BNL-222674-2022-COPA

QUANTUM LONG SHORT-TERM MEMORY

S. Chen

Submitted to the ICASSP 2022 (2022 IEEE International Conference on Acoustics, Speech and Signal Processing) Conference
to be held at Singapore
May 22 - 27, 2022

Computational Science Initiative

Brookhaven National Laboratory

U.S. Department of Energy

USDOE Office of Science (SC), Advanced Scientific Computing Research (SC-21)

DISCLAIMER

QUANTUM LONG SHORT-TERM MEMORY

[1]*Samuel Yen-Chi Chen,* [1]*Shinjae Yoo,* [1]*Yao-Lung L. Fang*

[1]Computational Science Initiative, Brookhaven National Laboratory, Upton, NY 11973, USA

ABSTRACT

Long short-term memory (LSTM) is a kind of recurrent neural networks (RNN) for sequence and temporal dependency data modeling and its effectiveness has been extensively established. In this work, we propose a hybrid quantum-classical model of LSTM, which we dub QLSTM. We demonstrate that the proposed model successfully learns several kinds of temporal data. In particular, we show that for certain testing cases, this quantum version of LSTM converges faster, or equivalently, reaches a better accuracy, than its classical counterpart. Due to the variational nature of our approach, the requirements on qubit counts and circuit depth are eased, and our work thus paves the way toward implementing machine learning algorithms for sequence modeling such as natural language processing, speech recognition on noisy intermediate-scale quantum (NISQ) devices.

Index Terms— Quantum machine learning, Variational quantum circuits, Long short-term memory and Recurrent neural network

1. INTRODUCTION

Recently, machine learning (ML), in particular deep learning (DL), has found tremendous success in computer vision [1, 2, 3], natural language processing [4], and mastering the game of Go [5]. One of the most commonly used ML architectures is *recurrent neural networks* (RNN), which is capable of modeling sequential data. RNNs have been applied to study several natural language processing tasks such as machine translation [4], speech recognition [6] as well as other signal processing and function approximation tasks in scientific research [7, 8].

In the meantime, quantum computers, both general- and special-purpose ones, are introduced to the general public by several technology companies such as IBM [9], Google [10], and D-Wave [11]. While in theory quantum computers can provide exponential speedup to certain classes of problems and simulations of highly-entangled physical systems that are intractable on classical computers, quantum circuits with a large number of qubits and/or a long circuit depth cannot yet be faithfully executed on these noisy intermediate-scale quantum (NISQ) devices [12] due to the lack of quantum error correction [13, 14]. Therefore, it is non-trivial to design an application framework that can be potentially executed on the NISQ devices with meaningful outcomes.

Recently, Mitarai *et al*. proposed variational quantum algorithms, circuits, and encoding schemes [15] which are potentially applicable to NISQ devices. These quantum algorithms successfully tackled several simple ML tasks, including function approximation and classification. It takes advantage of quantum entanglement [15, 16] to reduce the number of parameters in a quantum circuit, and iterative optimization procedures are utilized to update the circuit parameters. With such an iterative process, the noise in quantum devices can be effectively absorbed into the learned parameters without incorporating any knowledge of the noise properties. With these in hand, hybrid quantum-classical algorithms becomes viable and could be realized on the available NISQ devices. Such variational quantum algorithms have succeeded in classification tasks [17, 18], generative adversarial learning [19] and deep reinforcement learning [20]. However, the problem of learning sequential data, to our best knowledge, has not been investigated thoroughly in the quantum domain.

In this work, we address the issue of learning sequential, or temporal, data with quantum machine learning (QML) [21, 22, 23]. We propose a novel framework to demonstrate the feasibility of implementing RNNs with *variational quantum circuits* (VQC) — a kind of quantum circuits with gate parameters optimized (or trained) classically — and show that quantum advantages can be harvested in this scheme. Specifically, we implement long short-term memory (LSTM) — a famous variant of RNNs capable of modeling long temporal dependencies — with VQCs, and we refer to our QML architecture as *quantum* LSTM, or QLSTM for brevity. In the proposed framework, we use a hybrid quantum-classical approach, which is suitable for NISQ devices through iterative optimization while utilizing the greater expressive power granted by quantum entanglement. Through numerical simulations we show that the QLSTM learns faster (takes less epochs) than the classical LSTM does with a similar number of network parameters. In addition, the convergence of our QLSTM is more stable than its classical counterpart; specifically, no peculiar spikes that are typical in LSTM's loss functions is observed with QLSTM.

This paper is organized as follows. First, in Section 2 we introduce VQCs, the building block of the proposed framework. Next, we discuss our QLSTM architecture and its detailed mechanism in Section 3. In Section 4 we investigate through simulations the QLSTM capability for several different kinds of temporal data and compare with the outcomes of their classical counterparts. Finally, we conclude in Section 5.

2. VARIATIONAL QUANTUM CIRCUITS

VQCs are a kind of quantum circuits that have *tunable* parameters subject to iterative optimizations, see Figure1 for a generic VQC architecture. There, the $U(\mathbf{x})$ block is for the state preparation that encodes the classical data $\mathbf{x}$ into the quantum state of the circuit and is not subject to optimization, and the $V(\boldsymbol{\theta})$ block represents the variational part with *learnable* parameters $\boldsymbol{\theta}$ that will be optimized through gradient methods. Finally, we measure a subset (or all) of the qubits to retrieve a (classical) bit string like 0100.

Previous results have shown that such circuits are robust against quantum noise [24, 25, 26] and therefore suitable for the NISQ devices. VQCs have been successfully applied to function approximation [15], classification [17, 18, 27, 28], generative modeling [19], deep reinforcement learning [20], and transfer learning [29]. Furthermore, it has been pointed out that the VQCs are more expressive

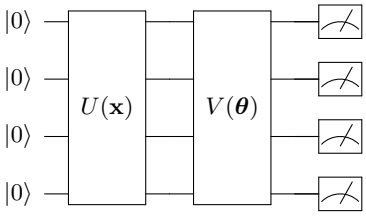

Fig. 1: Generic architecture for variational quantum circuits (VQC). $U(\mathbf{x})$ is the quantum routine for encoding the (classical) input data $\mathbf{x}$ and $V(\boldsymbol{\theta})$ is the variational circuit block with tunable parameters $\boldsymbol{\theta}$. A quantum measurement over some or all of the qubits follows.

than classical neural networks [30, 11, 16] and so are potentially better than the latter. Here, the *expressive power* refers to the ability to represent certain functions or distributions with a limited number of parameters. Indeed, artificial neural networks (ANN) are said to be *universal approximators* [31], meaning that a neural network, even with only one single hidden layer, can in theory approximate any computable function. As we will see below, using VQCs as the building blocks of quantum LSTM enables faster learning.

3. QUANTUM LSTM

In this paper, we extend the classical LSTM into the quantum realm by replacing the classical neural networks in the LSTM cells with VQCs, which would play the roles of both feature extraction and data compression, see Figure3 for a schematic of the proposed QLSTM architecture. The VQC components used in QLSTM are shown in Figure2. The VQC is composed of three major parts: *data encoding*, *variational layer* and *quantum measurements*. The data encoding circuit is used to transform the classical vector (input) into a quantum state. The variational layer is the actual learnable components, with circuit parameters updated via gradient descent algorithms. Finally, the quantum measurements are used to retrieve the values for further processing. The mathematical construction of QLSTM is given in Equation1.

