

Learning temporal data with variational quantum recurrent neural network

Yuto Takaki,¹ Kosuke Mitarai,^{1, 2, 3,*} Makoto Negoro,^{2, 4} Keisuke Fujii,^{1, 2, 5} and Masahiro Kitagawa^{1, 2}

¹*Graduate School of Engineering Science, Osaka University,
1-3 Machikaneyama, Toyonaka, Osaka 560-8531, Japan*

²*Center for Quantum Information and Quantum Biology,
Institute for Open and Transdisciplinary Research Initiatives, Osaka University, Japan*

³*JST, PRESTO, 4-1-8 Honcho, Kawaguchi, Saitama 332-0012, Japan*

⁴*Institute for Quantum Life Science, National Institutes for Quantum and Radiological Science and Technology, Japan*

⁵*Center for Emergent Matter Science, RIKEN, Wako Saitama 351-0198, Japan*

(Dated: December 22, 2020)

We propose a method for learning temporal data using a parametrized quantum circuit. We use the circuit that has a similar structure as the recurrent neural network which is one of the standard approaches employed for this type of machine learning task. Some of the qubits in the circuit are utilized for memorizing past data, while others are measured and initialized at each time step for obtaining predictions and encoding a new input datum. The proposed approach utilizes the tensor product structure to get nonlinearity with respect to the inputs. Fully controllable, ensemble quantum systems such as an NMR quantum computer is a suitable choice of an experimental platform for this proposal. We demonstrate its capability with Simple numerical simulations, in which we test the proposed method for the task of predicting cosine and triangular waves and quantum spin dynamics. Finally, we analyze the dependency of its performance on the interaction strength among the qubits in numerical simulation and find that there is an appropriate range of the strength. This work provides a way to exploit complex quantum dynamics for learning temporal data.

I. INTRODUCTION

Quantum machine learning [1] is gaining attention in the hope of speeding up machine learning tasks with quantum computers. One direction is to develop quantum machine learning algorithms with proven speedups over classical approaches [2–5]. Another direction, which is becoming popular recently, is to construct heuristic algorithms using parametrized quantum circuits. Among such, a popular idea is to use an exponentially large Hilbert space of a quantum system for storing features of dataset [6–9]. In this approach, each datum is first encoded in a quantum state, and then we employ a parametrized quantum circuit to extract features that are not classically tractable.

Here, we specifically consider the extension of the latter approaches for the learning of temporal data. In practice, such machine learning tasks frequently appear in various fields such as natural language processing, speech recognition, and stock price prediction. A previous idea of using a quantum system for the time-series analysis can be found in Ref. [10], where the authors proposed quantum reservoir computing (QRC). QRC employs complex quantum dynamics itself as a computational reservoir. It learns to perform temporal tasks by optimizing the readout from the reservoir while the quantum system is left unchanged. This is in contrast to the approach taken in this work, that is, we directly tune the dynamics of the quantum system in use.

In this work, we propose a type of parametrized quantum circuit for learning temporal data in analogy with the recurrent neural network (RNN) which is a popular machine learning model on a classical computer for the task [11]. We call the proposed circuit quantum recurrent neural network (QRNN). In QRNN, we repeat the following three steps; encoding an input, applying a parametrized quantum circuit, and measuring a portion of qubits to obtain a prediction from the system. The circuit applied at the second step remains the same at every loop, introducing a recurrent structure. The parameters in the circuit are optimized with respect to a suitable cost function whose minimization leads to an accurate prediction of a given temporal dataset. This algorithm employs the tensor product structure of quantum systems to obtain the nonlinearity of the output with respect to input in a way similar to Ref. [7]. To use expectation values to obtain the output prediction while maintaining the recurrent structure, we assume the use of an ensemble quantum system such as NMR quantum computers which allows us to efficiently extract them. We also test the validity of the proposed method by numerically simulating the training and prediction processes. We believe this work opens up a way to exploit parametrized quantum circuits for time-series analysis.

The rest of this work is organized as follows. In Sec. II, we first review the classical recurrent neural network and existing machine learning algorithms with parametrized quantum circuits. Then, the proposed QRNN is described in detail in Sec. III. In Sec. IV, we present numerical simulations where we apply QRNN to some simple time-series prediction tasks and analyze results. Conclusion and outlook are given in Sec. V.

* mitarai@qc.ee.es.osaka-u.ac.jp

II. PRELIMINARY

A. Learning of temporal data

In a (supervised) temporal learning task, we are given with a input sequence $\{\mathbf{x}_t\}_{t=0}^{T-1}$ and a target teacher sequence $\{\mathbf{y}_t\}_{t=0}^{T-1}$ as a training dataset. Our task is to construct a model $\bar{\mathbf{y}}_k = f(\{\mathbf{x}_t\}_{t=0}^k)$ such that difference between \mathbf{y}_k and $f(\{\mathbf{x}_t\}_{t=0}^k)$ is small from the training dataset. A popular example is the time-series prediction task where $\mathbf{y}_t = \mathbf{x}_{t+1}$, which will be used for demonstration of the proposed method in Sec. IV. After constructing a model that is accurate enough, we can perform the prediction of \mathbf{y}_t .

B. Recurrent neural network

RNN [11] is a famous technique to analyze time-series data. There are a number of variants of RNN models such as Elman network, long short term memory (LSTM), and gated recurrent unit (GRU) [12–14]. Among those, we use a basic form of RNN shown in Fig. 1 (a) as a reference to construct its quantum version.

RNN consists of an input layer \mathbf{x}_t , hidden layer \mathbf{h}_t , and output layer $\bar{\mathbf{y}}_{t+1}$, which are real-valued vectors. When \mathbf{x}_t is injected into an input layer of RNN, the value of the hidden layer is updated using the value of \mathbf{x}_t and \mathbf{h}_{t-1} as,

$$\mathbf{h}_t = g_h(V_{\text{in}}\mathbf{x}_t + W\mathbf{h}_{t-1} + \mathbf{b}_{\text{in}}) \quad (1)$$

where V_{in} and W are matrices, \mathbf{b}_{in} is a bias vector, and $g_h(\cdot)$ is an activation function which element-wisely apply a nonlinear transformation to a vector. The output $\bar{\mathbf{y}}_{t+1}$ is computed as,

$$\bar{\mathbf{y}}_{t+1} = g_o(V_{\text{out}}\mathbf{h}_t + \mathbf{b}_{\text{out}}) \quad (2)$$

where V_{out} , \mathbf{b}_{out} , and $g_o(\cdot)$ are a matrix, a bias vector, and an activation function like g_h , respectively. W , V_{in} , \mathbf{b}_{in} , V_{out} , and \mathbf{b}_{out} are parameters to be optimized to minimize a cost function L which represents the difference between \mathbf{y}_{t+1} and $\bar{\mathbf{y}}_{t+1}$ such as the squared error $L = \sum_t \|\bar{\mathbf{y}}_t - \mathbf{y}_t\|^2$. The recurrent structure where the value of the hidden layer at time t is computed from \mathbf{h}_{t-1} allows the RNN to hold the information given in the past time steps [11].

