algorithms

MDPI

Article

Quantum Recurrent Neural Networks: Predicting the Dynamics of Oscillatory and Chaotic Systems

Yuan Chen [1,†] and Abdul Khaliq [2,*,†]

[1] Computational and Data Science Program, Middle Tennessee State University, Murfreesboro, TN 37132, USA; yc3y@mtmail.mtsu.edu
[2] Department of Mathematical Sciences, Middle Tennessee State University, Murfreesboro, TN 37132, USA
* Correspondence: abdul.khaliq@mtsu.edu
† These authors contributed equally to this work.

Abstract: In this study, we investigate Quantum Long Short-Term Memory and Quantum Gated Recurrent Unit integrated with Variational Quantum Circuits in modeling complex dynamical systems, including the Van der Pol oscillator, coupled oscillators, and the Lorenz system. We implement these advanced quantum machine learning techniques and compare their performance with traditional Long Short-Term Memory and Gated Recurrent Unit models. The results of our study reveal that the quantum-based models deliver superior precision and more stable loss metrics throughout 100 epochs for both the Van der Pol oscillator and coupled harmonic oscillators, and 20 epochs for the Lorenz system. The Quantum Gated Recurrent Unit outperforms competing models, showcasing notable performance metrics. For the Van der Pol oscillator, it reports MAE 0.0902 and RMSE 0.1031 for variable x and MAE 0.1500 and RMSE 0.1943 for y; for coupled oscillators, Oscillator 1 shows MAE 0.2411 and RMSE 0.2701 and Oscillator 2 MAE is 0.0482 and RMSE 0.0602; and for the Lorenz system, the results are MAE 0.4864 and RMSE 0.4971 for x, MAE 0.4723 and RMSE 0.4846 for y, and MAE 0.4555 and RMSE 0.4745 for z. These outcomes mark a significant advancement in the field of quantum machine learning.

Keywords: Quantum Long Short-Term Memory (QLSTM); Quantum Gated Recurrent Unit (QGRU); Van der Pol oscillator; harmonic oscillators; Lorenz system

Citation: Chen, Y.; Khaliq, A. Quantum Recurrent Neural Networks: Predicting the Dynamics of Oscillatory and Chaotic Systems. *Algorithms* **2024**, *17*, 163. https://doi.org/10.3390/a17040163

Academic Editor: Frank Werner

Received: 24 March 2024
Revised: 13 April 2024
Accepted: 17 April 2024
Published: 19 April 2024

1. Introduction

Ordinary differential equations (ODEs) are foundational mathematical models that play a critical role across diverse domains, such as physics [1], biology [2,3], and economics [4]. The numerical resolution of ODE systems is pivotal for understanding the dynamics of these varied processes. To this end, numerous methodologies have been devised, including linear multistep methods [5,6] and Runge–Kutta methods [7,8]. Nevertheless, the practical application of these methods frequently encounters the complexity of systems characterized by nonlinear and time-varying elements, alongside a pronounced sensitivity to initial conditions. Such complexities [9] significantly challenge the efficacy of traditional analytical and numerical strategies, especially in terms of achieving desired levels of precision, stability, and computational efficiency. Given these considerations, there is a pressing need for more sophisticated techniques capable of navigating the intricate landscape of ODE systems.

In recent years, recurrent neural networks (RNNs) have achieved significant breakthroughs in processing sequential data. RNNs have been used extensively in various domains, including but not limited to, time series data prediction [10–12], machine translation [13], and speech recognition [14]. Notably, RNNs have also played a critical role in scientific research [15,16], an area of particular interest to us. Within this framework, RNNs and their specific variants, such as Long Short-Term Memory (LSTM) [17] and Gated

Recurrent Units (GRUs) [18], have proven to be efficacious in learning or forecasting complex, nonlinear challenges. The intrinsic nature of RNNs, which are capable of capturing temporal dependencies and learning from sequential data, makes them particularly suited for modeling dynamical systems. Unlike conventional approaches that require explicit numerical schemes for integration over time, RNNs can implicitly learn the underlying dynamics from data, offering a way to bypass the complexities of directly solving ordinary differential equations (ODEs). This capability allows for the modeling of complex systems with high degrees of freedom and nonlinearity, providing a flexible and potentially more accurate alternative to traditional methods. Consequently, the use of RNNs in solving ODEs, especially systems of ODEs, represents a significant shift towards data-driven approaches in the prediction of dynamical [19–22] and time series chaotic systems [23–26].

Concurrently, quantum computing has been advancing rapidly, spearheaded by technology giants such as IBM [27], Google [28], PennyLane [29], and D-Wave Systems [30]. These quantum systems offer theoretical advantages for specific computational tasks and simulations of intricate quantum phenomena. However, the practical utility of current Noisy Intermediate-Scale Quantum (NISQ) devices is constrained by limitations in quantum error correction [31–33]. This limitation poses a significant challenge, as it restricts the reliability and scalability of quantum computations, impeding their broader application in complex problems.