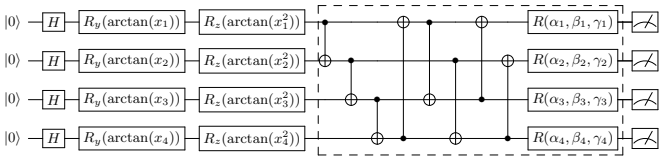

Fig. 2: Generic VQC architecture for QLSTM. It consists of three layers: the data encoding layer (with the H, R_y, and R_z gates), the variational layer (dashed box), and the quantum measurement layer. Note that the number of qubits and the number of measurements can be adjusted to fit the problem of interest, and the variational layer can contain several dashed boxes to increase the number of parameters, all subject to the capacity and capability of the quantum machines used for the experiments.

$$f_t = \sigma\left(VQC_1(v_t)\right) \tag{1a}$$
$$i_t = \sigma\left(VQC_2(v_t)\right) \tag{1b}$$
$$\tilde{C}_t = \tanh\left(VQC_3(v_t)\right) \tag{1c}$$
$$c_t = f_t * c_{t-1} + i_t * \tilde{C}_t \tag{1d}$$
$$o_t = \sigma\left(VQC_4(v_t)\right) \tag{1e}$$
$$h_t = VQC_5(o_t * \tanh\left(c_t\right)) \tag{1f}$$
$$y_t = VQC_6(o_t * \tanh\left(c_t\right)), \tag{1g}$$

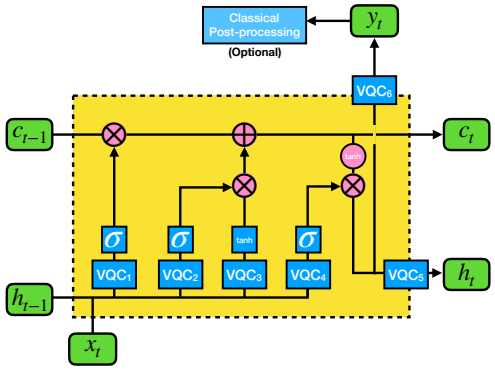

Fig. 3: The proposed quantum long short-term memory (QLSTM) architecture. Each *VQC* box is of the form as detailed in Figure2. The σ and $\tanh$ blocks represent the sigmoid and the hyperbolic tangent activation function, respectively. x_t is the input at time t, h_t is for the hidden state, c_t is for the cell state, and y_t is the output. $\otimes$ and $\oplus$ represents element-wise multiplication and addition, respectively.

4. EXPERIMENTS AND RESULTS

In this section we study and compare the capability and performance of the QLSTM with its classical counterpart. Specifically, we study QLSTM's capability to learn the representation of various functions of time. We present numerical simulations of the proposed QLSTM architecture applied to several scenarios.

To make a fair comparison, we employ a classical LSTM with the number of parameters comparable to that of the QLSTM. The classical LSTM architecture is implemented using PyTorch [32] with the hidden size 5. It has a linear layer to convert the output to a single target value y_t. The total number of parameters is 166 in the classical LSTM. As for the QLSTM, there are 6 VQCs (Figure3), in each of which we use 4 qubits with depth $= 2$ in the variational layer. In addition, there are 2 parameters for the final scaling. Therefore, the number of parameters in our QLSTM is $6\times4\times2\times3+2 = 146$. We use the same (Q)LSTM architecture throughout this section. Finally, we use PennyLane [33, 34] and Qulacs [35] for the simulation of quantum circuits, and train the QLSTM in the same PyTorch framework applied to LSTM.

We consider the following scheme for training and testing: the (Q)LSTM is expected to predict the $(N+1)$-th value given the first N values in the time sequence. For example, at step t if the input is $[x_{t-4}, x_{t-3}, x_{t-2}, x_{t-1}]$ (i.e., $N = 4$), then the QLSTM is expected to generate the output y_t, which should be close to the ground truth x_t. We set $N = 4$ throughout. For data generated by mathematical functions, we rescale them to the interval $[-1, 1]$. We use the first 67% elements in the sequence for training and the rest (33%) for testing. For each experiment, we train with maximum 100 epochs.

The optimization method is chosen to be RMSprop [36], a variant of gradient descent methods with an adaptive learning rate that updates the parameters θ as:

$$E\left[g^2\right]_t = \alpha E\left[g^2\right]_{t-1} + (1-\alpha) g_t^2, \tag{2a}$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{E\left[g^2\right]_t + \epsilon}} g_t, \tag{2b}$$

where g_t is the gradient at step t and $E\left[g^2\right]_t$ is the weighted moving average of the squared gradient with $E[g^2]_{t=0} = g_0^2$. The hyperparameters are set as follows for both LSTM and QLSTM: learning rate $\eta = 0.01$, smoothing constant $\alpha = 0.99$, and $\epsilon = 10^{-8}$.

4.1. Periodic Functions

We first investigate our QLSTM's capability in learning the sequential dependency in periodic functions. Without loss of generality we consider the sine function, a simple periodic function with constant amplitude and period:

$$y = \sin(x) \tag{3}$$

It is expected that such a function is easier to model or represent compared to functions with time-dependent amplitudes or more structure, which we discuss later. The result is shown in Figure4. By comparing the results from different epochs, it can be seen that both the QLSTM and LSTM successfully learn the sine function. While both of them converge well, we point out that the QLSTM learns significantly more information after the first training epoch than the LSTM does. For example, QLSTM's training loss at Epoch 15 is slightly lower than LSTM's (see Table 1), a trend that will become more evident later). In addition, QLSTM's loss is more stably decreasing than LSTM's; there are no spikes in the quantum case (see the right panels in Figure 4).

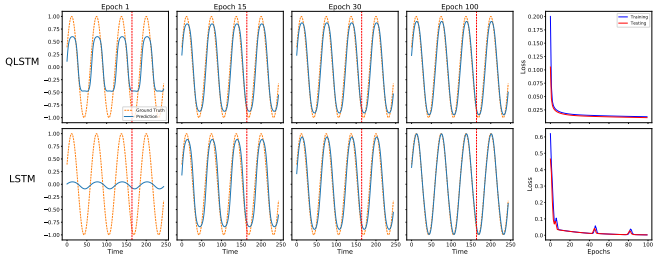

Fig. 4: Learning the sine function. QLSTM already learns the essence of $\sin(x)$ by Epoch 1, and has no peculiar bumps in the loss function. The orange dashed line represents the ground truth $\sin(x)$ [that we train the (Q)LSTM to learn] while the blue solid line is the output from the (Q)LSTM. The vertical red dashed line separates the *training* set (left) from the *testing* set (right).

	Training Loss	Testing Loss
QLSTM	1.89×10^{-2}	1.69×10^{-2}
LSTM	2.86×10^{-2}	2.81×10^{-2}

Table 1: The comparison of loss values at Epoch 15 for the sine function experiment.

4.2. Physical Dynamics

In this part of the experiments, we study the capability of the proposed QLSTM in learning the sequential dependency in physical dynamics.