Frequently, the training of RNN is performed with the unfolded expression as shown in Fig. 1 (b), which allows us to treat RNN in the same manner as feed-forward neural networks and to use standard techniques such as back-propagation to optimize the parameters W , V_{in} , \mathbf{b}_{in} , V_{out} , and \mathbf{b}_{out} . After the training using the dataset $\{\mathbf{x}_t\}_{t=0}^{T-1}$ and $\{\mathbf{y}_t\}_{t=0}^{T-1}$, we can proceed to the prediction of \mathbf{y}_T , \mathbf{y}_{T+1}, \dots by repeating the structure. However, for the time-series prediction task, the next input data \mathbf{x}_T itself is unknown and to be predicted. Therefore, the prediction has to be performed using the strategy given in Fig. 1

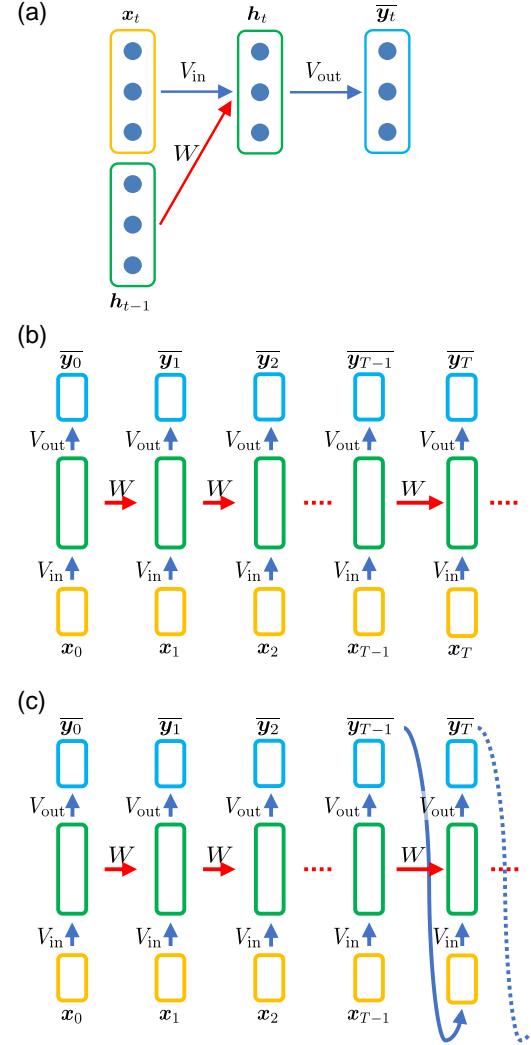


FIG. 1. (a) Schematic picture of a basic RNN. Blue circles represent elements of vectors \mathbf{x}_t , \mathbf{h}_t and $\bar{\mathbf{y}}_t$. The matrices V_{in} , V_{out} and W connecting layers are the parameters to be optimized during training of the model. The bias vectors \mathbf{b} and \mathbf{c} are abbreviated for simplicity. (b) Unfolded version of (a). (c) RNN for time-series prediction.

(c) where we feed its prediction $\bar{\mathbf{y}}_t$ as the input. This strategy is justified since we expect the trained network to output $\bar{\mathbf{y}}_t$ that is close to the true value \mathbf{x}_{t+1} . We follow the above structure closely to construct its quantum version in the following sections.

C. Machine learning with parametrized quantum circuit

Many researchers have developed a wide range of algorithms [9] using parametrized circuits, including quantum chemistry simulations [15], combinational optimization problems [16], and machine learning [6–8, 17]. Algorithms involving parametrized quantum circuits usually works in the following way. First, we apply a

parametrized quantum circuit $U(\boldsymbol{\theta})$ to some initialized state $|\psi_0\rangle$, where $\boldsymbol{\theta}$ is the parameter of the circuit. Then, we measure an expectation value of a specific observable, $\langle O(\boldsymbol{\theta}) \rangle$. Based on the value, we compute a cost function $L(\langle O(\boldsymbol{\theta}) \rangle)$ to be minimized by optimizing $\boldsymbol{\theta}$.

For machine learning tasks, we define $L(\langle O(\boldsymbol{\theta}) \rangle)$ such that the output $\langle O(\boldsymbol{\theta}) \rangle$ provides us appropriate predictions when it is minimized. Let us describe an example of supervised learning [6–8]. In supervised learning, we are provided with a training dataset consisting of input data $\{\mathbf{v}_i\}$ and corresponding teacher data $\{u_i\}$. Ideas introduced in Refs. [6–8] are to use a parametrized quantum circuit $U(\boldsymbol{\theta}, \mathbf{v}_i)$ which depends also on \mathbf{v}_i . This circuit encodes an input \mathbf{v}_i into a quantum state which results in the input-dependent output $\langle O(\boldsymbol{\theta}, \mathbf{v}_i) \rangle$. The cost function is defined as, for example, the mean squared error between $\{u_i\}$ and $\{\langle O(\boldsymbol{\theta}, \mathbf{v}_i) \rangle\}$. Minimization of such a cost function by optimizing the parameters $\boldsymbol{\theta}$ results in $\langle O(\boldsymbol{\theta}_{\text{opt}}, \mathbf{v}_i) \rangle$ that is close to u_i , where $\boldsymbol{\theta}_{\text{opt}}$ is the optimized parameter. Finally, we can use $\langle O(\boldsymbol{\theta}_{\text{opt}}, \mathbf{v}) \rangle$ for an unknown input \mathbf{v} to predict corresponding u . In the following sections, we extend this approach to the task of predicting time-series data by combining it with the ideas of RNN.

III. QUANTUM RECURRENT NEURAL NETWORK

In this section, we first describe the proposed algorithm and its theoretical capability. Then, a possible experimental setup for its realization and its relation with previous works are discussed.

A. Algorithm

Fig. 2 shows a schematic of QRNN algorithm using $n = n_A + n_B$ qubits. QRNN is composed of two groups of qubits called 'A' and 'B'. Groups A and B have n_A and n_B qubits respectively. The qubits in group A are never measured throughout the algorithm so that they hold past information. On the other hand, group B qubits are measured and initialized at each time step t to output the prediction and to take input to the system.