The advent of variational quantum algorithms and circuits, pioneered by Mitarai et al. [34], has been a landmark development in the integration of quantum computing with machine learning. These algorithms leverage quantum entanglement to address fundamental machine learning challenges, such as function approximation and classification [34,35]. More studies about variational quantum algorithms and circuits can be found here [36]. This innovation has facilitated the development of hybrid quantum–classical algorithms, adaptable for use on existing Noisy Intermediate-Scale Quantum (NISQ) devices. Such hybrid approaches have shown promise across various domains, including classification [37,38], generative adversarial learning [39], and deep reinforcement learning [40]. Furthermore, the exploration of quantum machine learning [41–46] has opened a new path for processing and analyzing data, harnessing the unique computational capabilities of quantum systems.

Among the studies, the study by Chen et al. [47,48] innovated in the quantum machine learning domain by introducing Quantum LSTM (QLSTM) and Quantum GRU (QGRU), integrating variational quantum circuits (VQCs) with recurrent neural network architectures. Designed for Noisy Intermediate-Scale Quantum (NISQ) devices, these quantum models utilize quantum entanglement to enhance learning efficiency and stability. Their research showcases that QLSTM and QGRU outperform traditional LSTM models in terms of learning efficiency and convergence stability, marking a significant step in sequential data learning within the quantum computing sphere. However, the scenarios examined in Chen's studies [47,48] only involved singular ODEs, which prompted us to explore the application of QRNN to systems of ODEs.

In this research, we explore the application of Quantum Long Short-Term Memory (QLSTM) and Quantum Gated Recurrent Unit (QGRU) models, based on the Variational Quantum Circuits (VQCs) developed by Chen et al. [47,48], to complex dynamical systems described by ODEs. Our approach involves a nuanced modification to the VQC, thereby refining its compatibility with QLSTM and QGRU models for their application in this context. Our focus is on the Van der Pol oscillator, two coupled damped harmonic oscillators, and the Lorenz equations. We conduct a comparative analysis of these quantum models against their classical counterparts, LSTM and GRU, focusing on error metrics and loss evolution across epochs. This comparison aims to evaluate their effectiveness in modeling and predicting the dynamics of nonlinear and chaotic systems.

The structure of this paper is organized as follows: Section 2 presents an overview of fundamental concepts in quantum computing, including qubits, superposition, quantum gates, and entanglement. In Section 3, we introduce the computational models under inves-

tigation, namely variational quantum circuits, classical Long Short-Term Memory (LSTM), classical Gated Recurrent Unit (GRU), Quantum Long Short-Term Memory (QLSTM), and Quantum Gated Recurrent Unit (QGRU). Section 4 introduces three numerical experiments conducted to evaluate these models, including the Van der Pol oscillator, two coupled damped harmonic oscillators, and the Lorenz equations system. Section 5 is the discussion and Section 6 is the conclusion.

2. Fundamental Quantum Computing Concepts

2.1. Qubits

The core distinction between classical and quantum computing lies in their fundamental units of information: the classical bit and the quantum bit (qubit), respectively. In classical computing, a bit is a binary unit that can exist in one of two states, either 0 or 1. This binary nature underpins all classical computing processes, with data storage and computations executed through combinations of these binary states.

In contrast, the qubit, the fundamental unit of quantum computing, transcends this binary limitation. A qubit differs from a classical bit in its ability to exist in a superposition of states. Rather than being limited to a strict 0 or 1, a qubit can represent both 0 and 1 simultaneously, a phenomenon that is central to quantum computing's potential for processing complexity.

This unique property of qubits arises from the principles of quantum mechanics. Unlike classical bits, whose state is definite and observable, a qubit's state is described probabilistically. The actual state of a qubit is not determined until it is measured. Before measurement, a qubit exists in a superposition of the states $|0\rangle$ and $|1\rangle$, where the probabilities of these states are determined by quantum amplitudes. These amplitudes, which are complex numbers, dictate the likelihood of the qubit collapsing into either the $|0\rangle$ or $|1\rangle$ state upon measurement.

The concept of superposition enables qubits to perform computations on a scale and at a speed unattainable by classical bits. This capability forms the basis for quantum computing's potential to solve problems that are currently intractable for classical computers, which will be introduced in the following section.

2.2. Superpositions

Superposition in a quantum system allows a qubit to exist in multiple states simultaneously until it is measured. This characteristic differentiates a quantum bit from a classical bit, which can only be in one state at any given time. In a single qubit system, the state of a qubit can be described as a linear combination of the basis states $|0\rangle$ and $|1\rangle$. Mathematically, this is represented as:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle, \tag{1}$$

where α and β are complex numbers representing the probability amplitudes for the qubit being in state $|0\rangle$ and $|1\rangle$, respectively. According to the Born's rule, the probabilities of finding the qubit in either state upon measurement are $|\alpha|^2$ and $|\beta|^2$, with the condition that $|\alpha|^2 + |\beta|^2 = 1$.

