Quantum Long Short-Term Memory

Samuel Yen-Chi Chen
Wells Fargo
46 PUBLICATIONS **620** CITATIONS
SEE PROFILE

Yao-Lung Leo Fang
NVIDIA
24 PUBLICATIONS **634** CITATIONS
SEE PROFILE

Quantum Long Short-Term Memory

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

Computational Science Initiative, Brookhaven National Laboratory

(Dated: September 4, 2020)

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 on noisy intermediate-scale quantum (NISQ) devices.

[*] ychen@bnl.gov

[†] sjyoo@bnl.gov

[‡] leofang@bnl.gov

I. INTRODUCTION

Recently, machine learning (ML), in particular deep learning (DL), has found tremendous success in computer vision [1–3], natural language processing [4], and mastering the game of Go [5]. In addition to commercial applications, DL-based methods have also been employed in solving important physics problems, such as quantum many-body physics [6–8], phase transitions [9], quantum control [10, 11], and quantum error correction [12, 13]. 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 the evolution of superconducting qubits [11] and the control of quantum memory [14]. These studies therefore demonstrate the possibilities of using ML to model the time evolution of quantum states.

In the meantime, quantum computers, both general- and special- purpose ones, are introduced to the general public by several technology companies such as IBM [15], Google [16], and D-Wave [17]. 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 [18] due to the lack of quantum error correction [19, 20]. 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 [21] 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 [21, 22] 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 [23, 24], generative adversarial learning [25] and deep reinforcement learning [26]. However, the problem of learning sequential data, to our best knowledge, has not been investigated in the quantum domain.

In this work, we address the issue of learning sequential, or temporal, data with quantum machine learning (QML) [27–29]. 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.

We envision our QLSTM to be applicable to various temporal scientific challenges, one of which is open quantum systems (OQS) from modern physics [30, 31]. OQS aims to explore the consequences of isolated quantum systems interacting with their surrounding environment. In many OQSs, including those with strong system-environment couplings or quantum feedbacks, the memory effect is not negligible [32], meaning the system's quantum state depends on its past trajectory if one chooses to ignore (or is unable to keep track of) the environment's dynamics. Such *non-Markovian* effects may reveal themselves through observable phenomena such as non-exponential decays and population death and revival [32, 33]. Strongly non-Markovian systems, including those with *delayed* quantum feedback and control [34–37], have been shown to exhibit interesting quantum behaviors that can be harnessed for designing novel quantum information processing devices [34, 35, 38], and their temporal behaviors are an ideal testbed for QLSTM, whose internal memory may capture well the memory effects in non-Markovian physics [39].

This paper is organized as follows. First, in Section II we briefly review the key ingredients in our work, RNN and LSTM, from the aspect of classical ML. Then, in Section III we introduce VQCs, the building block of the proposed framework. Next, we discuss our QLSTM architecture and its detailed mechanism in Section IV. In Section V we investigate through simulations the QLSTM capability for several different kinds of temporal data, including two from OQS problems, and compare with the outcomes of their classical counterparts. Finally, we conclude in Section VI.

II. 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.

A. Recurrent neural network

The RNNs (Figure 1) 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 [40, 41]. 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 [11, 14, 42, 43]. 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.

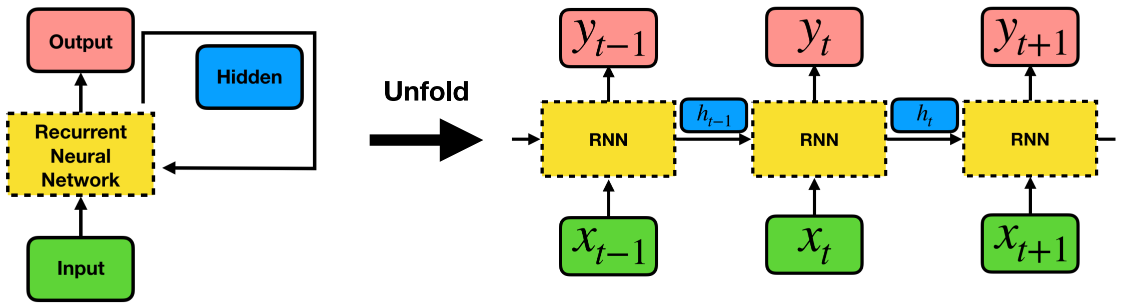

FIG. 1: A schematic for recurrent neural networks (RNN). For each time step t, the RNN takes an input value x_t, output a value o_t and a *hidden* value h_t which will be fed into itself at step $t+1$, enabling such architectures to learn temporal dependency. By unfolding the RNN in time, each time step 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 Figure 1 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 Figure 1 on the right).

B. Long short-term memory

The LSTM [44] 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 [45] 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 [11, 14].