4.2.1. Damped harmonic oscillator

Damped harmonic oscillators are one of the most classic textbook examples in science and engineering. It can describe or approximate a wide range of systems, from mass on a spring to electrical circuits. The differential equation describing the damped simple harmonic oscillation is,

$$\frac{\mathrm{d}^2 x}{\mathrm{d}t^2} + 2\zeta\omega_0 \frac{\mathrm{d}x}{\mathrm{d}t} + \omega_0^2 x = 0, \tag{4}$$

where $\omega_0 = \sqrt{\frac{k}{m}}$ is the (undamped) system's characteristic frequency and $\zeta = \frac{c}{2\sqrt{mk}}$ is the damping ratio. In this work, we consider a specific example from the simple pendulum with the following formulation,

$$\frac{d^2\theta}{dt^2} + \frac{b}{m}\frac{d\theta}{dt} + \frac{g}{L}\sin\theta = 0 \tag{5}$$

in which we set the system with the parameters gravitational constant $g = 9.81$, damping factor $b = 0.15$, pendulum length $l = 1$ and mass $m = 1$. The initial condition at $t = 0$ is with angular displacement $\theta = 0$ and the angular velocity $\dot{\theta} = 3$ rad/sec. We present the QLSTM learning result of the angular velocity $\dot{\theta}$.

The simulation results are shown in Figure5. Like the previous (sine) case, QLSTM surprisingly learns more on the damped oscillation as early as Epoch 1, refines faster than the classical LSTM (cf. Epoch 15), and has stabler decreasing in loss. We further note two observations: first, the testing loss values are significantly lower than the training ones. The reason is that the testing set (on the right of the red dashed line) has smaller amplitude compared to the training set. After the training, both the training and testing loss converge to a low value. Second, while both QLSTM and LSTM have undershots at the local minima/maxima (cf. Epoch 100), QLSTM's symptom is milder. In addition, QLSTM does not have overshots as seen in the LSTM (Epoch 30).

With these two case studies, we hope to establish that the QLSTM's advantages we see are a common pattern that is portable across different input functions, as we will see below.

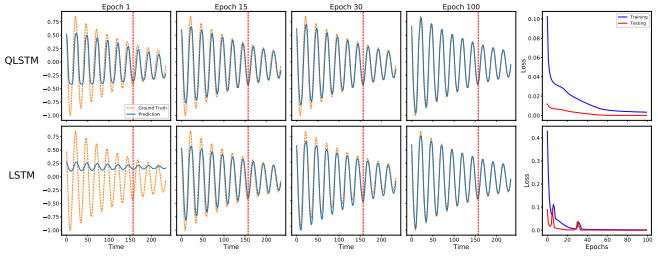

Fig. 5: Learning damped oscillations. The QLSTM learns faster and predicts more accurately than the classical LSTM with a fixed number of epochs. The orange dashed line represents the ground truth $\dot{\theta}$ [that we train the (Q)LSTM to learn] while the blue solid line is the output from the (Q)LSTM. The vertical red dashed line separates the *training* set (left) from the *testing* set (right).

	Training Loss	Testing Loss
QLSTM	2.92×10^{-2}	6×10^{-3}
LSTM	3.15×10^{-2}	5×10^{-3}

Table 2: The comparison of loss values at Epoch 15 for the damped oscillator experiment.

4.2.2. *Bessel functions*

Bessel functions of the first kind, $J_\alpha(x)$, obeys the following differential equation

$$x^2 \frac{d^2y}{dx^2} + x\frac{dy}{dx} + \left(x^2 - \alpha^2\right) y = 0, \tag{6}$$

to which the solution is

$$J_\alpha(x) = \sum_{m=0}^{\infty} \frac{(-1)^m}{m!\Gamma(m+\alpha+1)} \left(\frac{x}{2}\right)^{2m+\alpha}, \tag{7}$$

where $\Gamma(x)$ is the Gamma function. Bessel functions are also commonly encountered in physics and engineering problems, such as electromagnetic fields or heat conduction in a cylindrical geometry.

In this example, we choose J_2 for the training. The results are shown in Figure6. As in the case of damped oscillation, QLSTM learns faster, converges stabler, and has a milder symptom in undershooting. It is particularly interesting to note the poor prediction made by the LSTM at Epoch 1 and 15, in sharp contrast to that by the QLSTM.

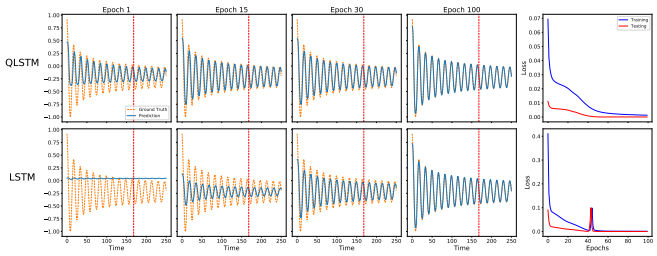

Fig. 6: Learning the Bessel function of order 2 (J_2). QLSTM's performance in prediction and convergence is even better than LSTM's with a slightly more complicated input (a non-exponential decay) compared to the previous cases. The orange dashed line represents the ground truth J_2 [that we train the (Q)LSTM to learn] while the blue solid line is the output from the (Q)LSTM. The vertical red dashed line separates the *training* set (left) from the *testing* set (right).

	Training Loss	Testing Loss
QLSTM	2.26×10^{-2}	5.5×10^{-3}
LSTM	5.43×10^{-2}	1.28×10^{-2}

Table 3: The comparison of loss values at Epoch 15 for the Bessel function J_2 experiment.

5. CONCLUSION AND OUTLOOK

We provide and study the first hybrid quantum-classical model of long short-term memory (QLSTM) which is able to learn data with temporal dependency. We show that under the constraint of similar number of parameters, the QLSTM learns significantly more information than the LSTM does right after the first training epoch, and its loss decreases more stably and faster than that of its classical counterpart. It also learns the local features (minima, maxima, etc) better than the LSTM does in general, especially when the input data has a complicated temporal structure. Our work paves the way toward using quantum circuits to model sequential data or physical dynamics, and strengthens the potential applicability of QML to many other scientific problems or commercial applications.

6. REFERENCES

[1] Karen Simonyan and Andrew Zisserman, "Very deep convolutional networks for large-scale image recognition," in *International Conference on Learning Representations*, 2015.

[2] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[3] Athanasios Voulodimos, Nikolaos Doulamis, Anastasios Doulamis, and Eftychios Protopapadakis, "Deep Learning for Computer Vision: A Brief Review," *Computational Intelligence and Neuroscience*, vol. 2018, pp. 1–13, 2018.

[4] Ilya Sutskever, Oriol Vinyals, and Quoc V Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, 2014, pp. 3104–3112.

[5] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 1 2016.

[6] Alex Graves, Navdeep Jaitly, and Abdel-rahman Mohamed, "Hybrid speech recognition with deep bidirectional lstm," in *2013 IEEE workshop on automatic speech recognition and understanding*. IEEE, 2013, pp. 273–278.

[7] Emmanuel Flurin, Leigh S Martin, Shay Hacohen-Gourgy, and Irfan Siddiqi, "Using a recurrent neural network to reconstruct quantum dynamics of a superconducting qubit from physical observations," *Physical Review X*, vol. 10, no. 1, pp. 011006, 2020.