Each time step of QRNN consists of three parts, namely, encoding part, evolution part, and measurement part, which are schematically depicted in Fig. 2 (a). At the encoding part of time step t , we encode a training datum \mathbf{x}_t to the quantum state of qubits in group B by applying $U_{\text{in}}(\mathbf{x}_t)$ to the initialized state $|0\rangle^{\otimes n_B}$. Hereafter, we abbreviate $|0\rangle^{\otimes n_B}$ as $|0\rangle$ if there is no confusion. Note that group A already holds information about $\mathbf{x}_0, \dots, \mathbf{x}_{t-1}$ as a density matrix $\rho_{t-1}^A(\boldsymbol{\theta}, \mathbf{x}_0, \dots, \mathbf{x}_{t-1})$ resulted from previous steps. At the evolution part, we apply a parameterized unitary circuit $U(\boldsymbol{\theta})$ to the entire qubits in the system. $U(\boldsymbol{\theta})$ transfers the information injected in group B to group A by introducing the inter-

action among the qubits. We denote the reduced density matrix of A and B as ρ_t^A and ρ_t^B after the evolution, respectively. At the measurement part, we first measure expectation values of a set of commuting observables, $\{O_i\}$, of group B to obtain,

$$\langle O_i \rangle_t = \text{Tr}[\rho_t^B O_i]. \quad (3)$$

Then, we transform this expectation value to get the prediction $\bar{\mathbf{y}}_t$ of \mathbf{y}_t by some function g ; $\bar{\mathbf{y}}_t = g(\{\langle O_i \rangle_t\})$. g can, for example, be a linear combination of $\{\langle O_i \rangle_t\}$. Note that the transformation g can be chosen arbitrarily and optimized in general. Afterward, the qubits in group B are initialized to $|0\rangle$.

We repeat the three parts to obtain the predictions $\bar{\mathbf{y}}_0, \dots, \bar{\mathbf{y}}_{T-1}$ for $\mathbf{y}_0, \dots, \mathbf{y}_{T-1}$. After obtaining the predictions, we compute the cost function $L(\{\mathbf{y}_0, \dots, \mathbf{y}_{T-1}\}, \{\bar{\mathbf{y}}_0, \dots, \bar{\mathbf{y}}_{T-1}\})$ which represents the difference between the training data $\{\mathbf{y}_0, \dots, \mathbf{y}_{T-1}\}$ and prediction $\{\bar{\mathbf{y}}_0, \dots, \bar{\mathbf{y}}_{T-1}\}$ obtained by QRNN. The parameter $\boldsymbol{\theta}$ is optimized to minimize L . This optimization is performed by standard optimizers running on a classical computer. For example, we can utilize gradient-free optimization methods such as Nelder-Mead or simultaneous perturbation stochastic approximation (SPSA) algorithms. Another choice would be to use a gradient-based method such as gradient descent. The analytic gradient of the cost function can be obtained using the so-called parameter shift rule [7, 18, 19] by running QRNN for $O(T)$ times for each parameter, see Appendix for detail.

After the training, we expect the trained QRNN to be able to predict \mathbf{y}_t with $\bar{\mathbf{y}}_t$ for $t \geq T$ after running it through $t = 0$ to $T - 1$. For the time-series prediction task, we use its prediction $\bar{\mathbf{y}}_t$ as input to the system similarly to the ordinary RNN case, since we expect QRNN to output $\bar{\mathbf{y}}_t$ which is close to its true value \mathbf{x}_{t+1} .

B. Capability of QRNN

Here, we will discuss what types of function QRNN can model using a simple example. QRNN exploits the tensor product structure to gain the nonlinearity of the output with respect to the input in a way similar to Ref. [7].

For simplicity, here we assume that the input and teacher data are scalar. Suppose that we use a single qubit as group B, and take $U_{\text{in}}(x) = R_y(\arccos x)$. This gate applied to $|0\rangle$ encodes x to the Z-expectation value of the qubit, that is, at $t = 0$, the state after injecting the first input datum x_0 is,

$$\frac{1}{2}(I + \sqrt{1 - x_0^2}X + x_0Z) \otimes |0\rangle\langle 0|^{\otimes n_A}. \quad (4)$$

After the evolution $U(\boldsymbol{\theta})$, the density matrix of the system is,

$$\sum_{P \in \mathcal{P}_n} \left(c_{1P}(\boldsymbol{\theta})x_0 + c_{2P}(\boldsymbol{\theta})\sqrt{1 - x_0^2} + c_{3P}(\boldsymbol{\theta}) \right) P, \quad (5)$$

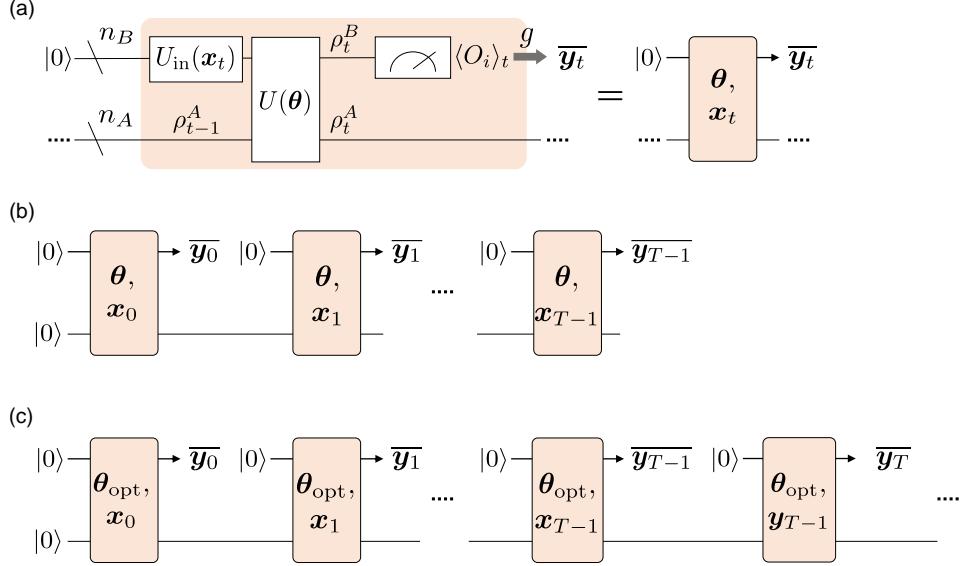


FIG. 2. Structure of QRNN. (a) Structure of QRNN for a single time step. (b) QRNN in the training phase. (c) QRNN in the prediction phase. Note that the obtained prediction $\bar{\mathbf{x}}_T$ is used as the input to the next time step.

where $\mathcal{P}_n = \{I, X, Y, Z\}^{\otimes n}$ and $c_{1P}(\boldsymbol{\theta})$, $c_{2P}(\boldsymbol{\theta})$, and $c_{3P}(\boldsymbol{\theta})$ are real coefficients. This means that an expectation value of any Pauli operator P can be written as a linear combination of $\{x_0, \sqrt{x_0 - 1}, 1\}$, and the output $\bar{\mathbf{y}}_0$ can also be written in terms of them.