For example, if $\alpha = \sqrt{0.3}$ and $\beta = \sqrt{0.7}$, the qubit has a 30% probability of being measured in state $|0\rangle$ and a 70% probability of being in state $|1\rangle$.

Extending the concept of superposition to a two-qubit system, we find a more complex scenario. In a two-qubit system, the combined state can be represented as a superposition of all four possible states of the two qubits. The state vector for such a system is given by:

$$|\Psi\rangle = \alpha|00\rangle + \beta|01\rangle + \gamma|10\rangle + \delta|11\rangle, \tag{2}$$

where α, β, γ, and δ are complex probability amplitudes corresponding to each of the four states. As per the normalization condition, the sum of the squares of the absolute values of these amplitudes must equal one:

$$|\alpha|^2 + |\beta|^2 + |\gamma|^2 + |\delta|^2 = 1. \tag{3}$$

For instance, if the amplitudes are $\alpha = \sqrt{0.1}, \beta = \sqrt{0.2}, \gamma = \sqrt{0.3}, and \delta = \sqrt{0.4}$, the probabilities of measuring the system in the states $|00\rangle$, $|01\rangle$, $|10\rangle$, and $|11\rangle$ are 10%, 20%, 30%, and 40%, respectively.

In both single and two-qubit systems, superposition allows quantum computers to process and encode information in ways that are fundamentally different from classical computers, enabling them to perform complex calculations more efficiently.

In traditional computing, gates are fundamental building blocks that process binary information, operating on bits that exist in one of two states: 0 or 1. Common logic gates include AND, OR, NOT, and XOR, each performing a specific logical operation for computational tasks.

Similarly, quantum computing has quantum gates, which manipulate qubits, such as the Hadamard, Pauli-X, Y, Z, and Controlled-NOT (CNOT) gates, operating on these qubits, enabling complex operations that can entangle qubits, and creating correlations between them that are essential for quantum computation's power. These gates are unitary, meaning they are reversible, a property that contrasts with some irreversible classical gates.

Here, we introduce one of the fundamental gates in quantum computing, the Hadamard gate, often used to create a superposition of states. The Hadamard gate acts on a single qubit and transforms it into a superposition of its basis states.

The Hadamard gate is represented by the following matrix:

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}. \tag{4}$$

This matrix operates on a qubit state vector, transforming it from a definite state ($|0\rangle$ or $|1\rangle$) into a superposition. When the Hadamard gate acts on the state $|0\rangle$ (represented by the vector $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$), the resulting state is:

$$H|0\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \frac{|0\rangle + |1\rangle}{\sqrt{2}}. \tag{5}$$

Similarly, when it acts on the state $|1\rangle$ (represented by the vector $\begin{pmatrix} 0 \\ 1 \end{pmatrix}$), the output is:

$$H|1\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ -1 \end{pmatrix} = \frac{|0\rangle - |1\rangle}{\sqrt{2}}. \tag{6}$$

The Hadamard gate, therefore, creates an equal superposition of the $|0\rangle$ and $|1\rangle$ states.

2.3. Quantum Entanglement

Quantum entanglement is a phenomenon in which multiple qubits become interconnected in such a way that the state of one qubit cannot be described independently of the state of the other qubits, even when the qubits are separated by large distances.

To explain the significance of entanglement in this study, consider an analogy involving the conventional method of information exchange between two individuals. The most traditional method of information exchange involves one individual physically moving within audible distance to relay news to another person. This method, while effective, is indeed time-consuming and limits the speed of information transfer. However, by incorporating quantum entanglement into our framework to enable instantaneous information exchange between two entangled parties we significantly enhance the efficiency and speed of learning. This quantum advantage mirrors the leap from having to physically move towards someone to share information, to an immediate and direct exchange of knowledge.

Entanglement can be achieved using certain quantum gates. For instance, the combination of a Hadamard gate followed by a Controlled-NOT (CNOT) gate is commonly used to entangle two qubits.

First, the Hadamard gate is applied to one of the qubits to create a superposition, as previously discussed. Then, the CNOT gate, which flips the state of the second qubit if the first qubit is in the $|1\rangle$ state, is applied. The operation of the CNOT gate can be described by the following matrix:

$$\text{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}. \tag{7}$$

This operation creates entanglement between the two qubits.