The information flow in a classical LSTM cell (Figure 2) is

$$f_t = \sigma\left(W_f \cdot v_t + b_f\right), \tag{1a}$$

$$i_t = \sigma\left(W_i \cdot v_t + b_i\right), \tag{1b}$$

$$\tilde{C}_t = \tanh\left(W_C \cdot v_t + b_C\right), \tag{1c}$$

$$c_t = f_t * c_{t-1} + i_t * \tilde{C}_t, \tag{1d}$$

$$o_t = \sigma\left(W_o \cdot v_t + b_o\right), \tag{1e}$$

$$h_t = o_t * \tanh\left(c_t\right), \tag{1f}$$

where σ denotes the sigmoid function, $\{W_n\}$ are classical neural networks ($n = f, i, C, o$),

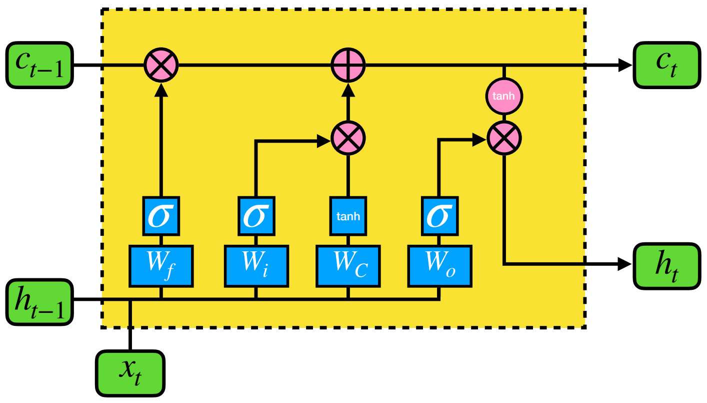

FIG. 2: A schematic for a classical long short-term memory (LSTM) cell. See the main text and Eq. (1) for the meaning of each component.

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. This will be contrasted with QLSTM to be introduced below.

III. VARIATIONAL QUANTUM CIRCUITS

VQCs are a kind of quantum circuits that have *tunable* parameters subject to iterative optimizations, see Figure 3 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.

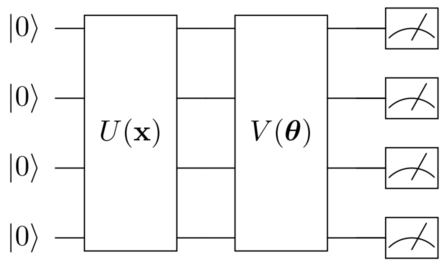

FIG. 3: Generic architecture for variational quantum circuits (VQC). $U(\mathbf{x})$ is the quantum routine for encoding the (classical) input data $\mathbf{x}$ and $V(\boldsymbol{\theta})$ is the variational circuit block with tunable parameters $\boldsymbol{\theta}$. A quantum measurement over some or all of the qubits follows.

Previous results have shown that such circuits are robust against quantum noise [46–48] and therefore suitable for the NISQ devices. VQCs have been successfully applied to function approximation [21], classification [23, 24, 49, 50], generative modeling [25], deep reinforcement learning [26], and transfer learning [51]. Furthermore, it has been pointed out that the VQCs are more expressive than classical neural networks [17, 22, 52] 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* [53], 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.

IV. 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 Figure 5 for a schematic of the proposed QLSTM architecture. The mathematical construction is given in Equation 5, which we discuss in detail below.

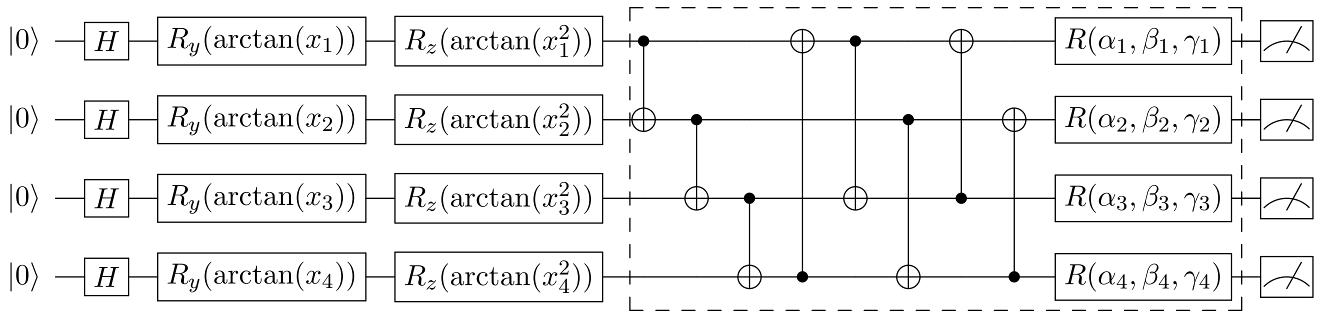

FIG. 4: Generic VQC architecture for QLSTM. It consists of three layers: the data encoding layer (with the H, R_y, and R_z gates), the variational layer (dashed box), and the quantum measurement layer. Note that the number of qubits and the number of measurements can be adjusted to fit the problem of 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.

A. Circuit Blocks

Here we describe the building blocks for our proposed QLSTM framework. The VQC used here is presented in Figure 4. Every circuit blocks used in a QLSTM cell consist of three layers: the data encoding layer, variational layer, and quantum measurement layer.

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:

$$|\psi\rangle = \sum_{(q_1,q_2,\cdots,q_N)\in\{0,1\}} c_{q_1,q_2,\cdots,q_N} |q_1\rangle \otimes |q_2\rangle \otimes \cdots \otimes |q_N\rangle , \tag{2}$$

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:

$$\sum_{(q_1,\cdots,q_N)\in\{0,1\}} ||c_{q_1,\cdots,q_N}||^2 = 1. \tag{3}$$

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. [21], 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,

$$\begin{aligned}(H|0\rangle)^{\otimes N} &= \frac{1}{\sqrt{2^N}} (|0\rangle + |1\rangle)^{\otimes N} \\ &= \frac{1}{\sqrt{2^N}} (|0\rangle \otimes \cdots \otimes |0\rangle + \cdots + |1\rangle \otimes \cdots \otimes |1\rangle) \\ &\equiv \frac{1}{\sqrt{2^N}} \sum_{i=0}^{2^N-1} |i\rangle ,\end{aligned} \tag{4}$$

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. [21], 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. (4) 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.