In the next time step of QRNN, ρ_0^A , which is the reduced density matrix of Eq. (5) obtained by tracing out B , is generally in the form of,

$$\rho_0^A = \sum_{P \in \mathcal{P}_{n_A}} \left(c'_{1P}(\boldsymbol{\theta})x_0 + c'_{2P}(\boldsymbol{\theta})\sqrt{1 - x_0^2} + c'_{3P}(\boldsymbol{\theta}) \right) P, \quad (6)$$

with some coefficient $c'_{1P}(\boldsymbol{\theta})$, $c'_{2P}(\boldsymbol{\theta})$, and $c'_{3P}(\boldsymbol{\theta})$. Then, x_1 is injected into B by $U_{\text{in}}(\mathbf{x})$, and $U(\boldsymbol{\theta})$ is applied to the whole system. This results in the density matrix in the form of,

$$\sum_{P \in \mathcal{P}_{n_A}} \sum_i c''_{iP}(\boldsymbol{\theta}) \phi_i(x_0, x_1) P, \quad (7)$$

where $\{\phi_i(x_0, x_1)\} = \{x_0, x_1, \sqrt{1 - x_0^2}, \sqrt{1 - x_1^2}, x_0x_1, x_0\sqrt{1 - x_1^2}, x_1\sqrt{1 - x_0^2}, 1\}$, and c''_{iP} are coefficients. Note that the nonlinear functions such as x_0x_1 originate from the tensor product structure of the system. Therefore, we can conclude the output $\bar{\mathbf{y}}_1$ can be written in terms of a linear combination of $\phi_i(x_0, x_1)$ in a way similar to the above.

Repeating the above discussion, we can see that the output of QRNN can have highly nonlinear terms such as $\prod_{t=0}^T x_t$ thanks to the tensor product structure. Note that QRNN can “choose” which term to be kept in group A qubits by tuning $U(\boldsymbol{\theta})$. For example, QRNN can fully keep x_0 term in ρ_0^A if $U(\boldsymbol{\theta})$ transforms $X \otimes I^{\otimes n_A}$ to a local Pauli operator only acting on A . On the other hand,

if $U(\boldsymbol{\theta})$ transforms $X \otimes I^{\otimes n_A}$ to a nonlocal Pauli operator acting both on A and B , x_0 vanishes in ρ_0^A since the partial trace removes such a term. If $U(\boldsymbol{\theta})$ acts in an intermediate manner of the above two extreme cases, the magnitude of x_0 in ρ_0^A becomes smaller than the former case, that is, QRNN partially “forgets” x_0 . We believe that we can employ these insights to construct more efficient QRNNs in future research.

C. Experimental realization

As mentioned above, expectation values of an observable $\langle O \rangle$ are used as a new input in the prediction phase of QRNN, which requires us to obtain $\langle O \rangle$ in a single-shot manner. Ensemble quantum systems such as NMR quantum computers, where such measurements are possible, are desirable for this reason. While nuclear spins used as qubits in NMR are hard to initialize, we believe this problem can be resolved by e.g. a technique called dynamic nuclear polarization, which initializes nuclear spins by transferring the state of initialized electron spins [20–23].

On the other hand, the use of single quantum systems such as superconducting or ion trap qubits only allows us to extract bitstrings with single-shot measurements, which leads to inefficiency in the prediction phase for time-series prediction tasks. It requires us to run the whole QRNN to some time step t to obtain a prediction $\bar{\mathbf{y}}_t$, which must be obtained with some accuracy to proceed to the next time step since it is used as the input to the system for such tasks. However, we might be able to extend the framework of QRNN to use such single quantum systems; for example, they can be used for time-series prediction of binary data, where the predicted data itself is represented by bitstrings. We leave such an

extension for future work.

D. Relation between existing algorithms

The quantum reservoir computing (QRC) [10] has been proposed as a technique to tackle temporal learning tasks using complex quantum dynamics. QRC is a quantum-classical hybrid algorithm similar to QRNN from the viewpoint of the purpose; both of them are used to train and predict time-series data. The difference between QRNN and QRC is that while QRC updates readout transformation g to minimize the cost function, QRNN mainly tunes the quantum system by optimizing the circuit parameter θ . Subsequent theoretical [24–31] and experimental works [25, 32] have extended the QRC in various ways.

We also notice a similar title in Ref. [33]. This approach is based on quantum neurons presented in Ref. [34], and use sophisticated quantum circuits. This is in contrast to our approach, where the parametrized circuit $U(\theta)$ can be constructed in a hardware-efficient way, as we show in the next section by numerical simulations.

IV. NUMERICAL EXPERIMENTS

Here, we demonstrate and analyze the performance of the proposed QRNN by simulating its training and prediction phases for time-series prediction tasks. We use scalar input data $\{x_t\}$, and the task is to construct a QRNN that can output $\bar{y}_t \approx x_{t+1}$. Throughout the simulations presented in this section, we investigate a QRNN circuit using $n = 6$ qubits; groups A and B have $n_A = 3$ and $n_B = 3$ qubits, respectively. As the encoding gate, we use,

$$U_{\text{in}}(x_t) = R_y(\arccos x_t), \quad (8)$$

acting on each qubit in group B, following the ideas presented in Ref. [7]. Note that the input gate can be chosen differently; for example, we might be able to improve the performance of QRNN by changing it to encode orthogonal polynomials. At the evolution part, the circuit shown in Fig. 3 is applied as the parametrized gate $U(\theta)$, which consists of alternating layers of single-qubit rotations and Hamiltonian dynamics. We fix the Hamiltonian and optimize the angles of the single-qubit rotations. The single-qubit rotations are parametrized as,

$$U_1(\alpha, \beta, \gamma) = R_x(\alpha)R_z(\beta)R_x(\gamma), \quad (9)$$

where α , β , and γ are real numbers, and R_x and R_z are single-qubit rotations around x and z -axis, respectively. After the rotation, the whole system is evolved with the following Hamiltonian,

$$H_{\text{int}} = \sum_{j=1}^n a_j X_j + \sum_{j=1}^n \sum_{k=1}^{j-1} J_{jk} Z_j Z_k, \quad (10)$$

