). This effect is not directly observed in cases (b) and (c) because losses in noiseless optimisations are clearly above this threshold. Analysing figure 6, the exact evaluation of the RMSE after the optimisation gives for 10^3 and 10^4 shots in every dataset region a result that slightly depends on the level of noise. In this sense, conclusions about the training capability of the QRNN with noiseless gradients can be extended to the noisy QRNN, at least for the cases studied. In contrast, $N_{\text{shots}} = 10^2$ often represents a limit for the desired precision of predictions.

The convergence curves are also identical for numerical and analytical without noise, thus giving proof of the numerical gradients being very accurate with the chosen step size ϵ. However, they are different for each case, strongly depending on the dataset. The optimiser configuration and the number of iterations were chosen to ensure a good final convergence, and not for minimising the number of epochs. For the six cases, the optimiser reaches RMSE values below 0.1 before 250 epochs with no sampling noise.

However, the optimisation slows down in cases (b) to (f) after the initial sharp drop, and validation loss stops improving from less than 500 epochs onwards. This does not happen in case (a). A possible explanation is the difficulty of finding the correct temporal correlations among input and output series. Moreover, case (c) is a representative case of overfitting, because training and validation curves substantially move away from each other. The effect of sampling noise in this dataset is clear. After the initial drop, the optimisation

Table 2. Hyperparameters of classical neural networks: input size s_{in}, hidden size s_h and the corresponding number of weights N_θ. The output size s_{out} is consistently set to 1, and the number of layers L is fixed at 1, indicating single-layered stacked neural networks.

	cRNN			LSTM			GRU		
	s_{in}	s_h	N_θ	s_{in}	s_h	N_θ	s_{in}	s_h	N_θ
(a)	1	3	22	1	1	18	1	2	33
(b)	1	4	33	1	2	43	1	2	33
(c)	2	6	67	2	2	51	2	3	67
(d)	1	4	33	1	2	43	1	2	33

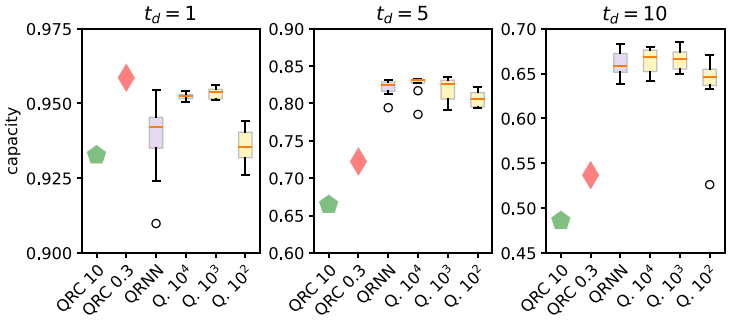

Figure 8. Comparison of results between the study in [56] and the QRNN. The first two points are the capacities in the test series from [56] (figure 5). These points correspond to measurement strength $g = 0.3$ and $g = 10$ respectively. The box plots illustrate the distribution of capacities in the full test region with 10 different trainings of the noiseless QRNN and 8 trainings. These are initialized with the parameters yielding the best RMSE validation among the 10 noiseless optimisations. The labels 'Q. N_{shots}' denote the results with noisy optimisation and analytical gradients. Reproduced from [56].

with 10^2 and 10^3 shots is slower than in the ideal emulation, because gradients are less accurate and the optimiser cannot advance so fast when evolving the parameters. This effect is almost not present when evaluating with 10^4 shots.

The number of epochs to find the best validation loss also depends on the dataset (see markers in figure 7). In (a), as the optimisation slowly continues converging, it is close to the limit, but very low values are achieved after a few epochs. Only sampling noise can avoid a high accuracy. In (b), the results show no meaningful differences after 500 epochs. In fact, the number of iterations until the solution fluctuates a lot because the curve is almost flat. The noise of 10^2 and 10^3 weakly affects the final result, looking at both the curves and the box plots. In (c), the conclusions are the same for 10^4 shots and analytical, the number of epochs increases with the level of sampling noise (inverse of number of shots). In the noisy emulations, the RMSE values tend to be higher because of the noise. In any case, the final validation is always very close to the noiseless one. Training with dataset (d) reveals plateaus in all cases following the initial drop. Thus, the average best validation point does not vary significantly, regardless of the noise level.

4.1. Comparison of QRNN with classical models

We have repeated the forecasting task using classical RNN, LSTM and GRU models, in order to address the capabilities of the QRNN model in comparison with classical common approaches. For a fair comparison, the data distribution and the optimisation procedure are the same as for the variational quantum method, using Adam with learning rate 0.001 and default parameters and sample shuffle in each epoch. Moreover, we set the neural network's hidden size, so that the number of trainable weights is as close as possible to the number of trainable parameters in QRNN. We use `pytorch.nn` recurrent models plus a linear layer, `nn.Linear`, for these trainings [59]. The total number of weights is displayed in table 2.