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 Figure4). 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.

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 [54] and IBM Qiskit [55], 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.

B. Stack All the Blocks

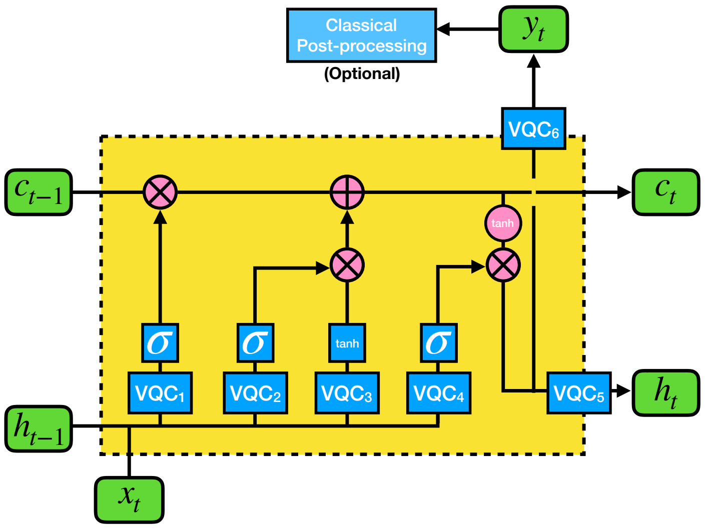

FIG. 5: The proposed quantum long short-term memory (QLSTM) architecture. Each VQC box is of the form as detailed in Figure 4. The σ and tanh blocks represent the sigmoid and the hyperbolic tangent activation function, respectively. x_t is the input at time t, h_t is for the hidden state, c_t is for the cell state, and y_t is the output. $\otimes$ and $\oplus$ represents element-wise multiplication and addition, respectively.

To construct the basic unit of the proposed QLSTM architecture, a QLSTM cell, we stack the aforementioned VQC blocks together. In Figure 5, each of the VQC_i block is described in the previous section (see also Figure 4). 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. (1) for classical LSTM]

$$f_t = \sigma\left(VQC_1(v_t)\right) \tag{5a}$$

$$i_t = \sigma\left(VQC_2(v_t)\right) \tag{5b}$$

$$\tilde{C}_t = \tanh\left(VQC_3(v_t)\right) \tag{5c}$$

$$c_t = f_t * c_{t-1} + i_t * \tilde{C}_t \tag{5d}$$

$$o_t = \sigma\left(VQC_4(v_t)\right) \tag{5e}$$

$$h_t = VQC_5(o_t * \tanh(c_t)) \tag{5f}$$

$$y_t = VQC_6(o_t * \tanh(c_t)), \tag{5g}$$

which can be grouped into three layers for their purposes:

- **Forget Block** [Eq. (5a)]: 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. (5b)-(5d)]: 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. (5e)-(5g)]: 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.

C. Optimization Procedure

In the optimization procedure, we employ the *parameter-shift* method [54, 56] to derive the analytical gradient of the quantum circuits. For example, given the expectation value of an observable $\hat{B}$

$$f\left(x;\theta_i\right) = \left\langle 0 \left| U_0^\dagger(x) U_i^\dagger\left(\theta_i\right) \hat{B} U_i\left(\theta_i\right) U_0(x) \right| 0 \right\rangle = \left\langle x \left| U_i^\dagger\left(\theta_i\right) \hat{B} U_i\left(\theta_i\right) \right| x \right\rangle, \tag{6}$$

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 [21] that the gradient of f with respect to the parameter θ_i is

$$\nabla_{\theta_i} f(x;\theta_i) = \frac{1}{2}\left[f\left(x;\theta_i + \frac{\pi}{2}\right) - f\left(x;\theta_i - \frac{\pi}{2}\right)\right]. \tag{7}$$

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.

V. 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 [57] 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 (Figure 5), 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 $6 \times 4 \times 2 \times 3 + 2 = 146$. We use the same (Q)LSTM architecture throughout this section. Finally, we use PennyLane [54, 58] and Qulacs [59] 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 [60], a variant of gradient descent methods with an adaptive learning rate that updates the parameters θ as:

$$E\left[g^2\right]_t = \alpha E\left[g^2\right]_{t-1} + (1-\alpha) g_t^2, \tag{8a}$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{E\left[g^2\right]_t + \epsilon}} g_t, \tag{8b}$$

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 $\epsilon = 10^{-8}$.

A. 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:

$$y = \sin(x) \tag{9}$$

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 Figure 6. 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 I), 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 6).