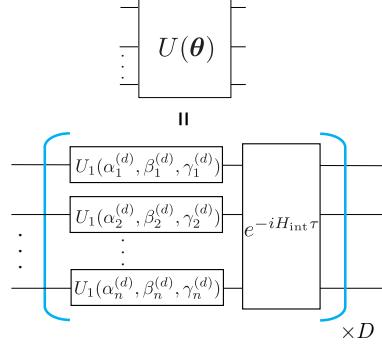


FIG. 3. The evolution part of the circuit utilized in the numerical simulations. $\alpha_i^{(d)}$, $\beta_i^{(d)}$, and $\gamma_i^{(d)}$ are rotation angles and act as independent parameters of the circuit, i.e., θ consists of $\{\alpha_i^{(d)}, \beta_i^{(d)}, \gamma_i^{(d)}\}$. $d = 1, \dots, D$ is the layer index.

following Ref. [7, 10]. We denote the evolution time by τ . We repeat applications of $U_1(\alpha, \beta, \gamma)$ and $e^{-iH_{\text{int}}\tau}$ for $D = 3$ times, as shown in Fig. 3. The coefficients a_j, J_{jk} are taken randomly from a uniform distribution on $[-1, 1]$ and fixed during the training. At the measurement part, we measure Z -expectation value of each qubit in group B and take \bar{y}_t as their average multiplied by a real coefficient c . In the simulation presented in the following subsections, the coefficient c is also optimized along with $(\alpha_i^{(d)}, \beta_i^{(d)}, \gamma_i^{(d)})$ for each qubit. For classical optimizer, we employ the BFGS algorithm implemented in SciPy [35]. All of the parameters are initialized to 0 except for the coefficient c which is initialized to 1.

A. Demonstration

First, we train the above described QRNN to predict cosine wave and triangular wave. For the cosine wave, we use $x_t = \cos(\pi t')/2$, and for the triangular wave, we use

$$x_t = \begin{cases} -t' + \frac{1}{2} & (0 \leq t' \leq 1) \\ t' - \frac{3}{2} & (1 \leq t' \leq 2) \\ -t' + \frac{5}{2} & (2 \leq t' \leq 3) \\ t' - \frac{7}{2} & (3 \leq t' \leq 4) \end{cases}, \quad (11)$$

for $t' = \frac{8}{199}t$. For both of them, $\{x_t\}_{t=0}^{99}$ is used as training data. After the training, we test the accuracy of the predictions for $100 \leq t < 200$. The evolution time τ is set to 0.2 in this numerical experiment.

Secondly, we train QRNN to predict quantum spin dynamics since we envision that the proposed approach is suited for such applications, that is, for predictions of quantum phenomena. Here, we use open quantum dynamics of a 3-spin system generated by a Lindblad mas-

TABLE I. Mean squared error between the trained output and the test data shown in Fig. 4 evaluated with the first 25 points after the training region.

Data	cos	triangle	spin dynamics
MSE	3.33×10^{-4}	2.6×10^{-3}	2.95×10^{-3}

ter equation,

$$\begin{aligned} \frac{d}{dt'}\sigma(t') &= -i[H, \sigma(t')] \\ &+ \sum_k \frac{1}{2} \left[2C_k\sigma(t')C_k^+ - \sigma(t')C_k^\dagger C_k - C_k^\dagger C_k\sigma(t') \right] \end{aligned} \quad (12)$$

with $H = -\frac{1}{2}\sum_{i=1}^3 h^{(i)}Z_i - \frac{1}{2}\sum_{i=1}^2 (J_x^{(i)}X_iX_{i+1} + J_y^{(i)}Y_iY_{i+1} + J_z^{(i)}Z_iZ_{i+1})$ and $\{C_k\} = \{c(X_i + Y_i)\}_{i=1}^3$ as the data to be predicted. The coefficients are set to $h^{(i)} = 2\pi$, $J_\mu^{(i)} = 0.1\pi$ for $\mu = x, y, z$, and $c = \sqrt{0.002}$. The initial state of the system is chosen to be $|0\rangle^{\otimes 3}$. We train the QRNN to predict the X -expectation value of the first spin, $\langle X_1(t') \rangle$, at time $t' = \frac{100}{499}t$, that is, we take $x_t = \langle X_1(t') \rangle$. $\{x_t\}_{t=0}^{199}$ is used for the training, and we test the prediction afterwards for $200 \leq t < 500$. The evolution time τ is set to 0.18 in this numerical experiment.

Figure 4 shows the results of the numerical simulations whose mean squared error (MSE) between the output and the teacher on the first 25 test points are the smallest among 10 randomly generated coefficients a_j, J_{jk} . In the figure, the initial output shows a sequence of predictions $\{\bar{y}_t\}$ obtained from QRNN with the initial parameters. The reason it resembles the true data x_t in the training region is because we input x_t by $R_y(\arccos x_t)$, which encodes x_t to Z -expectation values of qubits. This means that $x_t = \bar{y}_t$ if $\tau = 0$. For nonzero τ , we can naturally expect \bar{y}_t to be somewhat smaller than x_t because of the interaction. Looking at the optimized output, we can see that QRNN can be successfully trained to fit a given training dataset and to make predictions after the training. This can also be verified with MSE between the output and the teacher evaluated on the first 25 test points, which is on the order of 10^{-3} for all data types (Tab. I). During the experiments, we found that the quality of the prediction strongly depends on the parameter τ which determines the interaction strength between the qubits. This phenomenon is further analyzed in the next subsection.

B. Dependence on the interaction strength

Here, we vary τ from 0 to 10 and perform the training of the QRNN on the cosine wave used in Fig. 4 (a) to investigate the dependency of the performance of the QRNN on the interaction strength. For this purpose, we

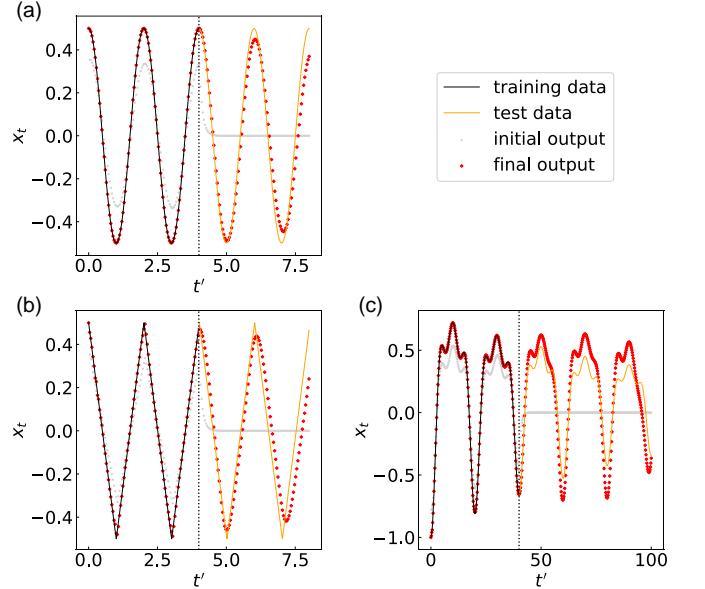


FIG. 4. Results of numerical simulation of training the QRNN to predict (a) cosine wave, (b) triangular wave, and (c) three-spin Lindblad dynamics. The data points on the left of the vertical grey dashed line are used as the training dataset. The solid lines represent the function which the QRNN is trained to model. The parts of the data used in the training phase and the prediction phase are depicted as the black and orange lines, respectively.