Classical models usually provide better results in the training and validation series, but their performance strongly depends on the dataset and the random parameter initialisation. However, in all the cases, except for (d) with delays 1 and 5, QRNN distribution and classical distributions overlap in both training and validation RMSE. In dataset (a), noiseless QRNN shows a better performance. In the test series, the noiseless quantum model shows less variation compared to the training and validation series than the classical models in most cases, especially for the Santa Fe task with the greater delay. These results are of great interest, since QRNN demonstrates the capacity of generalisation and the ability to forecast over a significant number of future time steps.

4.2. Comparison of QRNN with a QRC model

The problem of extracting information from the quantum circuit was previously addressed in [56], in the context of quantum reservoir computing. They propose weak measurements, a type of measurement that perturbs the quantum state but does not collapse or destroy it, as a resource for iteratively exchanging information with the quantum circuit and keeping quantum memory simultaneously. The memory capacity of the reservoir is evaluated for different delays, t_d, of the output series with respect to the input. That capacity, C, is measured by the squared Pearson correlation coefficient, which relates the system outputs and the targets. This metric is scale-invariant.

In figure 8 we compare the metric C between our predictions in the complete test region and their predictions in the test region. Both models show similar performances and our method achieves better accuracy for larger delays. However, variational quantum neural networks are harder to train than reservoir computing models. Thus, there is a trade-off between achieving higher accuracy and having faster training. In any case, these results also demonstrate the capabilities of the QRNN as part of the toolkit of quantum algorithms for complex time series prediction.

5. Discussion

We have used an operator-sum representation for the formulation of the hidden states and the outputs in a QRNN with intermediate measurements, and we derived its first- and second-order partial derivatives with respect to its trainable parameters. With the density matrix formulation, it is possible to directly perform an ideal emulation of the parameterised quantum circuit that makes up the network and we showed the results for a machine learning task with three different datasets. The analytical gradients permit to train networks with noisy outputs coming from quantum circuits.

The QRNN was presented in previous works [24–29]. Some of them delve into the inner structure of such quantum circuits, but there was not an explicit formulation to see how quantum states are operated through the quantum circuit. With our work, we contribute to a better understanding of how information is processed inside the network, and we compare it with the classical RNN. Moreover, the given formulas allow a direct implementation of a code for emulation, available through the link in the Data Availability section.

The operator-sum representation provides various remarkable advantages related to other possible methods. First of all, it avoids computing the complete density matrix of the n-qubits quantum state, whose number of elements scales as $\mathcal{O}(4^n)$, by considering the ansatz quantum operator W as a 4-rank tensor. Secondly, density matrix formulation allows the computation of exact expectation values in circuits with intermediate measurements, which is not feasible with state vector emulation for cases of interest, where series are made up of many points.

Consequently, the gradients can be computed with high precision in a classical computer. We have seen that parameter optimisations with numerical and analytical gradients and ideal exact emulation converge to identical values. That being said, if the precision of the variables is lost, analytical forms become necessary. By simulating Gaussian sampling noise, we see that Adam can converge to loss values very close to the noiseless evaluations, in a comparable number of epochs. Therefore, our results point in the direction of using quantum circuits with noisy outputs as predictors of multivariate time series, provided that the optimiser for training supports random variations in the loss function. The great difference between both methods is the computational cost. Analytical forms require $2T$ circuit evaluations for a gradient component, being T the number of time steps (circuit blocks inside a single circuit). Meanwhile, a forward finite difference approximation simultaneously shifts the parameter in all circuit blocks, needing only one extra circuit evaluation per gradient component. Similar conclusions should apply to the Hessian calculation, which requires a T^2 factor of circuit evaluations to be computed analytically, but it is usually approximated due to its high computational cost in most cases.

In this context, the idea of distributed quantum computing, where several QPUs work simultaneously, gathers strength. Despite the number of circuit evaluations needed, the QRNN model shows a significant feature: it does not require a lot of qubits. For the hardware-efficient ansatz and the datasets we studied, an ideal quantum processor of 5 qubits would be enough to run the circuits. In this respect, future work should assess a possible trade-off between the use of quantum computers and the use of analytical gradients with the PSR, as well as improvements on the ansatz architectures for a better trainability.

In comparison with other models, both classical and quantum, the performance of the QRNN is satisfactory, achieving similar accuracies and prediction capabilities. In that sense, QRNN can be part of the toolkit of machine learning algorithms for multivariate series prediction. As a direction for future work, to assess their actual performance when deployed on quantum hardware, we propose developing specific routines for emulating hardware noise and examining its impact on trainability and prediction accuracy.

The results are promising for potential use in real cases. To this end, the model hyperparameters optimisation to enhance the power of the neural network, the adaptability to the quantum hardware limitations, research on circuit structures that capture correlations behind complex datasets and the adaptability of different optimisation techniques are starting points towards a generalised application of QRNNs to multivariate time series prediction.