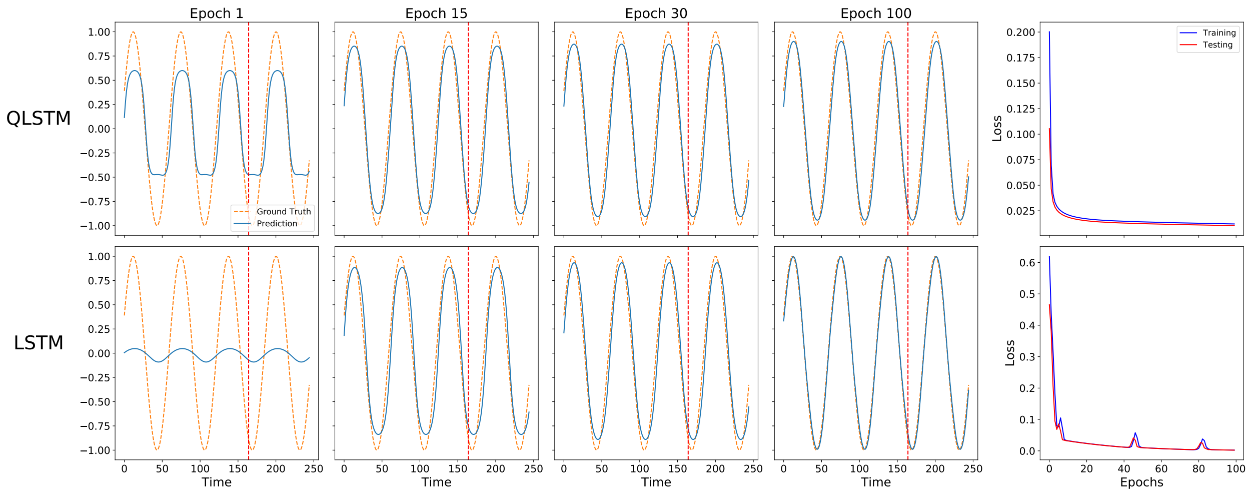

FIG. 6: Learning the sine function. QLSTM already learns the essence of $\sin(x)$ by Epoch 1, and has no peculiar bumps in the loss function. The orange dashed line represents the ground truth $\sin(x)$ [that we train the (Q)LSTM to learn] while the blue solid line is the output from the (Q)LSTM. The 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 I: The comparison of loss values at Epoch 15 for the sine function experiment.

B. Physical Dynamics

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

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,

$$\frac{\mathrm{d}^2x}{\mathrm{d}t^2} + 2\zeta\omega_0\frac{\mathrm{d}x}{\mathrm{d}t} + \omega_0^2 x = 0, \tag{10}$$

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,

$$\frac{d^2\theta}{dt^2} + \frac{b}{m}\frac{d\theta}{dt} + \frac{g}{L}\sin\theta = 0 \tag{11}$$

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 Figure 7. 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.

2. Bessel functions

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

$$x^2\frac{d^2y}{dx^2} + x\frac{dy}{dx} + \left(x^2 - \alpha^2\right)y = 0, \tag{12}$$

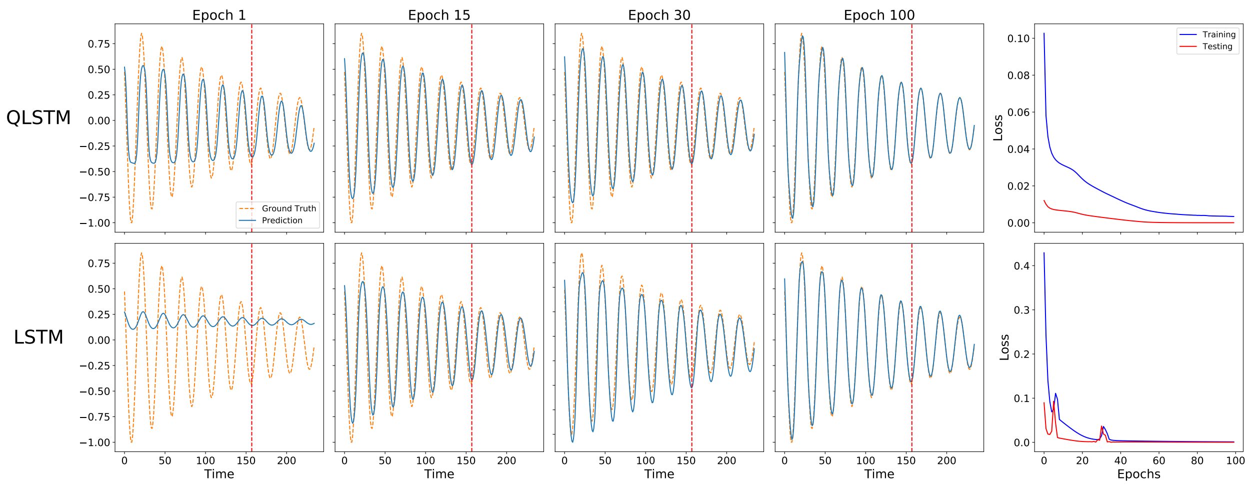

FIG. 7: Learning damped oscillations. The QLSTM learns faster and predicts more accurately than the classical LSTM with a fixed number of epochs. The orange dashed line represents the ground truth $\dot{\theta}$ [that we train the (Q)LSTM to learn] while the blue solid line is the output from the (Q)LSTM. 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 II: The comparison of loss values at Epoch 15 for the damped oscillator experiment.

to which the solution is

$$J_\alpha(x) = \sum_{m=0}^{\infty} \frac{(-1)^m}{m!\Gamma(m+\alpha+1)} \left(\frac{x}{2}\right)^{2m+\alpha}, \tag{13}$$

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 Figure 8. 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.