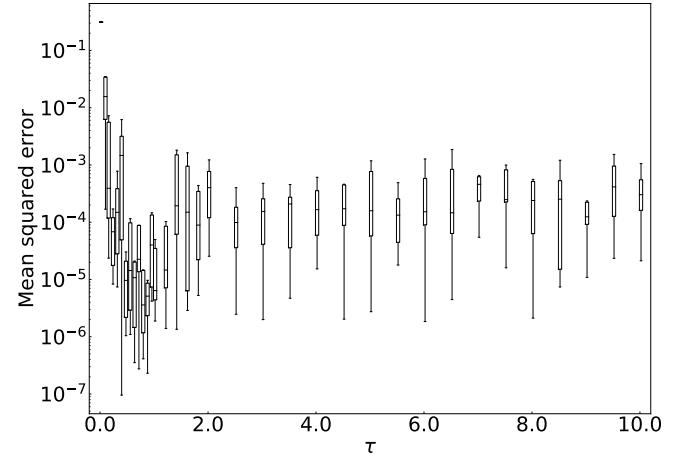


FIG. 5. Relation between the evolution time τ and the prediction error evaluated in terms of the mean squared error.

randomly drew the coefficients of the interaction Hamiltonian H_{int} (Eq. (10)) 10 times from uniform distribution on $[-1, 1]$, and trained each QRNN.

Fig. 5 shows the mean squared error between x_t and \bar{y}_t of the first 25 time steps. From Fig. 5, we notice that there exists a certain range of τ which provides us relatively accurate predictions. More specifically, the QRNN circuits with $\tau \approx a_j, J_{ij}$ work better than the other ones. This is as expected because a short τ leads to less infor-

mation transferred to the subsystem A . It means that the QRNN cannot store the past information, since the subsystem B is measured and initialized at every time step, which erases all information that is left in B . On the other hand, when τ is sufficiently larger than a_j and J_{ij} of the Hamiltonian, the dynamics under H_{int} becomes complex, which makes it hard for the single-qubit rotations to extract the required information from the system. This analysis shows that the QRNN indeed utilizes the past information stored in the subsystem B , and we must have a certain amount of entangling gates at the evolution part of the QRNN.

V. CONCLUSION

We proposed a quantum version of an RNN for temporal learning tasks. The proposed algorithm, QRNN, employs a parametrized quantum circuit with a recurrent structure. In QRNN, one group of qubits are measured and initialized at every time step to obtain outputs and inject inputs, while other qubits are never measured to store the past information. We provided numerical experiments to verify the validity of the idea.

It remains an open question whether the proposed QRNN performs better than the classical RNN, and we leave it as a future research direction. However, we note that the QRC which has a similar structure as the QRNN has been shown to have better performance than its classical counterpart [10]. We can, therefore, expect the QRNN to have similar performance when its structure is optimized. As the QRNN explicitly optimize the dynamics of the quantum system, we especially expect it to be able to predict quantum phenomena, such as the one we used in our numerical experiments.

ACKNOWLEDGMENTS

KM is supported by JSPS KAKENHI No. 19J10978, 20K22330 and JST PRESTO JPMJPR2019. KF is supported by KAKENHI No.16H02211, JST PRESTO JPMJPR1668, JST ERATO JPMJER1601, and JST CREST JPMJCR1673. MN is supported by JST PRESTO JPMJPR1666. This work is supported by MEXT Quantum Leap Flagship Program (MEXT Q-LEAP) Grant Number JPMXS0118067394 and JPMXS0120319794.

Appendix: Calculation of analytic gradient

Here we describe how to evaluate analytic gradient in the QRNN using the so-called parameter shift rule [7, 18, 19, 36]. In short, the rule allows us to evaluate an expectation value of any observable O with respect to the

following operator,

$$\frac{\partial}{\partial \theta_i} [U(\boldsymbol{\theta})\rho U^\dagger(\boldsymbol{\theta})] = \frac{\partial U(\boldsymbol{\theta})}{\partial \theta_i} \rho U^\dagger(\boldsymbol{\theta}) + U(\boldsymbol{\theta})\rho \frac{\partial U^\dagger(\boldsymbol{\theta})}{\partial \theta_i}, \quad (\text{A.1})$$

for any state ρ by evaluating the expectation value twice at shifted parameters if the parametrized gate corresponding to the parameter θ_i can be written as $e^{i\theta_i P}$ for some Pauli operator P .

For simplicity, we assume x_t to be scalar and that the output is obtained from a single observable O as $\bar{y}_t = g(\langle O \rangle_t)$. Also, the cost function is assumed to be $L = \frac{1}{2} \sum_{t=0}^{T-2} (\bar{y}_t - x_{t+1})^2$. However, generalization of the discussion below to other cost functions is straight forward. Now, the gradient of L can be written as,

$$\frac{\partial L}{\partial \theta_i} = \sum_{t=0}^{T-2} (\bar{y}_t(\boldsymbol{\theta}, x_0, \dots, x_t) - x_{t+1}) \frac{\partial \bar{y}_t}{\partial \theta_i}. \quad (\text{A.2})$$

Next, we can express $\frac{\partial \bar{y}_t}{\partial \theta_i}$ as,

$$\frac{\partial \bar{y}_t}{\partial \theta_i} = \frac{\partial g(\langle O \rangle_t)}{\partial \theta_i} = \frac{\partial g}{\partial \langle O \rangle_t} \frac{\partial \langle O \rangle_t}{\partial \theta_i}. \quad (\text{A.3})$$