Data availability statement

The data that support the findings of this study are openly available at the following URL/DOI: https://doi.org/10.5281/zenodo.12755194 (Source Code and Training Data).

Author contributions

J D V, M M J and A G conceived the problem. J D V developed the mathematical model and programmed the emulator. D F reviewed the mathematical model. All the authors contributed to analyse the data and review the manuscript.

Acknowledgment

We thank the CESGA Quantum Computing researchers for their feedback and the stimulating intellectual environment they provide. We thank Constantino Rodríguez Ramos, especially for theoretical insights. We also thank P Mujal and R Zambrini for the feedback and for sharing their numerical results with us. This work was supported by Axencia Galega de Innovación through the Grant Agreement 'Despregamento dunha infraestructura baseada en tecnoloxías cuánticas da información que permita impulsar a I+D+I en Galicia' within the program FEDER Galicia 2014-2020. This work was partially supported by the Galician Government under Grant ED431B 2024/44. A Gómez, D Faílde and M M Juane were supported by MICIN through the European Union NextGenerationEU recovery plan (PRTR-C17.I1), and by the Galician Regional Government through the 'Planes Complementarios de I+D+I con las Comunidades Autónomas' in Quantum Communication. J. D. Viqueira was supported by Axencia Galega de Innovación (Xunta de Galicia) through the 'Programa de axudas á etapa predoutoral'. Simulations on this work were performed using Galicia Supercomputing Center (CESGA) FinisTerrae III supercomputer with financing from the Programa Operativo Plurirregional de España 2014-2020 of ERDF, ICTS-2019-02-CESGA-3, and the Qmio quantum infrastructure, with financing from the European Union, through the Programa Operativo Galicia 2014-2020 of ERDF_REACT EU, as part of the European Union's response to the COVID-19 pandemic.

Appendix A. A further explanation on tensor notation

In this paper, the mathematical formulation contains tensors of ranks 0 to 5. A main character plus a set of indices identify the tensors. This notation is valuable (i) to have more compact formulas by decreasing the bra-ket notation and (ii) to show the expression to implement in a classical emulation code.

5-rank tensors are represented by an upper shift letter that contains a lower index in parenthesis, indicating the time step (circuit block number). Let $\mathcal{H}_E$ and $\mathcal{H}_M$ be the finite-dimensional complex Hilbert spaces where the state vectors of registers E and M live, respectively, and $\mathcal{H}$ be the Hilbert space of the n qubits. The equivalence for a quantum operator in terms of Hilbert space basis vectors $\{|a\rangle, |b\rangle, \langle c|, \langle d|\}$ is

$$U_{(t)} = \sum_{a,b,c,d} U^{ab}_{(t)cd} |a \otimes b\rangle\langle c \otimes d|,\ |a\rangle \in \mathcal{H}_E, \langle c| \in \mathcal{H}_E^*, |b\rangle \in \mathcal{H}_M, \langle d| \in \mathcal{H}_M^*, \tag{A1}$$

where $U_{(t)} \in L(\mathcal{H})$ represents the quantum operator at time t, and $U^{ab}_{(t)cd} \in \mathbb{C}$ denotes the numerical value of the corresponding component, that is, the tensor component with index values a, b, c, d. Here, $L(\mathcal{H})$ stands for the space of bounded linear operators. The tensor product is represented in such a way that $|a \otimes b\rangle = |a\rangle \otimes |b\rangle$. A density matrix is represented with the same form, represented with the greek character ρ.

4-rank tensors are equivalent to 5-rank ones, but omitting the time label (t). 3-rank tensors could appear in different forms, depending on the indices positioning.

2-rank tensors also depend on the index positioning. A lowercase letter with two indices above represents the tensor component of a quantum state living in the $\mathcal{H}_E \otimes \mathcal{H}_M$ space, which satisfies the following

equation:

$$|v\rangle = \sum_{a,b} v^{ab}|a \otimes b\rangle,\ |a\rangle \in \mathcal{H}_E, |b\rangle \in \mathcal{H}_M, \tag{A2}$$

while the Hermitian conjugate is

$$\langle v| = \sum_{a,b} \left(v^\dagger\right)_{ij} \langle i \otimes j|,\ \langle i| \in \mathcal{H}_E^*, \langle j| \in \mathcal{H}_M^*. \tag{A3}$$

A symbol with one index above and one below (optionally with a time label, (t)) represents an operator or a density matrix living in one of the Hilbert spaces,

$$U = \sum_{a,c} U_c^a |a\rangle\langle c|,\ |a\rangle \in \mathcal{H}_R, \langle c| \in \mathcal{H}_R^*, R \in \{E, M\}\,. \tag{A4}$$

1-rank tensors are actually (state)vectors living in only one of the (dual) Hilbert spaces, $\mathcal{H}_R$ ($\mathcal{H}_R^*$),

$$|v\rangle = \sum_a v^a |a\rangle \text{or } w_i = b\langle i|,\ |a\rangle \in \mathcal{H}_R, R \in \{E, M\}\,, \tag{A5}$$

$$\langle v| = \sum_a v_a \langle a| \text{or } w_i = b\langle i|,\ \langle a| \in \mathcal{H}_R^*, R \in \{E, M\}\,. \tag{A6}$$

0-rank tensors are represented by a lowercase Latin symbol without any indices, and they are scalars.