Consider two qubits, initially in the state $|00\rangle$. Applying a Hadamard gate to the first qubit creates the superposition $\frac{|0\rangle+|1\rangle}{\sqrt{2}}$ for the first qubit. Now, the combined state of the two qubits is $\frac{|00\rangle+|10\rangle}{\sqrt{2}}$. Applying a CNOT gate afterward results in the entangled state $\frac{|00\rangle+|11\rangle}{\sqrt{2}}$, where the state of each qubit cannot be described independently of the other.

3. Methodology

3.1. Variational Quantum Circuits (VQCs)

Variational Quantum Circuits (VQCs) are the unique part of the architecture of quantum recurrent neural networks. The specific VQC components used in this context are structured in three main parts: an encoding layer, a variational layer, and a quantum measurement layer. Figure 1 shows an example of a VQC.

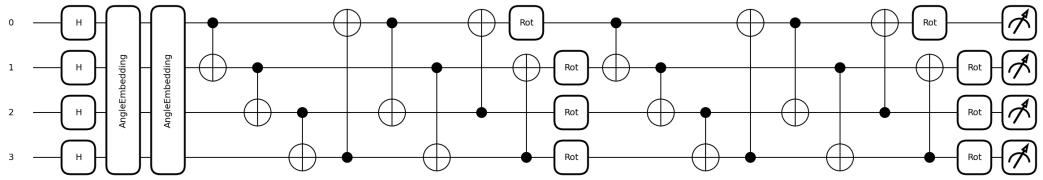

Figure 1. An example of VQC architecture with two layers of entanglements for QLSTM and QGRU.

The left-hand part of the figure is the encoding layer, which contains a Hadamard gate, two angle embedding layers working as R_y and R_z gates in the quantum concept. The middle part is the variational layer (entanglement layer). It is worth mentioning that the number of qubits (in this example is 4), and the number of variational layers (in this example is 2), can be modified to increase the capability to learn the data.

3.1.1. Encoding Layer

The encoding layer's primary function is to map classical data values into quantum amplitudes. It starts with initializing the circuit in the ground state and applying Hadamard gates to create an unbiased initial state. The state of an N-qubit quantum system can be represented as:

$$|\psi\rangle = \sum_{(q_1,q_2,\ldots,q_N)\in\{0,1\}^N} c_{q_1,q_2,\ldots,q_N}|q_1\rangle \otimes |q_2\rangle \otimes \ldots \otimes |q_N\rangle, \tag{8}$$

where $c_{q_1,\ldots,q_N} \in \mathbb{C}$ is the complex amplitude for each basis state and $\otimes$ stands for tensor product. Again, by the Born's rule, we know that:

$$\sum_{(q_1,q_2,\ldots,q_N)\in\{0,1\}^N} \|c_{q_1,\ldots,q_N}\|^2 = 1. \tag{9}$$

The encoding layer first transforms the input data into rotation angles to the rotation of each single qubit. We apply the Hadamard gate here and transform the initial state into a superposition, also called the unbiased state. After applying the Hadamard gate H N times (once to each qubit), we will obtain:

$$(H|0\rangle)^{\otimes N} = \left(\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)\right)^{\otimes N}. \tag{10}$$

The resulting state after applying the Hadamard gate N times is a uniform superposition of all possible states of N qubits, which can be expressed as:

$$\frac{1}{\sqrt{2^N}}\left(|0\rangle^{\otimes N} + \cdots + |1\rangle^{\otimes N}\right). \tag{11}$$

Hence,

$$\frac{1}{\sqrt{2^N}}\sum_{i=0}^{2^N-1}|i\rangle. \tag{12}$$

Here, i is used as an index to sum over all possible states, where i represents the decimal equivalent of the binary numbers formed by the qubits. For example, for $N = 2$, the sum would run over $|00\rangle$, $|01\rangle$, $|10\rangle$, and $|11\rangle$, which correspond to decimal 0, 1, 2, and 3, respectively.

The encoding process involves using two-angle encoding, where each data value is encoded to a qubit with a series of two gates, R_y and R_z. In this study, a template named *qml.templates.AngleEmbedding* is used to play the role of R_y and R_z.

The *qml.templates.AngleEmbedding* template effectively maps classical information onto the quantum states by applying specific rotation gates to each qubit in the system. This template can be customized for various aspects of the embedding, including selecting the axis of rotation R_x, R_y, or R_z. This adaptability makes the template well-suited for a broad spectrum of applications within the domain of quantum neural networks and other machine learning models.

After transforming the initial states into unbiased states, we will use $2N$ rotational angles from an N-dimensional input vector, $\vec{v} = (x_1, x_2, \ldots, x_N)$. For each component x_i of $\vec{v}$, in Chen's original paper, the two angles are calculated: $\theta_{i,1} = \arctan(x_i)$ for y-axis rotation and $\theta_{i,2} = \arctan(x_i^2)$ for z-axis rotation, where the rotations are effected through $R_y(\theta_{i,1})$ and $R_z(\theta_{i,2})$ gates, respectively.