Let us define the input state $\rho_{\text{in},t}^B := U_{\text{in}}(x_t) |0\rangle \langle 0| U_{\text{in}}^\dagger(x_t)$ and $\rho_{\text{in},t}^{AB}(\boldsymbol{\theta}, x_0, \dots, x_t) = \rho_{t-1}^A \otimes \rho_{\text{in},t}^B$. Below we abbreviate the dependence of $\rho_{\text{in},t}^{AB}$ on x_0, \dots, x_t and just write $\rho_{\text{in},t}^{AB}(\boldsymbol{\theta})$ for simplicity. Then, $\frac{\partial \langle O \rangle_t}{\partial \theta_i}$ can be written as,

$$\begin{aligned} \frac{\partial \langle O \rangle_t}{\partial \theta_i} &= \frac{\partial}{\partial \theta_i} \text{Tr} [\rho_t^B O] \\ &= \frac{\partial}{\partial \theta_i} \text{Tr} [U(\boldsymbol{\theta}) \rho_{\text{in},t}^{AB}(\boldsymbol{\theta}) U^\dagger(\boldsymbol{\theta}) O] \\ &= \text{Tr} \left[\frac{\partial U(\boldsymbol{\theta})}{\partial \theta_i} \rho_{\text{in},t}^{AB}(\boldsymbol{\theta}) U^\dagger(\boldsymbol{\theta}) O \right] \\ &\quad + \text{Tr} \left[U(\boldsymbol{\theta}) \rho_{\text{in},t}^{AB}(\boldsymbol{\theta}) \frac{\partial U^\dagger(\boldsymbol{\theta})}{\partial \theta_i} O \right] \\ &\quad + \text{Tr} \left[U(\boldsymbol{\theta}) \frac{\partial \rho_{\text{in},t}^{AB}(\boldsymbol{\theta})}{\partial \theta_i} U^\dagger(\boldsymbol{\theta}) O \right]. \end{aligned} \quad (\text{A.4})$$

In Eq. (A.4), the first two terms can easily be evaluated by the parameter shift rule. As for the last term,

$$\begin{aligned} \frac{\partial \rho_{\text{in},t}^{AB}(\boldsymbol{\theta})}{\partial \theta_i} &= \frac{\partial \rho_{t-1}^A(\boldsymbol{\theta})}{\partial \theta_i} \otimes \rho_{\text{in},t}^B \\ &= \frac{\partial}{\partial \theta_i} \text{Tr}_B [U(\boldsymbol{\theta}) \rho_{\text{in},t-1}^{AB}(\boldsymbol{\theta}) U^\dagger(\boldsymbol{\theta})] \otimes \rho_t^B \\ &= \left\{ \text{Tr}_B \left[\frac{\partial U(\boldsymbol{\theta})}{\partial \theta_i} \rho_{\text{in},t-1}^{AB}(\boldsymbol{\theta}) U^\dagger(\boldsymbol{\theta}) \right] \right. \\ &\quad \left. + \text{Tr}_B \left[U(\boldsymbol{\theta}) \rho_{\text{in},t-1}^{AB}(\boldsymbol{\theta}) \frac{\partial U^\dagger(\boldsymbol{\theta})}{\partial \theta_i} \right] \right. \\ &\quad \left. + \text{Tr}_B \left[U(\boldsymbol{\theta}) \frac{\partial \rho_{\text{in},t-1}^{AB}(\boldsymbol{\theta})}{\partial \theta_i} U^\dagger(\boldsymbol{\theta}) \right] \right\} \otimes \rho_t^B. \end{aligned} \quad (\text{A.5})$$

In the above, the first two terms can be evaluated with parameter shift rule. For the last term, we need to expand it again to express $U(\boldsymbol{\theta}) \frac{\partial \rho_{\text{in},t-1}^{AB}(\boldsymbol{\theta})}{\partial \theta_i} U^\dagger(\boldsymbol{\theta})$ in terms of $U(\boldsymbol{\theta}) \frac{\partial \rho_{\text{in},t-2}^{AB}(\boldsymbol{\theta})}{\partial \theta_i} U^\dagger(\boldsymbol{\theta})$. The above discussion implies that, to evaluate $U(\boldsymbol{\theta}) \frac{\partial \rho_t^{AB}(\boldsymbol{\theta})}{\partial \theta_i} U^\dagger(\boldsymbol{\theta})$ which appears in

Eq. (A.4), we have to perform the above expansion recursively down to $U(\boldsymbol{\theta}) \frac{\partial \rho_{\text{in},0}^{AB}}{\partial \theta_i} U^\dagger(\boldsymbol{\theta})$, where the expansion terminates since $\rho_{\text{in},0}^{AB} = |\rangle \langle| \otimes \rho_{\text{in}}^B(x_0)$ does not depend on $\boldsymbol{\theta}$. As every recursive expansion yields a pair of terms that can be evaluated by shifting the parameter twice, we need $2T$ additional evaluations of $\langle O \rangle$ to obtain the analytical gradient of the cost function for each θ_i .