We show the equivalences of operations in equation (A7). Indices can appear as single values or pairs, depending on whether the corresponding Hilbert space is $\mathcal{H} \in \{\mathcal{H}_E, \mathcal{H}_M\}$ or $\mathcal{H}_E \otimes \mathcal{H}_M$

$$\begin{array}{cll}
\text{(unitary operation)} & (U|v\rangle)^i & = U_k^i\, v^k \\
\text{(unitary product)} & (U \cdot V)_k^i & = U_m^i\, V_k^m \\
\text{(vectors tensor product)} & (|v\rangle \otimes |w\rangle)^{ij} & = v^i\, w^j \\
\text{(vectors outer product)} & (|v\rangle\langle w|)_j^i & = v^i\, w_j^* \\
\text{(matrix tensor product)} & (U \otimes V)_{kl}^{ij} & = U_k^i\, V_l^j \\
\text{(hermitian conjugate)} & \left(U^\dagger\right)_k^i & = U_i^{k*} \\
\text{(trace)} & \text{Tr}\,\rho & = \rho_{jl}^{ik}\delta_{ik}^{jl} \\
\text{(partial trace A)} & (\text{Tr}_A\,\rho)_l^k & = \rho_{il}^{ik} = \rho_{jl}^{ik}\,\delta_i^j \\
\text{(partial trace B)} & (\text{Tr}_B\,\rho)_j^i & = \rho_{jk}^{ik} = \rho_{jl}^{ik}\,\delta_k^l.
\end{array} \tag{A7}$$

Note that the tensor product operation does not commute. Therefore, the writing order does affect the final result. We use the Einstein summation convention where indices that appear twice (up and down) in a term imply the summation of that term over all the values of the index. The symbol δ is the Kronecker delta and allows to raise and lower indices, as shown in partial trace operations. In order to avoid ambiguities or to highlight the operation, operators, quantum states, and density matrices are defined in the text and explicit summation operators are shown when necessary.

In section 2.3, summation runs over N_E for indices identifying register E (first index of a pair or index of the first element in a tensor product) and N_M for indices identifying register M (second index or index of the second element).

In the case of Jacobian (gradient) and Hessian elements (section 3), they can be also thought of as tensors, and the convention is: upper indices for rows, lower indices for columns; first index of a pair for circuit block and second index of a pair for the parameter number.

Appendix B. Derivation of the PSR for a QRNN circuit

Analytical partial derivatives of expectation values from circuits based on parameterised rotations are computed as [25]

$$\partial_i \langle O\rangle = \partial_i \text{Tr}\left[U\rho_{(t)}U^\dagger O\right] = \text{Tr}\left[\partial_i U\,\rho_{(t)}\,U^\dagger\,O\right] + \text{Tr}\left[U\,\rho_{(t)}\,\partial_i U^\dagger\,O\right] + \text{Tr}\left[U\partial_i\rho_{(t)}\,U^\dagger\,O\right], \tag{B1}$$

where the first two terms can be computed with the well-known PSR, by shifting the parameter θ_i at circuit block t,

$$\begin{aligned}
\text{Tr}\left[\partial_i U\,\rho_{(t)}\,U^\dagger\,O\right] + \text{Tr}\left[U\,\rho_{(t)}\,\partial_i U^\dagger\,O\right] = \frac{1}{2}\Big\{&\text{Tr}\left[U\left(\theta_i + \frac{\pi}{2}\right)\,\rho_{(t)}\,U^\dagger\left(\theta_i + \frac{\pi}{2}\right)\,O\right] \\
&-\text{Tr}\left[U\left(\theta_i - \frac{\pi}{2}\right)\,\rho_{(t)}\,U^\dagger\left(\theta_i - \frac{\pi}{2}\right)\,O\right]\Big\},
\end{aligned} \tag{B2}$$

as in [15, 33]. For the latter term,

$$\mathrm{Tr}\left[U\,\partial_i\rho_{(t)}\,U^\dagger\,O\right]=\mathrm{Tr}\left[U\,\partial_i\left(\rho_0\otimes\rho_{M\,(t-1)}\right)\,U^\dagger\,O\right]=\mathrm{Tr}\left[U\left(\rho_0\otimes\mathrm{Tr}_E\left[\partial_i\left(U\rho_{(t-1)}U^\dagger\right)\right]\right)\,U^\dagger\,O\right], \tag{B3}$$

where $\rho_0=(|0\rangle\langle 0|)^{\otimes n_E}$. By applying the chain rule, we get again three terms,

$$\partial_i\left(U\rho_{(t-1)}U^\dagger\right)=\partial_i U\rho_{(t-1)}U^\dagger+U\rho_{(t-1)}\partial_i U^\dagger+U\partial_i\rho_{(t-1)}U^\dagger. \tag{B4}$$