[8] Moritz August and Xiaotong Ni, "Using recurrent neural networks to optimize dynamical decoupling for quantum memory," *Physical Review A*, vol. 95, no. 1, pp. 012335, 2017.

[9] Andrew Cross, "The ibm q experience and qiskit open-source quantum computing software," in *APS Meeting Abstracts*, 2018.

[10] Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando GSL Brandao, David A Buell, et al., "Quantum supremacy using a programmable superconducting processor," *Nature*, vol. 574, no. 7779, pp. 505–510, 2019.

[11] Trevor Lanting, Anthony J Przybysz, A Yu Smirnov, Federico M Spedalieri, Mohammad H Amin, Andrew J Berkley, Richard Harris, Fabio Altomare, Sergio Boixo, Paul Bunyk, et al., "Entanglement in a quantum annealing processor," *Physical Review X*, vol. 4, no. 2, pp. 021041, 2014.

[12] John Preskill, "Quantum computing in the nisq era and beyond," *Quantum*, vol. 2, pp. 79, 2018.

[13] Daniel Gottesman, "Stabilizer codes and quantum error correction," *arXiv preprint quant-ph/9705052*, 1997.

[14] Daniel Gottesman, "Theory of fault-tolerant quantum computation," *Physical Review A*, vol. 57, no. 1, pp. 127, 1998.

[15] Kosuke Mitarai, Makoto Negoro, Masahiro Kitagawa, and Keisuke Fujii, "Quantum circuit learning," *Physical Review A*, vol. 98, no. 3, pp. 032309, 2018.

[16] Yuxuan Du, Min-Hsiu Hsieh, Tongliang Liu, and Dacheng Tao, "The expressive power of parameterized quantum circuits," *arXiv preprint arXiv:1810.11922*, 2018.

[17] Maria Schuld, Alex Bocharov, Krysta Svore, and Nathan Wiebe, "Circuit-centric quantum classifiers," *arXiv preprint arXiv:1804.00633*, 2018.

[18] Vojtěch Havlíček, Antonio D Córcoles, Kristan Temme, Aram W Harrow, Abhinav Kandala, Jerry M Chow, and Jay M Gambetta, "Supervised learning with quantum-enhanced feature spaces," *Nature*, vol. 567, no. 7747, pp. 209–212, 2019.

[19] Pierre-Luc Dallaire-Demers and Nathan Killoran, "Quantum generative adversarial networks," *Physical Review A*, vol. 98, no. 1, pp. 012324, 2018.

[20] Samuel Yen-Chi Chen, Chao-Han Huck Yang, Jun Qi, Pin-Yu Chen, Xiaoli Ma, and Hsi-Sheng Goan, "Variational quantum circuits for deep reinforcement learning," *IEEE Access*, vol. 8, pp. 141007–141024, 2020.

[21] Maria Schuld and Francesco Petruccione, *Supervised learning with quantum computers*, vol. 17, Springer, 2018.

[22] Jacob Biamonte, Peter Wittek, Nicola Pancotti, Patrick Rebentrost, Nathan Wiebe, and Seth Lloyd, "Quantum machine learning," *Nature*, vol. 549, no. 7671, pp. 195–202, 2017.

[23] Vedran Dunjko and Hans J Briegel, "Machine learning & artificial intelligence in the quantum domain: a review of recent progress," *Reports on Progress in Physics*, vol. 81, no. 7, pp. 074001, 2018.

[24] Abhinav Kandala, Antonio Mezzacapo, Kristan Temme, Maika Takita, Markus Brink, Jerry M Chow, and Jay M Gambetta, "Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets," *Nature*, vol. 549, no. 7671, pp. 242–246, 2017.

[25] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann, "A quantum approximate optimization algorithm," *arXiv preprint arXiv:1411.4028*, 2014.

[26] Jarrod R McClean, Jonathan Romero, Ryan Babbush, and Alán Aspuru-Guzik, "The theory of variational hybrid quantum-classical algorithms," *New Journal of Physics*, vol. 18, no. 2, pp. 023023, 2016.

[27] Edward Farhi and Hartmut Neven, "Classification with quantum neural networks on near term processors," *arXiv preprint arXiv:1802.06002*, 2018.

[28] Marcello Benedetti, Erika Lloyd, Stefan Sack, and Mattia Fiorentini, "Parameterized quantum circuits as machine learning models," *Quantum Science and Technology*, vol. 4, no. 4, pp. 043001, 2019.

[29] Andrea Mari, Thomas R Bromley, Josh Izaac, Maria Schuld, and Nathan Killoran, "Transfer learning in hybrid classical-quantum neural networks," *arXiv preprint arXiv:1912.08278*, 2019.

[30] Sukin Sim, Peter D Johnson, and Alán Aspuru-Guzik, "Expressibility and entangling capability of parameterized quantum circuits for hybrid quantum-classical algorithms," *Advanced Quantum Technologies*, vol. 2, no. 12, pp. 1900070, 2019.

[31] Kurt Hornik, Maxwell Stinchcombe, Halbert White, et al., "Multilayer feedforward networks are universal approximators.," *Neural networks*, vol. 2, no. 5, pp. 359–366, 1989.

[32] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala, "PyTorch: An Imperative Style, High-Performance Deep Learning Library," in *Advances in Neural Information Processing Systems 32*, H. Wallach and H. Larochelle and A. Beygelzimer and F. d'Alché-Buc and E. Fox and R. Garnett, Ed., pp. 8024–8035. Curran Associates, Inc., 2019.

[33] Ville Bergholm, Josh Izaac, Maria Schuld, Christian Gogolin, Carsten Blank, Keri McKiernan, and Nathan Killoran, "Pennylane: Automatic differentiation of hybrid quantum-classical computations," *arXiv preprint arXiv:1811.04968*, 2018.

[34] "Pennylane-Qulacs plugin," `https://github.com/PennyLaneAI/pennylane-qulacs`, Originally written by Steven Oud (@soudy).

[35] "Qulacs," `https://github.com/qulacs/qulacs`, 2018.

[36] T. Tieleman and G. Hinton, "Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude," COURSERA: Neural Networks for Machine Learning, 2012.

[37] Maria Schuld, Ville Bergholm, Christian Gogolin, Josh Izaac, and Nathan Killoran, "Evaluating analytic gradients on quantum hardware," *Physical Review A*, vol. 99, no. 3, pp. 032331, 2019.

[38] H. J. Carmichael, *An Open Systems Approach to Quantum Optics (Lecture Notes in Physics)*, Springer, Berlin, 1993.

[39] D. F. Walls and G. J. Milburn, *Quantum Optics*, Springer, second edition, 2007.

[40] C. W. Gardiner and P. Zoller, *Quantum Noise: A Handbook of Markovian and Non-Markovian Stochastic Process with Applications to Quantum Optics*, Springer, second edition, 2000.

[41] P. Lambropoulos and D. Petrosyan, *Fundamentals of Quantum Optics and Quantum Information*, Springer-Verlag Berlin, Heidelberg, 2007.

S1. CLASSICAL MACHINE LEARNING

Here we introduce the basic concepts of classical RNNs and its variant LSTM to set the stage for the discussion for their quantum counterparts.