-
- [1] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, Quantum machine learning, *Nature* **549**, 195 (2017).
- [2] M. Schuld, I. Sinayskiy, and F. Petruccione, Prediction by linear regression on a quantum computer, *Phys. Rev. A* **94**, 022342 (2016).
- [3] N. Wiebe, D. Braun, and S. Lloyd, Quantum Algorithm for Data Fitting, *Phys. Rev. Lett.* **109**, 050505 (2012).
- [4] P. Rebentrost, M. Mohseni, and S. Lloyd, Quantum support vector machine for big data classification, *Phys. Rev. Lett.* **113**, 130503 (2014).
- [5] H. Yamasaki, S. Subramanian, S. Sonoda, and M. Koashi, Learning with optimized random features: Exponential speedup by quantum machine learning without sparsity and low-rank assumptions (2020), arXiv:2004.10756 [quant-ph].
- [6] M. Schuld and N. Killoran, Quantum machine learning in feature hilbert spaces, *Phys. Rev. Lett.* **122**, 040504 (2019).
- [7] K. Mitarai, M. Negoro, M. Kitagawa, and K. Fujii, Quantum circuit learning, *Phys. Rev. A* **98**, 032309 (2018).
- [8] E. Farhi and H. Neven, Classification with quantum neural networks on near term processors (2018), arXiv:1802.06002 [quant-ph].
- [9] M. Cerezo, A. Arrasmith, R. Babbush, S. C. Benjamin, S. Endo, K. Fujii, J. R. McClean, K. Mitarai, X. Yuan, L. Cincio, and P. J. Coles, Variational quantum algorithms (2020), arXiv:2012.09265 [quant-ph].
- [10] K. Fujii and K. Nakajima, Harnessing disordered-ensemble quantum dynamics for machine learning, *Phys. Rev. Applied* **8**, 024030 (2017).
- [11] A. Sherstinsky, Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network, *Physica D: Nonlinear Phenomena* **404**, 132306 (2020).
- [12] S. Hochreiter and J. Schmidhuber, Long short-term memory, *Neural computation* **9**, 1735 (1997).
- [13] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, Empirical evaluation of gated recurrent neural networks on sequence modeling (2014), arXiv:1412.3555 [cs.NE].
- [14] J. L. Elman, Finding structure in time, *Cognitive science* **14**, 179 (1990).
- [15] A. Peruzzo, J. McClean, P. Shadbolt, M.-H. Yung, X.-Q. Zhou, P. J. Love, A. Aspuru-Guzik, and J. L. O’Brien, A variational eigenvalue solver on a photonic quantum processor, *Nature Communications* **5**, 4213 (2014).
- [16] E. Farhi, J. Goldstone, and S. Gutmann, A quantum approximate optimization algorithm (2014), arXiv:1411.4028 [quant-ph].
- [17] M. Benedetti, D. Garcia-Pintos, O. Perdomo, V. Leyton-Ortega, Y. Nam, and A. Perdomo-Ortiz, A generative modeling approach for benchmarking and training shallow quantum circuits, *npj Quantum Information* **5**, 45 (2019).
- [18] M. Schuld, V. Bergholm, C. Gogolin, J. Izaac, and N. Killoran, Evaluating analytic gradients on quantum hardware, *Phys. Rev. A* **99**, 032331 (2019).
- [19] B. Koczor and S. C. Benjamin, Quantum analytic descent (2020), arXiv:2008.13774 [quant-ph].
- [20] W. de Boer, M. Borghini, K. Morimoto, T. O. Niinikoski, and F. Udo, Dynamic polarization of protons, deuterons, and carbon-13 nuclei: Thermal contact between nuclear spins and an electron spin-spin interaction reservoir, *Journal of Low Temperature Physics* **15**, 249 (1974).
- [21] A. Henstra, T.-S. Lin, J. Schmidt, and W. Wenckebach, High dynamic nuclear polarization at room temperature, *Chemical Physics Letters* **165**, 6 (1990).
- [22] K. Tateishi, M. Negoro, S. Nishida, A. Kagawa, Y. Morita, and M. Kitagawa, Room temperature hyperpolarization of nuclear spins in bulk, *Proceedings of the National Academy of Sciences* **111**, 7527 (2014).
- [23] R. Fischer, A. Jarmola, P. Kehayias, and D. Budker, Optical polarization of nuclear ensembles in diamond, *Phys. Rev. B* **87**, 125207 (2013).
- [24] J. Chen and H. I. Nurdin, Learning nonlinear input-output maps with dissipative quantum systems, *Quantum Information Processing* **18**, 198 (2019).
- [25] J. Chen, H. I. Nurdin, and N. Yamamoto, Temporal information processing on noisy quantum computers, *Phys. Rev. Applied* **14**, 024065 (2020).
- [26] S. Ghosh, A. Opala, M. Matuszewski, T. Paterek, and T. C. H. Liew, Quantum reservoir processing, *npj Quantum Information* **5**, 35 (2019).
- [27] S. Ghosh, A. Opala, M. Matuszewski, T. Paterek, and T. C. H. Liew, Reconstructing quantum states with quantum reservoir networks, *IEEE Transactions on Neural Networks and Learning Systems* , 1 (2020).
- [28] S. Ghosh, T. Krisnanda, T. Paterek, and T. C. H. Liew, Universal quantum reservoir computing (2020), arXiv:2003.09569 [quant-ph].
- [29] L. C. G. Govia, G. J. Ribeill, G. E. Rowlands, H. K. Krovi, and T. A. Ohki, Quantum reservoir computing with a single nonlinear oscillator (2020), arXiv:2004.14965 [quant-ph].
- [30] R. Martínez-Peña, J. Nokkala, G. L. Giorgi, R. Zambrini, and M. C. Soriano, Information processing capacity of spin-based quantum reservoir computing systems, *Cognitive Computation* (2020).
- [31] A. Kutvonen, K. Fujii, and T. Sagawa, Optimizing a quantum reservoir computer for time series prediction, *Scientific Reports* **10**, 14687 (2020).
- [32] M. Negoro, K. Mitarai, K. Fujii, K. Nakajima, and

- M. Kitagawa, Machine learning with controllable quantum dynamics of a nuclear spin ensemble in a solid (2018), arXiv:1806.10910 [quant-ph].
- [33] J. Bausch, Recurrent quantum neural networks (2020), arXiv:2006.14619 [cs.LG].
- [34] Y. Cao, G. G. Guerreschi, and A. Aspuru-Guzik, Quantum neuron: an elementary building block for machine learning on quantum computers (2017), arXiv:1711.11240 [quant-ph].
- [35] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, A. Vijaykumar, A. P. Bardelli, A. Rothberg, A. Hilboll, A. Kloekner, A. Scopatz, A. Lee, A. Rokem, C. N. Woods, C. Fultton, C. Masson, C. Häggström, C. Fitzgerald, D. A. Nicholson, D. R. Hagen, D. V. Pasechnik, E. Olivetti, E. Martin, E. Wieser, F. Silva, F. Lenders, F. Wilhelm, G. Young, G. A. Price, G.-L. Ingold, G. E. Allen, G. R. Lee, H. Audren, I. Probst, J. P. Dietrich, J. Silterra, J. T. Webber, J. Slavić, J. Nothman, J. Buchner, J. Kulick, J. L. Schönberger, J. V. de Miranda Cardoso, J. Reimer, J. Harrington, J. L. C. Rodríguez, J. Nunez-Iglesias, J. Kuczynski, K. Tritz, M. Thoma, M. Newville, M. Kümmeler, M. Bolingbroke, M. Tartre, M. Pak, N. J. Smith, N. Nowaczyk, N. Shebanov, O. Pavlyk, P. A. Brodtkorb, P. Lee, R. T. McGibbon, R. Feldbauer, S. Lewis, S. Tygier, S. Sievert, S. Vigna, S. Peterson, S. More, T. Pudlik, T. Oshima, T. J. Pingel, T. P. Robitaille, T. Spura, T. R. Jones, T. Cera, T. Leslie, T. Zito, T. Krauss, U. Upadhyay, Y. O. Halchenko, Y. Vázquez-Baeza, and Scipy 1.0 Contributors, Scipy 1.0: fundamental algorithms for scientific computing in python, *Nature Methods* **17**, 261 (2020).
- [36] K. Mitarai and K. Fujii, Methodology for replacing indirect measurements with direct measurements, *Phys. Rev. Research* **1**, 013006 (2019).