The contribution from the first two is calculated by shifting the parameter θ_i twice at block $t-1$, as in equation (B2), while the third one depends on $t-2$ block. Then, the equation is recursive. Each block at $t_k<t$ contributes summing with two terms corresponding to the shifts $+\pi/2$ and $-\pi/2$. The recursion ends at $t_k=0$, then

$$\partial_i\langle O\rangle_{(t)}=\sum_{r=0}^{t}\frac{1}{2}\left(\langle O\rangle^{+}_{(t)}\Big|_{ri}-\langle O\rangle^{-}_{(t)}\Big|_{ri}\right), \tag{B5}$$

where $\langle O\rangle^{\chi}_{(t)}\Big|_{ri}$ is the expectation value at time t after a χ-shift of the parameter θ_i at time t_k, with $\chi\in\{+\pi/2,-\pi/2\}$. The shift is represented uniquely by the sign for simplicity. This is the equation in the main text, where we have indicated all the shifts positioning only after the brackets.

For the Hessian, we take every term from equation (B5), $\langle O\rangle^{\chi}_{(t)}\Big|_{ri}$, and apply again the PSR,

$$\partial_j\langle O\rangle^{\chi}_{(t)}\Big|_{ri}=\mathrm{Tr}\left(\partial_j U'\,\rho_{(t)}\,U'^\dagger\,O\right)^{\chi}\Big|_{ri}+\mathrm{Tr}\left(U'\,\rho_{(t)}\,\partial_i U'^\dagger\,O\right)^{\chi}\Big|_{ri}+\mathrm{Tr}\left(U'\,\partial_i\rho_{(t)}\,U'^\dagger\,O\right)^{\chi}\Big|_{ri}, \tag{B6}$$

being $U'=U(\theta_i+\chi)$. The subsequent derivation is now analogue to the one for the first-order PSR. From equation (B5), now

$$\partial_i\partial_j\langle O\rangle_{(t)}=\frac{1}{4}\sum_{r}^{t}\sum_{s}^{t}\left(\langle O\rangle^{++}_{(t)}\Big|^{ri}_{sj}+\langle O\rangle^{--}_{(t)}\Big|^{ri}_{sj}-\langle O\rangle^{+-}_{(t)}\Big|^{ri}_{sj}-\langle O\rangle^{-+}_{(t)}\Big|^{ri}_{sj}\right), \tag{B7}$$

where $\langle O\rangle^{\chi\lambda}_{(t)}\Big|^{ri}_{sj}$ is the expectation value at time t after a shift χ in the parameter θ_i at block r and a shift λ in the parameter θ_j at block s. Then, we have 4 different circuits per term in sums. For the case $i=j$, $\langle O\rangle^{\chi\lambda}_{(t)}\Big|^{ri}_{sj}$ is invariant under $r\leftrightarrow s$ exchange, then, we do not need to expand the full sum over s in equation (B7), but only up to r. Moreover, when $i=j$ and $r=s$, equal-sign shifts become a single π-shift and contrary-sign shifts cancel. Then,

$$\begin{aligned}\partial_i^2\langle O\rangle_{(t)}=&\frac{1}{2}\sum_{r}^{t}\sum_{s}^{r-1}\left(\langle O\rangle^{++}_{(t)}\Big|^{ri}_{si}+\langle O\rangle^{--}_{(t)}\Big|^{ri}_{si}\right)+\frac{1}{2}\sum_{r}^{t}\left(\langle O\rangle^{++}_{(t)}\Big|^{ri}_{ri}+\langle O\rangle^{--}_{(t)}\Big|^{ri}_{ri}\right)-\\&-\frac{1}{2}\sum_{r}^{t}\sum_{s}^{r-1}\left(\langle O\rangle^{+-}_{(t)}\Big|^{ri}_{si}+\langle O\rangle^{-+}_{(t)}\Big|^{ri}_{si}\right)-\frac{1}{2}\sum_{r}^{t}\langle O\rangle_{(t)},\end{aligned} \tag{B8}$$

which is equivalent to the equation in the main text. Since all the possible combinations of r and s are computed, the Hessian is symmetric under $i\leftrightarrow j$ exchange.