S1.1. Recurrent neural network

The RNNs (FigureS1) are a class of ML models that can effectively handle sequential data by memorizing previous inputs so as to make better predictions and perform temporal modeling [42, 43]. Temporal data can be processed and fed into ML models that are equipped with finite memory in the time domain. It then becomes possible to make predictions using the ML models after trained with known data, say, retrieved from experiments [7, 8, 46, 47]. For example, to design a ML model capable of generating control signals to guide the state evolution of a physical system of interest, one can train an RNN with the measured temporal data from that system by minimizing a given loss function at each time step t.

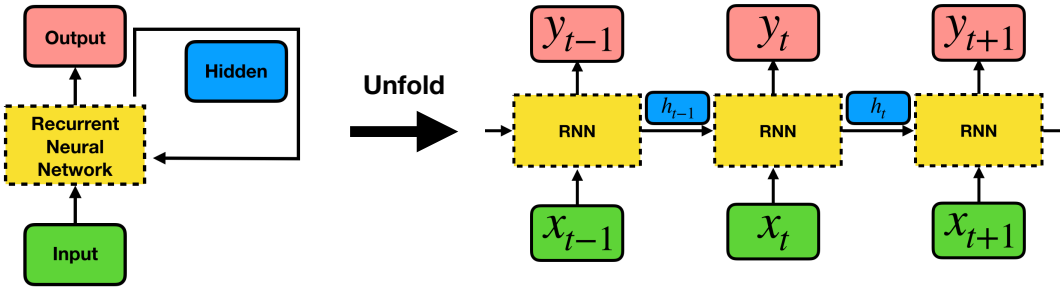

Fig. S1: A schematic for recurrent neural networks (RNN). For each time step t, the RNN takes an input value x_t, output a value o_t and a *hidden* value h_t which will be fed into itself at step $t+1$, enabling such architectures to learn temporal dependency. By unfolding the RNN in time, each time step can be seen as a unit cell of the RNN architecture.

The reason that RNNs can capture well the temporal dependency is because the network not only outputs a target value for the current time step but also keeps another value, referred to as the *hidden* state, that loops back to the network itself, making the information from the previous time steps retained. The hidden state in RNNs is critical to the memory capability. As an illustration, in FigureS1 we consider a time sequence $\{x_0, x_1, \cdots, x_n\}$ as the input of the RNNs, and their outputs are sequences as well. The input x_t is fed into the RNN at the time step t, and the returned output is y_t. At each time step, the RNN also outputs another hidden value h_t, which will be fed into itself in the next time step. This feedback mechanism is the key that distinguishes RNNs from conventional feed-forward neural networks that do not retain the information from previous steps. The information flow can be seen more clearly by *unfolding* along the time axis (see FigureS1 on the right).

S1.2. Long short-term memory

The LSTM [48] is a special kind of RNNs that can learn a longer range of sequential dependency in the data. It is one of the most popular ML approaches in sequence modeling and has found successes in a wide spectrum of applications, such as machine translation [4] and question answering [50] in natural language processing. It partially solves an important issue of *vanishing gradients* in the original RNNs: each LSTM cell at time step t has an additional *cell state*, denoted by c_t, which allows the gradients to flow unchanged and can be seen as the *memory* of the LSTM cell (so LSTM has two memory components h_t and c_t while the RNN has only h_t). This property makes the LSTM numerically more stable in training processes and predicts more accurately. These successes then further inspired RNN applications in learning quantum evolution dynamics from experimental data that also have a sequential characteristic [7, 8].

The information flow in a classical LSTM cell (FigureS2) is

$$f_t = \sigma\left(W_f \cdot v_t + b_f\right), \tag{S1a}$$
$$i_t = \sigma\left(W_i \cdot v_t + b_i\right), \tag{S1b}$$
$$\tilde{C}_t = \tanh\left(W_C \cdot v_t + b_C\right), \tag{S1c}$$
$$c_t = f_t * c_{t-1} + i_t * \tilde{C}_t, \tag{S1d}$$
$$o_t = \sigma\left(W_o \cdot v_t + b_o\right), \tag{S1e}$$
$$h_t = o_t * \tanh\left(c_t\right), \tag{S1f}$$

where σ denotes the sigmoid function, $\{W_n\}$ are classical neural networks ($n = f, i, C, o$), b_n is the corresponding bias for W_n, $v_t = [h_{t-1} x_t]$ refers to the concatenation of h_{t-1} and x_t, and the symbols $*$ and $+$ denotes element-wise multiplication and addition, respectively.

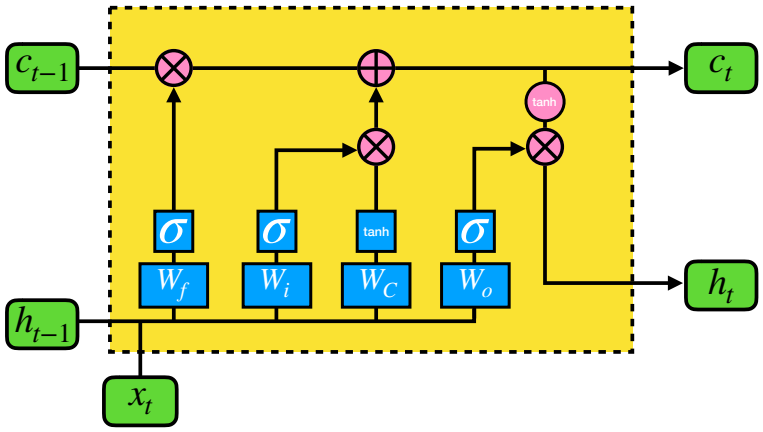

Fig. S2: A schematic for a classical long short-term memory (LSTM) cell. See the main text and Eq. (S1) for the meaning of each component.

S2. ANATOMY OF QUANTUM LONG SHORT-TERM MEMORY

S2.1. Circuit Blocks

Here we describe the building blocks for our proposed QLSTM framework. The VQC used here is presented in Figure2. Every circuit blocks used in a QLSTM cell consist of three layers: the data encoding layer, variational layer, and quantum measurement layer.

S2.1.1. Data Encoding Layer

Any classical data to be processed with a quantum circuit needs to be *encoded* into its quantum state. A general N-qubit quantum state can be represented as:

$$|\psi\rangle = \sum_{(q_1,q_2,\cdots,q_N)\in\{0,1\}} c_{q_1,q_2,\cdots,q_N} |q_1\rangle \otimes |q_2\rangle \otimes \cdots \otimes |q_N\rangle \,, \tag{S2}$$

where $c_{q_1,\cdots,q_N} \in \mathbb{C}$ is the complex *amplitude* for each basis state and each $q_i \in \{0,1\}$. The square of the amplitude $c_{q_1,\cdots,q_N}$ is the *probability* of measurement with the post-measurement state in $|q_1\rangle \otimes |q_2\rangle \otimes \cdots \otimes |q_N\rangle$ such that the total probability is equal to 1:

$$\sum_{(q_1,\cdots,q_N)\in\{0,1\}} ||c_{q_1,\cdots,q_N}||^2 = 1. \tag{S3}$$

An *encoding* scheme here refers to a predefined procedure that transforms the classical vector $\vec{v}$ into quantum amplitudes $c_{q_1,\cdots,q_N}$ that define the quantum state. In the proposed architecture, inspired by Ref. [15], the classical input vector will be transformed into rotation angles to guide the single-qubit rotations.