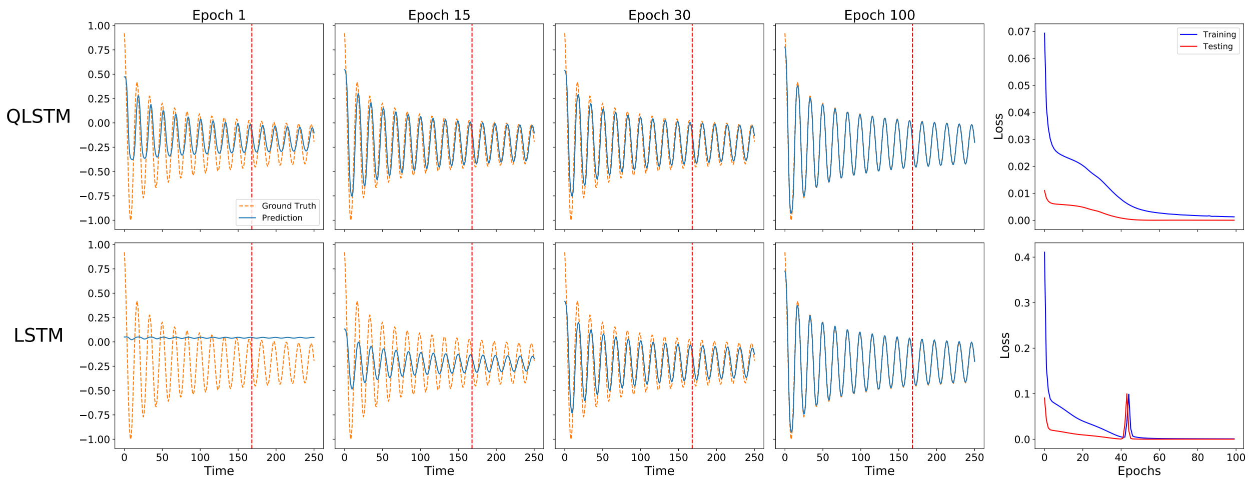

FIG. 8: Learning the Bessel function of order 2 (J_2). QLSTM's performance in prediction and convergence is even better than LSTM's with a slightly more complicated input (a non-exponential decay) compared to the previous cases. The orange dashed line represents the ground truth J_2 [that we train the (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 III: The comparison of loss values at Epoch 15 for the Bessel function J_2 experiment.

3. Delayed Quantum Control

Next, 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 [61–63], 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 [37, 61, 64]. 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 [61]. In Figure 9 we learn this temporal dependence using (Q)LSTM.

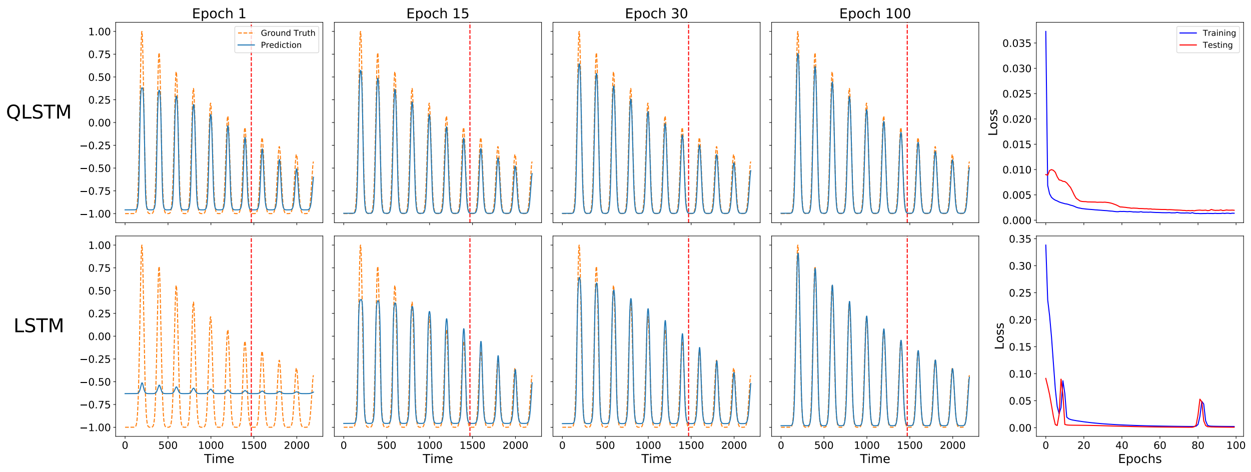

FIG. 9: Learning the dynamics with delayed quantum feedback. The QLSTM fits the local minima better than the LSTM does. The orange dashed line represents the ground truth [that we train the (Q)LSTM to learn] while the blue solid line is the output from the (Q)LSTM. The vertical red dashed line separates 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 [61], and the result is shown in Figure 9. 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 IV).

4. Population Inversion

Finally, we consider a textbook OQS problem: a simple cavity quantum electrodynamics (CQED) system [65–67], 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 quan-

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

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

tum computing and quantum information science, ranging from superconducting quantum computers [68] to optical quantum networks [69], due to its conceptual simplicity and yet high tunability and controllability.

By preparing the cavity in a coherent state

$$|\alpha\rangle = \exp\left(-|\alpha|^2/2\right) \sum_{n=0}^{\infty} \frac{\alpha^n}{\sqrt{n!}} |n\rangle \tag{14}$$

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 [70], 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

$$D(t) = p_g(t) - p_e(t) = \sum_{n=0}^{\infty} e^{-|\alpha|^2} \frac{|\alpha|^{2n}}{n!} \cos\left(2g\sqrt{n+1}t\right), \tag{15}$$

where g is the qubit-cavity coupling.