The number of different circuits to execute, needed to compute the Hessian of the T observables $\langle O\rangle_{(t)}$ with t from 0 to $T-1$, is then

$$4\times\frac{N_\theta\,(N_\theta-1)}{2}T^2+2\times N_\theta\frac{T(T+1)}{2}+2\times N_\theta\frac{T(T-1)}{2}+1=2N_\theta^2T^2+1. \tag{B9}$$

Appendix C. Machine learning task details

We have accomplished the machine learning of the three datasets by a classical machine emulating the ideal PQC by an algorithm based on the formulas derived in section 2. The emulation has been simply performed by matrix calculations with *NumPy*, leveraging the reductions in operations provided by the formulation. For parameter optimisation, we employed the Adam optimiser in `qml.AdamOptimizer`, from the open-source library *Pennylane* [57]. A DASK client was used for gradient components parallelisation [60].

The optimiser starts at a randomly generated point in the parameter landscape (except the bias b, which is initialised to 0), and we execute 10 different optimisations with different random initialisations. Maximum number of iterations is set to 2000.

The three different datasets were generated as explained in the following paragraphs.

Case (a) is a triangular dimmed signal described by the equation

$$s(t) = A\,\mathrm{e}^{-\mu t}\,g(P,t)\,, \tag{C1}$$

where $g(P,t)$ is a triangular signal of period P. Input series is directly got from this equation, while the reference output series is the same series several steps forward in time, i.e.

$$y(t) = s(t+t_\mathrm{d})\,. \tag{C2}$$

Case (b) is a forced Van der Pol signal, described by the differential equation

$$\frac{\mathrm{d}s^2}{\mathrm{d}t^2} - \mu\left(1-s^2\right)\frac{\mathrm{d}s}{\mathrm{d}t} + s = A\,\sin(\omega t)\,, \tag{C3}$$

where the right-hand side is a sinusoidal perturbation. The input series is

$$x(t) = c\,s(t)\,, \tag{C4}$$

while the label is

$$y(t) = c\,s(t+t_\mathrm{d})\,. \tag{C5}$$

Case (c) consists of two Van der Pol signals without perturbation, each of them described by the equation

$$\frac{\mathrm{d}s_i^2}{\mathrm{d}t^2} - \mu_i\left(1-s_i^2\right)\frac{\mathrm{d}s_i}{\mathrm{d}t} + s_i = 0. \tag{C6}$$

The input series are given by

$$x_i(t) = c_i\,s_i(t)\,, \tag{C7}$$

while the label series is

$$y(t) = d_0 x_0(t+t_0) + d_1 x_1(t+t_1)\,. \tag{C8}$$

All the series are 1000 points, filling times from $t=0$ to 100. Note that in this appendix, the value t represents the physical time, but not the time step (point) in the time series. The parameters of the dataset generators are summarised in equation (C9)

$$\begin{array}{llllllllllllllll}
\text{(a)} & A & =0.75 & \mu & =0.02 & T & =5 & t_\mathrm{d} & =12 & & & & & & & \\
\text{(b)} & A & =1 & c & =0.25 & \mu & =2 & \omega & =5 & t_\mathrm{d} & =15 & & & & & \\
\text{(c)} & c_0 & =0.25 & d_0 & =1 & \mu_0 & =2 & t_0 & =5 & c_1 & =0.25 & d_1 & =0.1 & \mu_1 & =1 & t_1 & =18.
\end{array} \tag{C9}$$

Dataset (d) was generated from the original Santa Fe laser series with the formula

$$0.75\frac{(\boldsymbol{x}_0 - \bar{x})}{\max(\mathrm{abs}(\boldsymbol{x}_0))}, \tag{C10}$$

where $\boldsymbol{x}_0$ denotes the original series and $\bar{x}$ denotes the average value.

ORCID iDs

J D Viqueira https://orcid.org/0000-0002-0041-6802
D Faílde https://orcid.org/0000-0002-7685-4331
M M Juane https://orcid.org/0009-0005-0692-7098
A Gómez https://orcid.org/0000-0001-7272-8488
D Mera https://orcid.org/0000-0002-0639-6574