The first step of our encoding scheme is to transform the initial state $|0\rangle \otimes \cdots \otimes |0\rangle$ into an *unbiased* state,

$$\begin{aligned}
(H|0\rangle)^{\otimes N} &= \frac{1}{\sqrt{2^N}} (|0\rangle + |1\rangle)^{\otimes N} \\
&= \frac{1}{\sqrt{2^N}} (|0\rangle \otimes \cdots \otimes |0\rangle + \cdots + |1\rangle \otimes \cdots \otimes |1\rangle) \\
&\equiv \frac{1}{\sqrt{2^N}} \sum_{i=0}^{2^N-1} |i\rangle \,,
\end{aligned} \tag{S4}$$

where the running index i is the decimal number for the corresponding bit string that labels the computational basis.

Next, we generate $2N$ rotation angles from the N-dimensional input vector $\vec{v} = (x_1, x_2, \cdots, x_N)$ by taking $\theta_{i,1} = \arctan(x_i)$ and $\theta_{i,2} = \arctan(x_i^2)$ for each element x_i. The first angle $\theta_{i,1}$ is for rotating along the y-axis by applying the $R_y(\theta_{i,1})$ gate and $\theta_{i,2}$ for the z-axis by the $R_z(\theta_{i,2})$ gate, respectively. We choose the $\arctan$ function here, as opposed to $\arcsin$ and $\arccos$ used in Ref. [15], because in general the input values are not in the interval of $[-1, 1]$ but in $\mathbb{R}$, which is also the domain of $\arctan$. Taking x^2 is for creating higher-order terms after the entanglement operations. The unbiased state Eq. (S4) is then transformed into the desired quantum state corresponding to the classical input vector $\vec{v}$, which is to be sent to the subsequent layers. The $2N$ rotation angles are for state preparation and are not subject to iterative optimization in the present work.

S2.1.2. Variational Layer

The encoded classical data, which is now a quantum state, will then go through a series of unitary operations. These quantum operations consist of several CNOT gates and single-qubit rotation gates (dashed box in Figure2). The CNOT gates are applied to every pairs of qubits with a fixed adjacency 1 and 2 (in a cyclic way) to generate multi-qubit entanglement. The 3 rotation angles $\{\alpha_i, \beta_i, \gamma_i\}$ along the axes x, y, and z, respectively, in the single-qubit rotation gates $\{R_i = R(\alpha_i, \beta_i, \gamma_i)\}$ are not fixed in advance; rather, they are to be updated in the iterative optimization process based on a gradient descent method. Note that the dashed box may repeat several times to increase the depth of this layer and thus the number of variational parameters. In this study, we set the depth to 2 in all experiments.

S2.1.3. Quantum Measurement Layer

The end of every VQC block is a quantum measurement layer. Here we consider the expectation values of every qubit by measuring in the computational basis. With quantum simulation software such as PennyLane [33] and IBM Qiskit [53], it can be calculated numerically on a classical computer, whereas with real quantum computers, such values are statistically estimated through repeated measurements, which should be in theory close to the value obtained from simulation in the zero-noise limit. The returned result is a fixed-length vector to be further processed on a classical computer. In the proposed QLSTM, the measured values from each of the VQCs will be processed within a QLSTM cell, to be discussed in the next section.

S2.2. Stack All the Blocks

To construct the basic unit of the proposed QLSTM architecture, a QLSTM cell, we stack the aforementioned VQC blocks together. In Figure3, each of the VQC_i block is described in the previous section (see also Figure2). There are six VQCs in a QLSTM cell. For VQC_1 to VQC_4, the input is the concatenation v_t of the hidden state h_{t-1} from the previous time step and the current input vector x_t, and the output is four vectors obtained from the measurements at the end of each VQCs. The measured values, which are Pauli Z expectation values of each qubit by design, then go through nonlinear activation functions (sigmoid and $\tanh$).

A formal mathematical formulation of a QLSTM cell is given by [cf. Eq. (S1) for classical LSTM]

$$f_t = \sigma\left(VQC_1(v_t)\right) \tag{S5a}$$
$$i_t = \sigma\left(VQC_2(v_t)\right) \tag{S5b}$$
$$\tilde{C}_t = \tanh\left(VQC_3(v_t)\right) \tag{S5c}$$
$$c_t = f_t * c_{t-1} + i_t * \tilde{C}_t \tag{S5d}$$
$$o_t = \sigma\left(VQC_4(v_t)\right) \tag{S5e}$$
$$h_t = VQC_5(o_t * \tanh\left(c_t\right)) \tag{S5f}$$
$$y_t = VQC_6(o_t * \tanh\left(c_t\right)), \tag{S5g}$$

which can be grouped into three layers for their purposes:

- **Forget Block** [Eq. (S5a)]: The VQC_1 block examines v_t and outputs a vector f_t with values in the interval $[0, 1]$ through the sigmoid function. The purpose of f_t is to determine whether to "forget" or "keep" the corresponding elements in the cell state c_{t-1} from the previous step, by operating element-wisely on c_{t-1} (i.e., $f_t * c_{t-1}$). For example, a value 1 (0) means that the corresponding element in the cell state will be completely kept (forgotten). In general, though, the vector operating on the cell state is not 0 or 1 but something in between, meaning that a part of the information carried by the cell state will be kept, making (Q)LSTM suitable to learn or model the temporal dependencies.
- **Input and Update Block** [Eqs. (S5b)-(S5d)]: The purpose of this part is to decide what new information will be added to the cell state. There are two VQCs in this part. First, VQC_2 processes v_t, and the output then goes through the sigmoid function so as to determine which values will be added to the cell state. In the meanwhile, VQC_3 processes the same concatenated input and passes through a tanh function to generate a new cell state candidate $\tilde{C}_t$. Finally, the result from VQC_2 is multiplied element-wisely by $\tilde{C}_t$, and the resulting vector is then used to update the cell state.
- **Output Block** [Eqs. (S5e)-(S5g)]: After the updates of the cell state, the QLSTM cell is ready to decide what to output. First, VQC_4 processes v_t and goes through the sigmoid function to determine which values in the cell state c_t are relevant to the output. The cell state itself goes through the $\tanh$ function and then is multiplied element-wisely by the result from VQC_4. This value can then be further processed with VQC_5 to get the hidden state h_t or VQC_6 to get the output y_t.

For a given problem size, the total number of qubits used in a VQC block is determined so as to match the dimension of the input vector $v_t = [h_{t-1}x_t]$ to that of the QLSTM cell, and the number of qubits to be measured is of the dimension of the *hidden state* of the QLSTM. In general, the dimensions of the cell state c_t, the hidden state h_t and the output y_t are not the same. To ensure we have the correct dimensions of these vectors and keep the flexibility of designing the architecture, we include VQC_5 to transform c_t to h_t, and likewise VQC_6 to transform c_t to y_t.