Despite normalizing the input data before the encoding layer, this work introduces a distinct methodology. Rather than employing arctan functions, we select the sin and cos functions for the $R_y(\theta_{i,1})$ and $R_z(\theta_{i,2})$ gates, respectively. This entails setting $\theta_{i,1} = \sin(x_i)$ for the R_y gate and $\theta_{i,2} = \cos(x_i^2)$ for the R_z gate, diverging from conventional arctan applications. This method represents a preliminary attempt in practice and it can be improved in future work, which is not covered within this paper. More studies related to this aspect can be found in Mitarai's paper [34].

3.1.2. Variational Circuit

The variational circuit is the trainable part of the VQC, consisting of parameterized unitary transformations. This section of the circuit includes multiple CNOT gates for qubit entanglement and unitary rotation gates controlled by learnable parameters α, β, and γ. It is worth mentioning that the variational circuit can be repeated more than one time in practice to increase the number of parameters and the model's expressive capacity.

3.1.3. Quantum Measurement

Quantum measurement is used for extracting classical information from the quantum circuit. It involves measuring the qubits, which, due to the probabilistic nature of quantum systems, yield varying bit strings upon each measurement. The expectation value of an operator $\hat{O}$ for a state $|\psi\rangle$ is given by:

$$E[\hat{O}] = \langle\psi|\hat{O}|\psi\rangle. \tag{13}$$

The expectation values can be either calculated analytically in a quantum simulation or obtained through multiple samplings in practical quantum devices with specific noise models.

The VQC architecture described here is pivotal in the QLSTM and QGRU, as a quantum-enhanced approach to processing and learning from data.

3.2. Long Short-Term Memory (LSTM)

Long Short-Term Memory (LSTM) networks are a type of recurrent neural network (RNN) particularly adept at learning from sequences of data. LSTMs are designed to overcome the limitations of traditional RNNs, especially issues related to long-term dependencies in data sequences. Figure 2 shows the structure of a single LSTM unit.

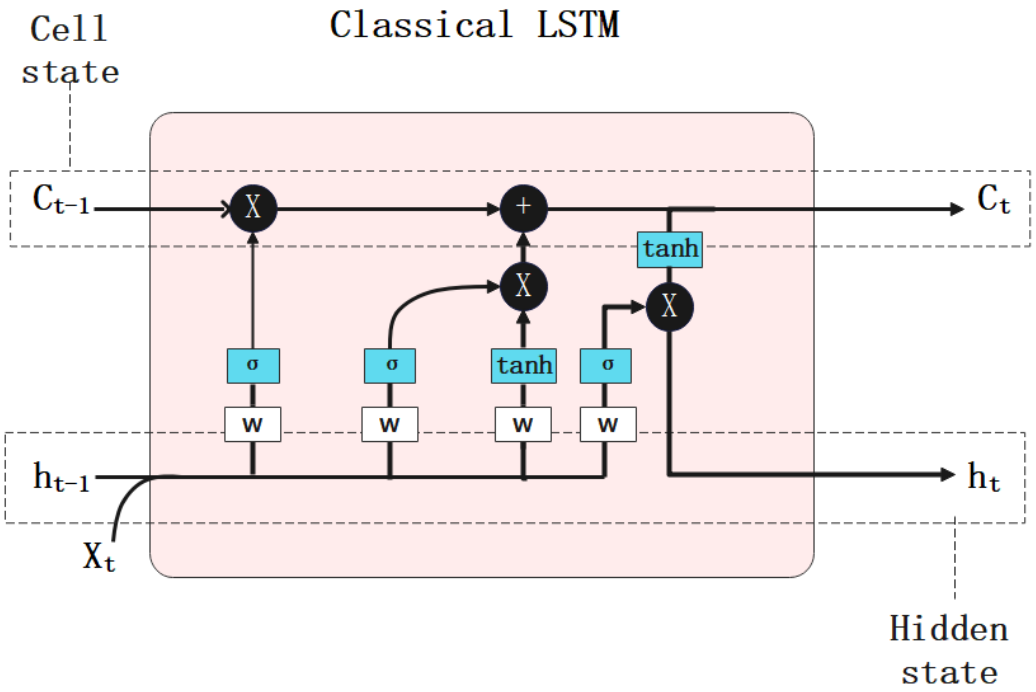

Figure 2. Structure of a single unit of classical LSTM.

An LSTM unit is composed of a cell, an input gate, an output gate, and a forget gate. These components work together to regulate the flow of information into and out of the cell, and to decide which information to store and which to discard.