In Figure 10 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 V).

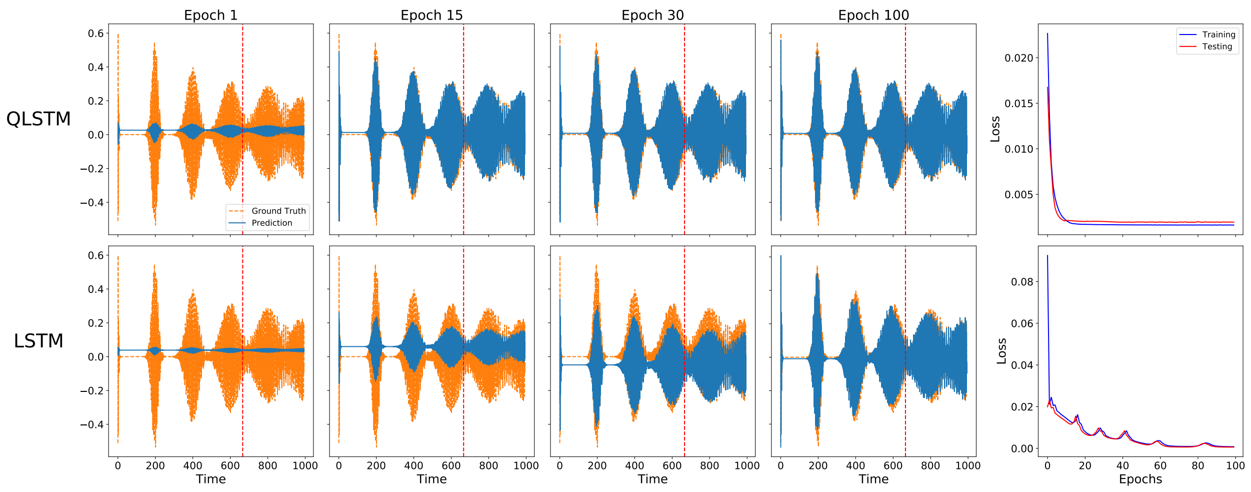

FIG. 10: Learning the population inversion. The QLSTM predicts better than the LSTM, in particular when the populations in the ground and excited states are balanced ($D = 0$). The orange dashed line represents the ground truth $D(t)$ [that we train the (Q)LSTM to learn] while the blue solid line is 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 V: The comparison of loss values at Epoch 15 for the population inversion experiment.

VI. 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 (see the comparisons for losses above, in particular the training losses). 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 scientific problems.

While it is impractical to run large-scale time-dependent data modeling due to the performance limitation in the quantum simulator software combined with the (classical) ML training framework, we emphasize that our proposed framework is rather general. For example, the VQCs in this work can have different gate sequences, more qubits, and/or more variational parameters that could potentially lead to better learning capability and higher expressive power.

Throughout this work, we use a predefined state preparation method (H–$R_y(\theta_{i,1})$–$R_z(\theta_{i,2})$, see Figure 4) to encode classical data into quantum states. However, the data encoding method can change. For example, it is possible to use *amplitude encoding* to encode the input vector, which in theory can provide more quantum advantage on the parameter saving [27].

It remains a significant challenge, as of today, to conduct the *training* phase on an actual NISQ device due to the excessive number of circuit evaluations, which depends on the number of circuits m, the number of parameters n, and the dataset size s. In this work we have $m = 6$, $n = 146$, and s ranges from 200 to 2000 depending on the experiments, so the number of quantum circuit evaluations *per epoch* grows at least as $\mathcal{O}(nms)$ in the training phase based on the parameter-shift gradient calculation [Eq. (7)]. However, it could be possible to perform the *inference* phase on a NISQ device with pre-trained variational parameters, which scales as $\mathcal{O}(ms')$ ($s' < s$ is the dataset size used for predicting the sequence).

Finally, we have assumed a perfect quantum computer in our numerical simulations — no noise (loss, decoherence, etc), precise control, and fully error-corrected — and the robustness of our QML framework against quantum noise is a very interesting subject, allowing evaluating the capability of NISQ devices. We leave these open questions and challenges for future work.

ACKNOWLEDGMENTS

We thank Meifeng Lin, Xin Zhang, Harold Baranger, Predrag Krstic, Hsi-Sheng Goan and Miles Stoudenmire for discussions. This work is supported by the U.S. Department of Energy, Office of Science, Office of High Energy Physics program under Award Number

DE-SC-0012704 and the Brookhaven National Laboratory LDRD #20-024.

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

[2] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.

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

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

[5] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, pp. 484–489, 1 2016.

[6] A. Borin and D. A. Abanin, "Approximating power of machine-learning ansatz for quantum many-body states," *arXiv preprint arXiv:1901.08615*, 2019.

[7] G. Carleo, K. Choo, D. Hofmann, J. E. Smith, T. Westerhout, F. Alet, E. J. Davis, S. Efthymiou, I. Glasser, S.-H. Lin, M. Mauri, G. Mazzola, C. B. Mendl, E. van Nieuwenburg, O. OReilly, H. Thveniaut, G. Torlai, F. Vicentini, and A. Wietek, "Netket: A machine learning toolkit for many-body quantum systems," *SoftwareX*, vol. 10, p. 100311, 2019.