S2.3. Optimization Procedure

In the optimization procedure, we employ the *parameter-shift* method [37, 33] to derive the analytical gradient of the quantum circuits. For example, given the expectation value of an observable $\hat{B}$

$$f\left(x;\theta_i\right)=\left\langle 0\left|U_0^{\dagger}(x)U_i^{\dagger}\left(\theta_i\right)\hat{B}U_i\left(\theta_i\right)U_0(x)\right|0\right\rangle=\left\langle x\left|U_i^{\dagger}\left(\theta_i\right)\hat{B}U_i\left(\theta_i\right)\right|x\right\rangle, \tag{S6}$$

where x is the input value, $U_0(x)$ is the state preparation routine to encode x into the quantum state, i is the circuit parameter index for which the gradient is to be calculated, and $U_i(\theta_i)$ is the single-qubit rotation generated by the Pauli operators, it can be shown [15] that the gradient of f with respect to the parameter θ_i is

$$\nabla_{\theta_i}f(x;\theta_i)=\frac{1}{2}\left[f\left(x;\theta_i+\frac{\pi}{2}\right)-f\left(x;\theta_i-\frac{\pi}{2}\right)\right]. \tag{S7}$$

This allows us to analytically evaluate the gradients of the expectation values and apply the *gradient descent* optimization from classical ML to VQC-based ML models.

S3. QLSTM FOR QUANTUM CONTROL PROBLEMS

In this section, we consider several interesting problems in open quantum systems (OQS). We again demonstrate the superiority of QLSTM over LSTM in efficiently learning sequential data. The results presented in this section, although with much harder datasets, are consistent with the results in the main text.

S3.0.1. Delayed Quantum Control

We consider a syetem with delayed quantum feedback: a two-level atom (or qubit) coupled to a semi-infinite, one-dimensional waveguide, one end of which is terminated by a perfect mirror that 100% reflects any incoming propagating photons. This system can be cast to an OQS problem by treating the waveguide as the environment seen by the qubit, and in this context it is known to be non-Markovian [54, 55, 56], in particular when the qubit-mirror separation, denoted by L, is an integer multiple of the qubit's resonant wavelength λ_0. Due to the delayed feedback (photons taking round trips to bounce in-between the qubit and the mirror) a bound state in the continuum (BIC) is formed in this case, causing a portion of incoming photons trapped in the interspace between the qubit and the mirror [57, 54, 58]. By "shaking", or modulating, the qubit frequency in time so as to change λ_0 and break the resonant condition, the trapped photon can be released to the waveguide and detected by measuring the output field intensity [54]. In FigureS3 we learn this temporal dependence using (Q)LSTM.

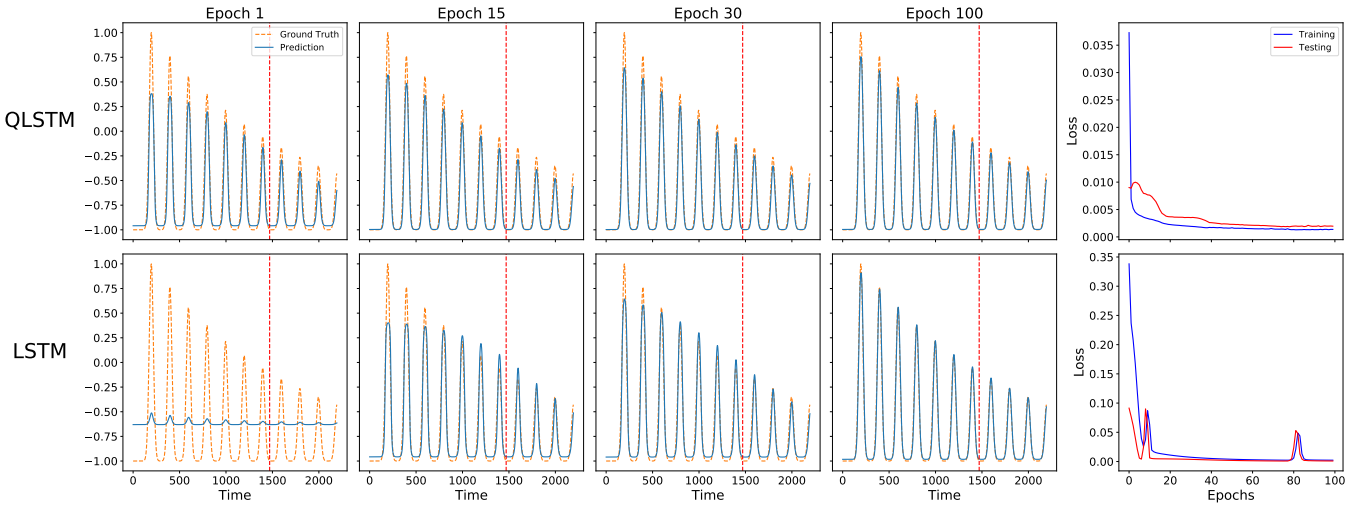

Fig. S3: Learning the dynamics with delayed quantum feedback. The QLSTM fits the local minima better than the LSTM does. The orange dashed line represents the ground truth [that we train the (Q)LSTM to learn] while the blue solid line is the output from the (Q)LSTM. The vertical red dashed line separates the *training* set (left) from the *testing* set (right).

In this example, we consider a sinusoidal modulation of the qubit frequency such that the average frequency satisfies the resonant condition [54], and the result is shown in FigureS3. Not only are QLSTM's advantages carried over to this case (without surprise by now we hope), but it also predicts better at the local minima than the LSTM does (cf. Epoch 100). In particular, note that QLSTM's training loss is almost one order of magnitude smaller than LSTM's by Epoch 15 (see Table S1).

	Training Loss	Testing Loss
QLSTM	2.88×10^{-3}	5.7×10^{-3}
LSTM	1.44×10^{-2}	4.7×10^{-3}

Table S1: The comparison of loss values at Epoch 15 for the delayed quantum control experiment.

S3.0.2. Population Inversion

Finally, we consider a textbook OQS problem: a simple cavity quantum electrodynamics (CQED) system [38, 39, 40], in which a qubit coherently interacts with a cavity, both subject to possible loss to the environment. CQED systems have been used as a cornerstone in quantum computing and quantum information science, ranging from superconducting quantum computers [59] to optical quantum networks [60], due to its conceptual simplicity and yet high tunability and controllability.

By preparing the cavity in a coherent state

$$|\alpha\rangle = \exp\left(-|\alpha|^2/2\right) \sum_{n=0}^{\infty} \frac{\alpha^n}{\sqrt{n!}} |n\rangle \tag{S8}$$

with a complex-valued amplitude α at $t = 0$ and letting it evolve in time, a population death and revival of the qubit can be observed [41], meaning the probabilities p_g and p_e of finding the qubit in its ground state $|g\rangle$ and excited state $|e\rangle$, respectively, oscillate in time. This is due to the interference among all possible bosonic number states $|n\rangle$ where an excitation can leave the qubit and goes to (and vice versa). This can be characterized by the population inversion

$$D(t) = p_g(t) - p_e(t) = \sum_{n=0}^{\infty} e^{-|\alpha|^2} \frac{|\alpha|^{2n}}{n!} \cos\left(2g\sqrt{n+1}t\right), \tag{S9}$$

where g is the qubit-cavity coupling.