The key equations governing the operations within an LSTM unit are as follows:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f), \tag{14}$$
$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i), \tag{15}$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C), \tag{16}$$
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t, \tag{17}$$
$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o), \tag{18}$$
$$h_t = o_t * \tanh(C_t). \tag{19}$$

Here, f_t, i_t, and o_t represent the activations of the forget, input, and output gates, respectively, at time t. σ denotes the sigmoid function, and $*$ represents element-wise multiplication. C_t is the cell state at time t, and h_t is the hidden state. W and b are the weights and biases associated with the respective gates and cell state updates.

3.3. Gated Recurrent Units (GRU)

Gated Recurrent Unit (GRU) is a variation of recurrent neural networks that aims to solve the vanishing gradient problem, similar to LSTM. GRUs simplify the architecture seen in LSTM by combining certain gates and states, which often results in more efficient training for certain types of problems. The structure of a single GRU is shown in Figure 3.

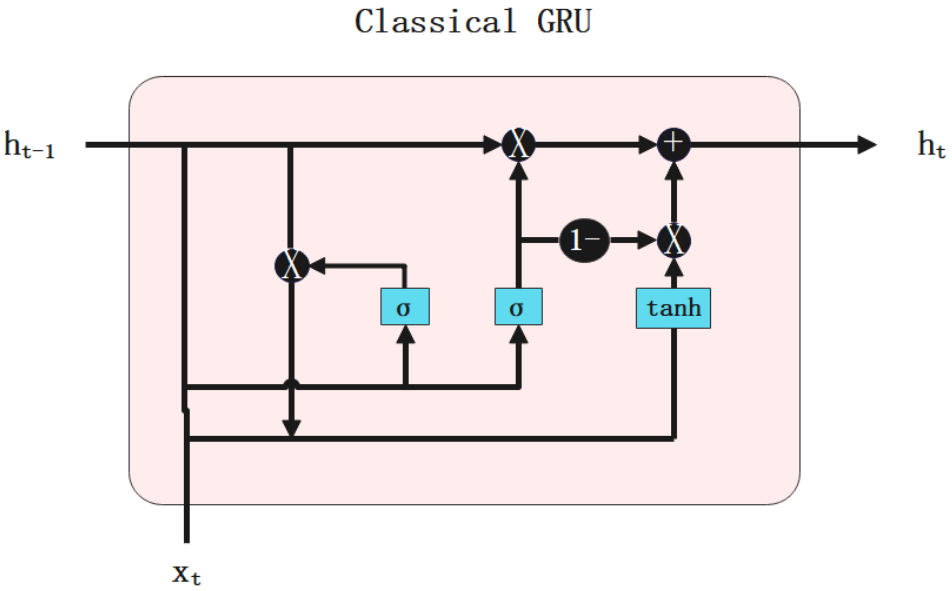

Figure 3. Structure of a single unit of classical GRU.

The GRU architecture is built around two gates: the update gate and the reset gate. These gates determine how much of the past information needs to be passed along to the future. The key equations defining a GRU are:

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t] + b_z), \tag{20}$$

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t] + b_r), \tag{21}$$

$$\tilde{h}_t = \tanh(W \cdot [r_t * h_{t-1}, x_t] + b), \tag{22}$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t. \tag{23}$$

In Equations (20)–(23), z_t and r_t represent the activations of the update and reset gates at time t, respectively. h_t is the hidden state at time t, and $\tilde{h}_t$ is the candidate hidden state. W and b are the weights and biases associated with the respective gates and hidden state updates. The symbol $*$ denotes element-wise multiplication, and σ represents the sigmoid activation function.

GRUs provide an efficient alternative to LSTM and are particularly useful in modeling sequences where LSTM's complex structure may not be necessary.

We need to consider the hardware realization of two activation functions here, which are the *tanh* and *sigmoid* functions. On classical hardware, *tanh* and *sigmoid* functions are computed directly, introducing nonlinearity in neural networks crucial for learning complex patterns:

$$\tanh(x) = \frac{e^{2x} - 1}{e^{2x} + 1}, \tag{24}$$

$$\sigma(x) = \frac{1}{1 + e^{-x}}. \tag{25}$$

Classical computers execute these operations efficiently using their processing capabilities. However, quantum hardware employs a different approach due to its linear operation nature, making direct computation of *tanh* and *sigmoid* functions non-trivial. The encoding and variational layers in the VQCs allow the quantum-based models to catch the nonlinear trends across data inputs. Additionally, the measurement layer outputs values within the range of $[-1, 1]$, which can then be processed classically to implement these nonlinear functions.

3.4. Quantum Long Short-Term Memory (QLSTM)

Quantum Long Short-Term Memory (QLSTM) is a quantum-enhanced version of the traditional LSTM networks. QLSTM integrates VQCs into the LSTM architecture, aiming to leverage the computational advantages of quantum mechanics. The structure of a single QLSTM unit is shown in Figure 4.