[8] G. Carleo, I. Cirac, K. Cranmer, L. Daudet, M. Schuld, N. Tishby, L. Vogt-Maranto, and L. Zdeborová, "Machine learning and the physical sciences," *Rev. Mod. Phys.*, vol. 91, p. 045002, Dec 2019.

[9] A. Canabarro, F. F. Fanchini, A. L. Malvezzi, R. Pereira, and R. Chaves, "Unveiling phase transitions with machine learning," *Phys. Rev. B*, vol. 100, p. 045129, Jul 2019.

[10] Z. An and D. L. Zhou, "Deep reinforcement learning for quantum gate control," *EPL (Europhysics Letters)*, vol. 126, p. 60002, jul 2019.

[11] E. Flurin, L. S. Martin, S. Hacohen-Gourgy, and I. Siddiqi, “Using a recurrent neural network to reconstruct quantum dynamics of a superconducting qubit from physical observations,” *Physical Review X*, vol. 10, no. 1, p. 011006, 2020.

[12] P. Andreasson, J. Johansson, S. Liljestrand, and M. Granath, “Quantum error correction for the toric code using deep reinforcement learning,” *Quantum*, vol. 3, p. 183, Sept. 2019.

[13] H. P. Nautrup, N. Delfosse, V. Dunjko, H. J. Briegel, and N. Friis, “Optimizing quantum error correction codes with reinforcement learning,” *arXiv preprint arXiv:1812.08451*, 2018.

[14] M. August and X. Ni, “Using recurrent neural networks to optimize dynamical decoupling for quantum memory,” *Physical Review A*, vol. 95, no. 1, p. 012335, 2017.

[15] A. Cross, “The ibm q experience and qiskit open-source quantum computing software,” in *APS Meeting Abstracts*, 2018.

[16] F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin, R. Barends, R. Biswas, S. Boixo, F. G. Brandao, D. A. Buell, *et al.*, “Quantum supremacy using a programmable superconducting processor,” *Nature*, vol. 574, no. 7779, pp. 505–510, 2019.

[17] T. Lanting, A. J. Przybysz, A. Y. Smirnov, F. M. Spedalieri, M. H. Amin, A. J. Berkley, R. Harris, F. Altomare, S. Boixo, P. Bunyk, *et al.*, “Entanglement in a quantum annealing processor,” *Physical Review X*, vol. 4, no. 2, p. 021041, 2014.

[18] J. Preskill, “Quantum computing in the nisq era and beyond,” *Quantum*, vol. 2, p. 79, 2018.

[19] D. Gottesman, “Stabilizer codes and quantum error correction,” *arXiv preprint quant-ph/9705052*, 1997.

[20] D. Gottesman, “Theory of fault-tolerant quantum computation,” *Physical Review A*, vol. 57, no. 1, p. 127, 1998.

[21] K. Mitarai, M. Negoro, M. Kitagawa, and K. Fujii, “Quantum circuit learning,” *Physical Review A*, vol. 98, no. 3, p. 032309, 2018.

[22] Y. Du, M.-H. Hsieh, T. Liu, and D. Tao, “The expressive power of parameterized quantum circuits,” *arXiv preprint arXiv:1810.11922*, 2018.

[23] M. Schuld, A. Bocharov, K. Svore, and N. Wiebe, “Circuit-centric quantum classifiers,” *arXiv preprint arXiv:1804.00633*, 2018.

[24] V. Havlíček, A. D. Córcoles, K. Temme, A. W. Harrow, A. Kandala, J. M. Chow, and J. M. Gambetta, “Supervised learning with quantum-enhanced feature spaces,” *Nature*, vol. 567, no. 7747, pp. 209–212, 2019.

[25] P.-L. Dallaire-Demers and N. Killoran, “Quantum generative adversarial networks,” *Physical Review A*, vol. 98, no. 1, p. 012324, 2018.

[26] S. Y.-C. Chen, C.-H. H. Yang, J. Qi, P.-Y. Chen, X. Ma, and H.-S. Goan, “Variational quantum circuits for deep reinforcement learning,” *IEEE Access*, vol. 8, pp. 141007–141024, 2020.

[27] M. Schuld and F. Petruccione, *Supervised learning with quantum computers*, vol. 17. Springer, 2018.

[28] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, “Quantum machine learning,” *Nature*, vol. 549, no. 7671, pp. 195–202, 2017.

[29] V. Dunjko and H. J. Briegel, “Machine learning & artificial intelligence in the quantum domain: a review of recent progress,” *Reports on Progress in Physics*, vol. 81, no. 7, p. 074001, 2018.

[30] H.-P. Breuer and F. Petruccione, *The Theory of Open Quantum Systems.* Oxford, UK: Oxford University Press, 2002.

[31] Ángel Rivas and S. F. Huelga, *Open Quantum Systems.* Springer Berlin Heidelberg, 2012.

[32] I. de Vega and D. Alonso, “Dynamics of non-Markovian open quantum systems,” *Rev. Mod. Phys.*, vol. 89, p. 015001, Jan. 2017.

[33] H.-P. Breuer, E.-M. Laine, J. Piilo, and B. Vacchini, “Colloquium: Non-Markovian dynamics in open quantum systems,” *Rev. Mod. Phys.*, vol. 88, p. 021002, Apr. 2016.