References

[1] Hochreiter S and Schmidhuber J 1997 Long short-term memory *Neural Comput.* **9** 1735
[2] Zhang G, Patuwo B E and Hu M Y 1998 Forecasting with artificial neural networks: the state of the art *Int. J. Forecast.* **14** 35
[3] Zhang G P 2003 Time series forecasting using a hybrid ARIMA and neural network model *Neurocomputing* **50** 159
[4] Cho K, van Merriénboer B, Gulcehre C, Bahdanau D, Bougares F, Schwenk H and Bengio Y 2014 Learning phrase representations using RNN encoder-decoder for statistical machine translation *Proc. 2014 Conf. on Empirical Methods in Natural Language Processing (EMNLP)* (Association for Computational Linguistics) pp 1724–34
[5] Lai G, Chang W-C, Yang Y and Liu H 2018 Modeling long- and short-term temporal patterns with deep neural networks *The 41st Int. ACMSIGIR Conf. on Research & Development in Information Retrieval, SIGIR '18* (Association for Computing Machinery) pp 95–104
[6] Zhang C, Song D, Chen Y, Feng X, Lumezanu C, Cheng W, Ni J, Zong B, Chen H and Chawla N V 2019 A deep neural network for unsupervised anomaly detection and diagnosis in multivariate time series data *Proc. AAAI Conf. on Artificial Intelligence* vol 33 pp 1409–16
[7] Blázquez-García A, Conde A, Mori U and Lozano J A 2021 A review on outlier/anomaly detection in time series data *ACM Comput. Surv.* **54** 1–33
[8] Lipton Z C 2015 A critical review of recurrent neural networks for sequence learning (arXiv:1506.00019)
[9] Cho K, van Merriënboer B, Bahdanau D and Bengio Y 2014 On the properties of neural machine translation: encoder-decoder approaches *Proc. SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation* (Association for Computational Linguistics) pp 103–11
[10] Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez A N, Kaiser L and Polosukhin I 2017 Attention is all you need *Advances in Neural Information Processing Systems* vol 30 (Curran Associates, Inc.) (available at: https://papers.nips.cc/paper_files/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html)
[11] Schmidhuber J 2015 Deep learning in neural networks: an overview *Neural Netw.* **61** 85
[12] Dunjko V and Wittek P 2020 A non-review of quantum machine learning: trends and explorations *Quantum Views* **4** 32
[13] Biamonte J, Wittek P, Pancotti N, Rebentrost P, Wiebe N and Lloyd S 2017 Quantum machine learning *Nature* **549** 195
[14] Cerezo M *et al* 2021 Variational quantum algorithms *Nat. Rev. Phys.* **3** 625
[15] Mitarai K, Negoro M, Kitagawa M and Fujii K 2018 Quantum circuit learning *Phys. Rev.* A **98** 309
[16] Zak M and Williams C P 1999 Quantum recurrent networks for simulating stochastic processes *Quantum Computing and Quantum Communications*ed C P Williams (Springer) pp 75–88
[17] Behera L and Sundaram B 2004 Stochastic filtering and speech enhancement using a recurrent quantum neural network *Int. Conf. on Intelligent Sensing and Information Processing, 2004. Proc. of* pp 165–70
[18] Behera L, Kar I and Elitzur A C 2005 A recurrent quantum neural network model to describe eye tracking of moving targets *Found. Phys. Lett.* **18** 357
[19] Chen Y, Li F, Wang J, Tang B and Zhou X 2020 Quantum recurrent encoder-decoder neural network for performance trend prediction of rotating machinery *Knowl.-Based Syst.* **197** 105863
[20] Chen S Y-C, Yoo S and Fang Y-L L 2022 Quantum long short-term memory *ICASSP 2022 - 2022 IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)* pp 8622–6
[21] Elkenawy A, El-Nagar A M, El-Bardini M and El-Rabaie N M 2021 Full-state neural network observer-based hybrid quantum diagonal recurrent neural network adaptive tracking control *Neural Comput. Appl.* **33** 9221
[22] Chen S Y-C 2023 Quantum deep recurrent reinforcement learning *ICASSP 2023 - 2023 IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)* pp 1–5
[23] Tsai S-T, Fields E, Xu Y, Kuo E-J and Tiwary P 2022 Path sampling of recurrent neural networks by incorporating known physics *Nat. Commun.* **13** 7231
[24] Bausch J 2020 Recurrent quantum neural networks *Advances in Neural Information Processing Systems* vol 33, edH Larochelle, M Ranzato, R Hadsell, M F Balcan and H Lin (Curran Associates, Inc.) pp 1368–79 (available at: https://proceedings.neurips.cc/paper/2020/hash/0ec96be397dd6d3cf2fecb4a2d627c1c-Abstract.html)
[25] Takaki Y, Mitarai K, Negoro M, Fujii K and Kitagawa M 2021 Learning temporal data with a variational quantum recurrent neural network *Phys. Rev.* A **103** 414
[26] Siemaszko M, Buraczewski A, Le Saux B and Stobińska M 2023 Rapid training of quantum recurrent neural networks *Quantum Mach. Intel.* **5** 31
[27] Bondarenko D, Salzmann R and Schmiesing V-S 2023 Learning quantum processes with memory -quantum recurrent neural networks (arXiv:2301.08167)
[28] Li Y, Wang Z, Han R, Shi S, Li J, Shang R, Zheng H, Zhong G and Gu Y 2023 Quantum recurrent neural networks for sequential learning *Neural Netw.* **166** 148
[29] Sun T-P, Chen Z-Y, Xue C, Ma S-X, Liu H-Y, Wu Y-C and Guo G-P 2023 Quantum-discrete-map-based recurrent neural networks (arXiv:2305.15976)
[30] Córcoles A D, Takita M, Inoue K, Lekuch S, Minev Z K, Chow J M and Gambetta J M 2021 Exploiting dynamic quantum circuits in aquantum algorithm with superconducting qubits *Phys. Rev. Lett.* **127** 501
[31] IBM 2022 Bringing the full power of dynamic circuits to Qiskit Runtime (available at: https://research.ibm.com/blog/quantum-dynamic-circuits)
[32] Foss-Feig M *et al* 2023 Experimental demonstration of the advantage of adaptive quantum circuits (arXiv:2302.03029)
[33] Schuld M, Bergholm V, Gogolin C, Izaac J and Killoran N 2019 Evaluating analytic gradients on quantum hardware *Phys. Rev.* A **99** 331
[34] Géron A 2017 *Hands-On Machine Learning With Scikit-Learn and Tensor Flow* 1st edn (O'Reilly)
[35] Krantz P, Kjaergaard M, Yan F, Orlando T P, Gustavsson S and Oliver W D 2019 A quantum engineer's guide to superconducting qubits *Appl. Phys. Rev.* **6** 021318
[36] Elman J L 1990 Finding structure in time *Cogn. Sci.* **14** 179
[37] Wood C J, Biamonte J D and Cory D G 2015 Tensor networks and graphical calculus for open quantum systems (arXiv:1111.6950 [quant-ph])
[38] Baydin A G, Pearlmutter B A, Radul A A and Siskind J M 2017 Automatic differentiation in machine learning: a survey *J. Mach. Learn. Res.* **18** 5595
[39] Werbos P 1990 Backpropagation through time: what it does and how to do it *Proc. IEEE* **78** 1550