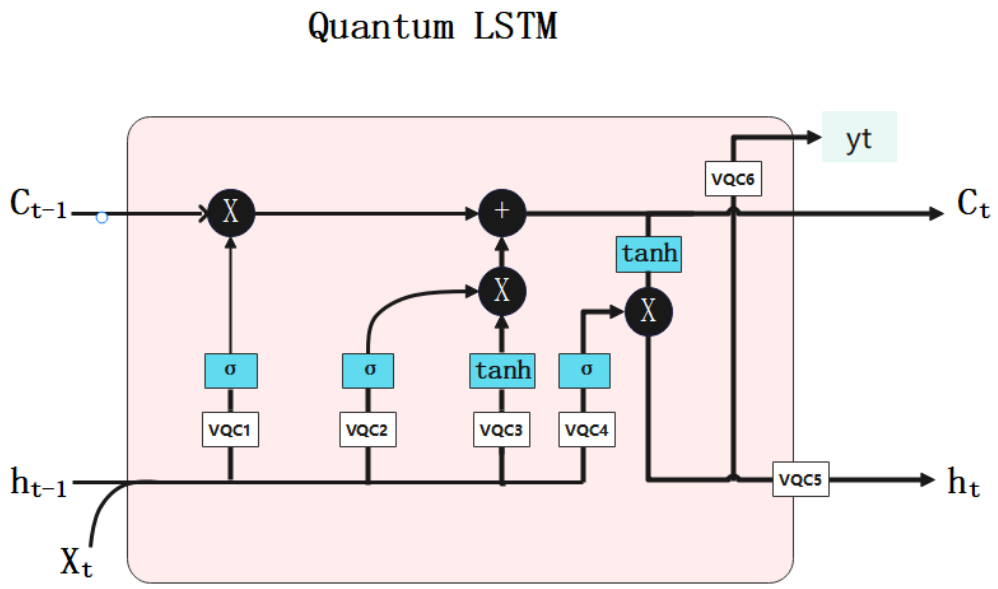

Figure 4. Structure of a single unit of QLSTM.

In a QLSTM, two key memory components are present: the hidden state h_t and the cell or internal state c_t. The functioning of a QLSTM cell can be mathematically described by the following equations:

$$f_t = \sigma(\text{VQC1}(v_t)), \tag{26}$$

$$i_t = \sigma(\text{VQC2}(v_t)), \tag{27}$$

$$\tilde{C}_t = \tanh(\text{VQC3}(v_t)), \tag{28}$$

$$c_t = f_t * c_{t-1} + i_t * \tilde{C}_t, \tag{29}$$

$$o_t = \sigma(\text{VQC4}(v_t)), \tag{30}$$

$$h_t = \text{VQC5}(o_t * \tanh(c_t)), \tag{31}$$

$$\tilde{y}_t = \text{VQC6}(o_t * \tanh(c_t)), \tag{32}$$

$$y_t = \text{NN}(\tilde{y}_t). \tag{33}$$

In Equations (26)–(33), σ represents the sigmoid function, and $*$ denotes element-wise multiplication. The input to the QLSTM cell at each time step is the concatenation v_t of the

previous hidden state h_{t-1} and the current input vector x_t. The VQCs mentioned in the equations refer to Variational Quantum Circuits.

3.5. Quantum Gated Recurrent Unit (QGRU)

Quantum Gated Recurrent Unit (QGRU) represents an evolution of traditional GRU networks, integrating with VQCs. The structure of a single QGRU unit is shown in Figure 5.