[34] A. L. Grimsmo, “Time-Delayed Quantum Feedback Control,” *Phys. Rev. Lett.*, vol. 115, p. 060402, Aug. 2015.

[35] H. Pichler and P. Zoller, “Photonic Circuits with Time Delays and Quantum Feedback,” *Phys. Rev. Lett.*, vol. 116, p. 093601, Mar. 2016.

[36] S. J. Whalen, A. L. Grimsmo, and H. J. Carmichael, “Open quantum systems with delayed coherent feedback,” *Quantum Science and Technology*, vol. 2, no. 4, p. 044008, 2017.

[37] G. Calajo, Y.-L. L. Fang, H. U. Baranger, and F. Ciccarello, “Exciting a Bound State in the Continuum through Multiphoton Scattering Plus Delayed Quantum Feedback,” *Phys. Rev. Lett.*, vol. 122, p. 073601, Feb. 2019.

[38] H. Pichler, S. Choi, P. Zoller, and M. D. Lukin, “Universal photonic quantum computation via time-delayed feedback,” *Proceedings of the National Academy of Sciences*, vol. 114, no. 43, pp. 11362–11367, 2017.

[39] B. Bakker, “Reinforcement learning with long short-term memory,” in *Advances in neural*

information processing systems, pp. 1475–1482, 2002.

[40] G. Lai, Z. Dai, Y. Yang, and S. Yoo, "Re-examination of the role of latent variables in sequence modeling," in *Advances in Neural Information Processing Systems*, pp. 7812–7822, 2019.

[41] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning.* MIT press, 2016.

[42] M. Ostaszewski, J. Miszczak, L. Banchi, and P. Sadowski, "Approximation of quantum control correction scheme using deep neural networks," *Quantum Information Processing*, vol. 18, no. 5, p. 126, 2019.

[43] L. Banchi, E. Grant, A. Rocchetto, and S. Severini, "Modelling non-Markovian quantum processes with recurrent neural networks," *New Journal of Physics*, vol. 20, no. 12, p. 123030, 2018.

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

[45] S. Wang and J. Jiang, "Machine comprehension using match-lstm and answer pointer," *arXiv preprint arXiv:1608.07905*, 2016.

[46] A. Kandala, A. Mezzacapo, K. Temme, M. Takita, M. Brink, J. M. Chow, and J. M. Gambetta, "Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets," *Nature*, vol. 549, no. 7671, pp. 242–246, 2017.

[47] E. Farhi, J. Goldstone, and S. Gutmann, "A quantum approximate optimization algorithm," *arXiv preprint arXiv:1411.4028*, 2014.

[48] J. R. McClean, J. Romero, R. Babbush, and A. Aspuru-Guzik, "The theory of variational hybrid quantum-classical algorithms," *New Journal of Physics*, vol. 18, no. 2, p. 023023, 2016.

[49] E. Farhi and H. Neven, "Classification with quantum neural networks on near term processors," *arXiv preprint arXiv:1802.06002*, 2018.

[50] M. Benedetti, E. Lloyd, S. Sack, and M. Fiorentini, "Parameterized quantum circuits as machine learning models," *Quantum Science and Technology*, vol. 4, no. 4, p. 043001, 2019.

[51] A. Mari, T. R. Bromley, J. Izaac, M. Schuld, and N. Killoran, "Transfer learning in hybrid classical-quantum neural networks," *arXiv preprint arXiv:1912.08278*, 2019.

[52] S. Sim, P. D. Johnson, and A. Aspuru-Guzik, "Expressibility and entangling capability of parameterized quantum circuits for hybrid quantum-classical algorithms," *Advanced Quantum Technologies*, vol. 2, no. 12, p. 1900070, 2019.

[53] K. Hornik, M. Stinchcombe, H. White, *et al.*, "Multilayer feedforward networks are universal

approximators.," *Neural networks*, vol. 2, no. 5, pp. 359–366, 1989.

[54] V. Bergholm, J. Izaac, M. Schuld, C. Gogolin, C. Blank, K. McKiernan, and N. Killoran, "Pennylane: Automatic differentiation of hybrid quantum-classical computations," *arXiv preprint arXiv:1811.04968*, 2018.

[55] H. Abraham *et al.*, "Qiskit: An open-source framework for quantum computing," 2019.

[56] M. Schuld, V. Bergholm, C. Gogolin, J. Izaac, and N. Killoran, "Evaluating analytic gradients on quantum hardware," *Physical Review A*, vol. 99, no. 3, p. 032331, 2019.

[57] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. 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.

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

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

[60] 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.

[61] T. Tufarelli, F. Ciccarello, and M. S. Kim, "Dynamics of spontaneous emission in a single-end photonic waveguide," *Phys. Rev. A*, vol. 87, p. 013820, Jan 2013.

[62] T. Tufarelli, M. S. Kim, and F. Ciccarello, "Non-Markovianity of a quantum emitter in front of a mirror," *Phys. Rev. A*, vol. 90, p. 012113, July 2014.

[63] Y.-L. L. Fang, F. Ciccarello, and H. U. Baranger, "Non-Markovian dynamics of a qubit due to single-photon scattering in a waveguide," *New J. Phys.*, vol. 20, p. 043035, Apr. 2018.

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

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

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

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

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

[69] H. J. Kimble, “The quantum internet,” *Nature*, vol. 453, p. 1023, 2008.

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