[40] Jaeger H 2002 Tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the echo state network approach *GMD-Forsch. Inf.* **5**
[41] Zhang A, Lipton Z C, Li M and Smola A J 2023 Dive into deep learning (arXiv:2106.11342)
[42] Tallec C and Ollivier Y 2017 Unbiasing truncated backpropagation through time (arXiv:1705.08209)
[43] Harrow A W and Napp J C 2021 Low-depth gradient measurements can improve convergence in variational hybrid quantum-classical algorithms *Phys. Rev. Lett.* **126** 502
[44] Wierichs D, Izaac J, Wang C and Lin C Y-Y 2022 General parameter-shift rules for quantum gradients *Quantum* **6** 677
[45] Benedetti M, Lloyd E, Sack S and Fiorentini M 2019 Parameterized quantum circuits as machine learning models *Quantum Sci. Technol.* **4** 043001
[46] Wisdom S, Powers T, Hershey J, Le Roux J and Atlas L 2016 Full-capacity unitary recurrent neural networks *30th Int. Conf. on Neural Information Processing Systems (NIPS'16)* ed D Lee M Sugiyama, U Luxburg, I Guyon and R Garnett (Curran Associates, Inc.) pp 4887–95
[47] McClean J R, Boixo S, Smelyanskiy V N, Babbush R and Neven H 2018 Barren plateaus in quantum neural network training landscapes *Nat. Commun.* **9** 4812
[48] Cerezo M, Sone A, Volkoff T, Cincio L and Coles P J 2021 Cost function dependent barren plateaus in shallow parametrized quantum circuits *Nat. Commun.* **12** 1791
[49] Huembeli P and Dauphin A 2021 Characterizing the loss landscape of variational quantum circuits *Quantum Sci. Technol.* **6** 025011
[50] Mitarai K and Fujii K 2019 Methodology for replacing indirect measurements with direct measurements *Phys. Rev. Res.* **1** 6
[51] Schuld M, Sweke R and Meyer J J 2021 Effect of data encoding on the expressive power of variational quantum-machine-learning models *Phys. Rev.* A **103** 430
[52] Pérez-Salinas A, Cervera-Lierta A, Gil-Fuster E and Latorre J I 2020 Data re-uploading for a universal quantum classifier *Quantum* **4** 226
[53] Casas B and Cervera-Lierta A 2023 Multidimensional Fourier series with quantum circuits *Phys. Rev.* A **107** 062612
[54] Hübner U, Abraham N B and Weiss C O 1989 Dimensions and entropies of chaotic intensity pulsations in a single-mode far-infrared NH_3 laser *Phys. Rev.* A **40** 6354
[55] Weigend A and Gershenfeld N 1993 Results of the time series prediction competition at the Santa Fe Institute *IEEE Int. Conf. on Neural Networks* (IEEE) pp 1786–93
[56] Mujal P, Martínez-Peña R, Giorgi G L, Soriano M C and Zambrini R 2023 Time-series quantum reservoir computing with weak and projective measurements *npj Quantum Inf.* **9** 16
[57] Bergholm V *et al* 2022 Pennylane: automatic differentiation of hybrid quantum-classical computations (arXiv:1811.04968 [quant-ph])
[58] Kingma D P and Ba J 2017 Adam: a method for stochastic optimization (arXiv:1412.6980)
[59] Paszke A *et al* 2019 Pytorch: an imperative style, high-performance deep learning library (arXiv:1912.01703 [cs.LG])
[60] Dask Development Team 2016 Dask: library for dynamic task scheduling (available at: http://dask.pydata.org) (Accessed July 2024)