In FigureS4 we study $D(t)$ with $g = 1$, $\bar{n} = |\alpha|^2 = 40$, and the summation truncated to $n_{max} = 100$. The QLSTM outperforms the LSTM, as before, in the learning speed, accuracy, and convergence stability. It is interesting to note that the LSTM has a hard time learning the zero offset (when $p_g = p_e$ s.t. $D = 0$): at Epoch 15 and 30, for example, the LSTM has a large nonzero offset whereas the QLSTM already learns this feature. Also, QLSTM's training loss is (again) one order of magnitude smaller than LSTM's by Epoch 15 (see Table S2).

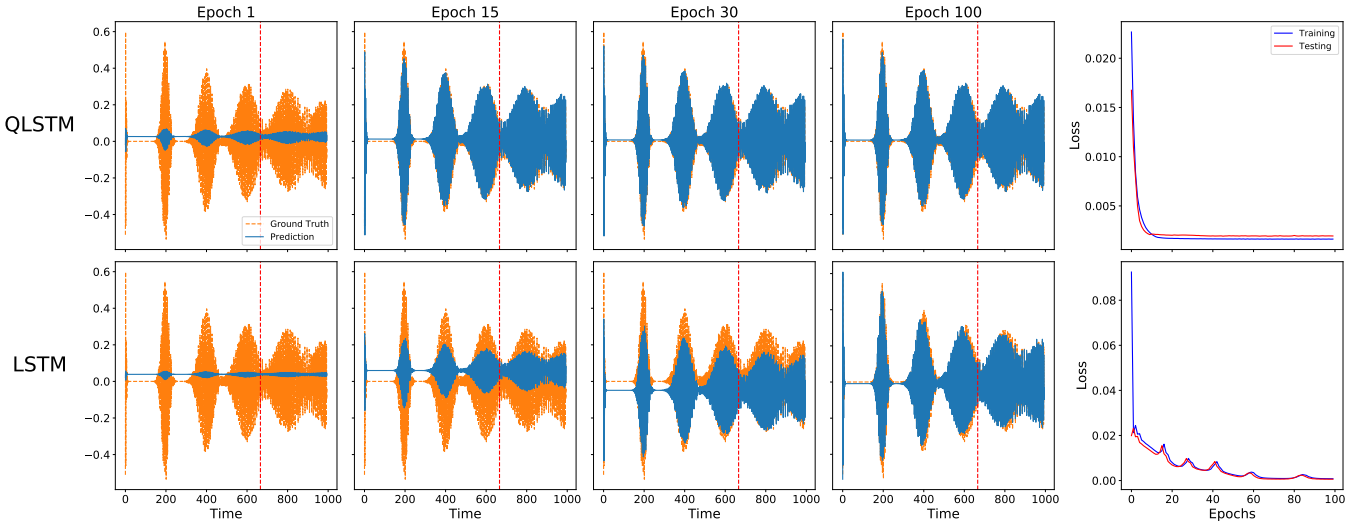

Fig. S4: Learning the population inversion. The QLSTM predicts better than the LSTM, in particular when the populations in the ground and excited states are balanced ($D = 0$). The orange dashed line represents the ground truth $D(t)$ [that we train the (Q)LSTM to learn] while the blue solid line is the output from the (Q)LSTM. The vertical red dashed line separates the *training* set (left) from the *testing* set (right).

	Training Loss	Testing Loss
QLSTM	1.78×10^{-3}	2.1×10^{-3}
LSTM	1.25×10^{-2}	1.26×10^{-2}

Table S2: The comparison of loss values at Epoch 15 for the population inversion experiment.

S4. REFERENCES

[42] Guokun Lai, Zihang Dai, Yiming Yang, and Shinjae Yoo, "Re-examination of the role of latent variables in sequence modeling," in *Advances in Neural Information Processing Systems*, 2019, pp. 7812–7822.

[43] Ian Goodfellow, Yoshua Bengio, and Aaron Courville, *Deep learning*, MIT press, 2016.

[44] Emmanuel Flurin, Leigh S Martin, Shay Hacohen-Gourgy, and Irfan Siddiqi, "Using a recurrent neural network to reconstruct quantum dynamics of a superconducting qubit from physical observations," *Physical Review X*, vol. 10, no. 1, pp. 011006, 2020.

[45] Moritz August and Xiaotong Ni, "Using recurrent neural networks to optimize dynamical decoupling for quantum memory," *Physical Review A*, vol. 95, no. 1, pp. 012335, 2017.

[46] Mateusz Ostaszewski, JA Miszczak, Leonardo Banchi, and Przemyslaw Sadowski, "Approximation of quantum control correction scheme using deep neural networks," *Quantum Information Processing*, vol. 18, no. 5, pp. 126, 2019.

[47] Leonardo Banchi, Edward Grant, Andrea Rocchetto, and Simone Severini, "Modelling non-Markovian quantum processes with recurrent neural networks," *New Journal of Physics*, vol. 20, no. 12, pp. 123030, 2018.

[48] Sepp Hochreiter and Jürgen Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[49] Ilya Sutskever, Oriol Vinyals, and Quoc V Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, 2014, pp. 3104–3112.

[50] Shuohang Wang and Jing Jiang, "Machine comprehension using match-lstm and answer pointer," *arXiv preprint arXiv:1608.07905*, 2016.

[51] Kosuke Mitarai, Makoto Negoro, Masahiro Kitagawa, and Keisuke Fujii, "Quantum circuit learning," *Physical Review A*, vol. 98, no. 3, pp. 032309, 2018.

[52] Ville Bergholm, Josh Izaac, Maria Schuld, Christian Gogolin, Carsten Blank, Keri McKiernan, and Nathan Killoran, "Pennylane: Automatic differentiation of hybrid quantum-classical computations," *arXiv preprint arXiv:1811.04968*, 2018.

[53] Héctor Abraham et al., "Qiskit: An open-source framework for quantum computing," 2019.

[54] Tommaso Tufarelli, Francesco Ciccarello, and M. S. Kim, "Dynamics of spontaneous emission in a single-end photonic waveguide," *Phys. Rev. A*, vol. 87, pp. 013820, Jan 2013.

[55] T. Tufarelli, M. S. Kim, and F. Ciccarello, "Non-Markovianity of a quantum emitter in front of a mirror," *Phys. Rev. A*, vol. 90, no. 1, pp. 012113, July 2014.

[56] Yao-Lung L Fang, Francesco Ciccarello, and Harold U Baranger, "Non-Markovian dynamics of a qubit due to single-photon scattering in a waveguide," *New J. Phys.*, vol. 20, no. 4, pp. 043035, Apr. 2018.

[57] H. Dong, Z.R. Gong, H. Ian, Lan Zhou, and C.P. Sun, "Intrinsic cavity QED and emergent quasinormal modes for a single photon," *Phys. Rev. A*, vol. 79, pp. 063847, Jun 2009.

[58] Giuseppe Calajo, Yao-Lung L Fang, Harold U Baranger, and Francesco Ciccarello, "Exciting a Bound State in the Continuum through Multiphoton Scattering Plus Delayed Quantum Feedback," *Phys. Rev. Lett.*, vol. 122, no. 7, pp. 073601, Feb. 2019.

[59] S. M. Girvin, M. H. Devoret, and R. J. Schoelkopf, "Circuit qed and engineering charge-based superconducting qubits," *Physica Scripta*, vol. T137, pp. 014012, 2009.

[60] H. J. Kimble, "The quantum internet," *Nature*, vol. 453, pp. 1023, 2008.