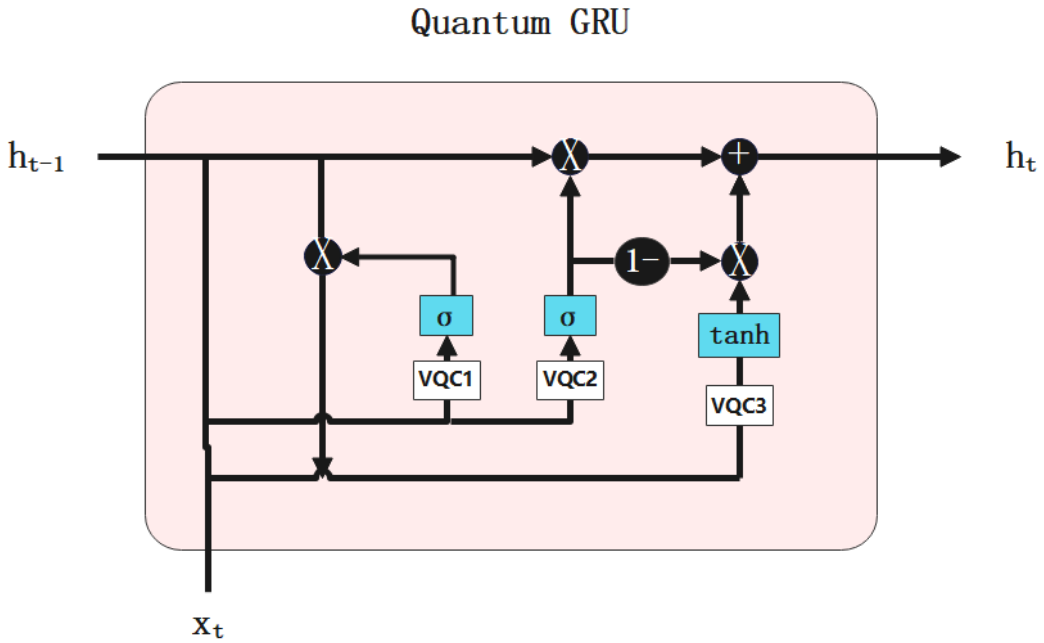

Figure 5. Structure of a single unit of QGRU.

A QGRU cell operates based on the following equations:

$$r_t = \sigma(\text{VQC1}(v_t)), \tag{34}$$
$$z_t = \sigma(\text{VQC2}(v_t)), \tag{35}$$
$$o_t = \text{cat}(x_t, r_t * H_{t-1}), \tag{36}$$
$$\tilde{H}_t = \tanh(\text{VQC3}(o_t)), \tag{37}$$
$$H_t = z_t * H_{t-1} + (1 - z_t) * \tilde{H}_t, \tag{38}$$
$$y_t = \text{NN}(H_t). \tag{39}$$

In Equations (34)–(39), r_t and z_t represent the reset and update gates of the QGRU at time t, respectively. H_t denotes the hidden state, and $\tilde{H}_t$ is the candidate hidden state. The input to the QGRU cell, v_t, is the concatenation of the previous hidden state H_{t-1} and the current input vector x_t. The VQCs are used to process the quantum aspects of the data.

4. Numerical Experiments

4.1. Hyperparameter Configuration

For the numerical experiments conducted in this study, specific hyperparameters were chosen to optimize the performance of the quantum models. Table 1 outlines the hyperparameter configuration used in the Van der Pol oscillator simulation and the simulation of two coupled damped harmonic oscillators:

More specifically, we are using a single-layer model with hidden size 4, and the models are evaluating over 100 epochs. We are using the backend *default.qubit* by PennyLane [16].

Table 1. Hyperparameter configuration.

Hyperparameter	QRNNs Value	RNNs Value
Optimizer	RMSprop	Adam
Loss Function	MSE	MSE
Backend	default.qubit	-
Number of Qubits	4	-
Layer of Entanglements	4	-
Number of Data Points	250	250
Percentage of Train Set	67%	67%
Percentage of Test Set	33%	33%
Learning Rate	0.01	0.01

4.2. Van der Pol Oscillator Simulation

In the first experiment, we consider the Van der Pol oscillator, a classical example of a non-conservative oscillator with nonlinear damping. The oscillator is modeled by the following second-order differential equation:

$$\frac{d^2x}{dt^2} - \mu(1 - x^2)\frac{dx}{dt} + x = 0. \tag{40}$$

The parameter μ, representing the nonlinearity and strength of the damping, is set to 1.0 in our simulations. This choice of μ provides a balance between linear and nonlinear dynamical behaviors, also making the system ideal for predicting a wide range of oscillatory patterns.

The initial conditions for the oscillator are chosen as $x_0 = 2.0$ and $y_0 = 0.0$, where x_0 and y_0 represent the initial position and velocity, respectively.

To numerically solve the Van der Pol differential equation, we convert it into a system of first-order equations:

$$\begin{aligned} \frac{dx}{dt} &= y, \\ \frac{dy}{dt} &= \mu(1 - x^2)y - x. \end{aligned} \tag{41}$$

The numerical solution is obtained over a time span of 0 to 50 s, discretized into 250 time steps.

Following the numerical solution of the differential equations, the resultant time series data of x and y are normalized and prepared for analysis. The data are reshaped to fit the input requirements of RNN models, enabling us to predict the Van der Pol oscillator's dynamics.

Simulation Results for the Van der Pol Oscillator

The behavior of the Van der Pol oscillator was examined through the LSTM, QLSTM, GRU, and QGRU models by evaluating their performance on the Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) metrics.

Mean Absolute Error (MAE) is a measure of errors between paired observations. It is calculated as the average of the absolute differences between the predicted values and the actual values, disregarding the direction of the error. The MAE is given by the formula:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|. \tag{42}$$

Root Mean Square Error (RMSE) is a quadratic scoring rule that also measures the average magnitude of the error. It is the square root of the average of squared differences between prediction and actual observation. The RMSE is given by the formula:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}, \tag{43}$$

where:

- n is the number of observations;
- y_i is the actual value of the ith observation;
- $\hat{y}_i$ is the predicted value of the ith observation.

Data from the numerical solution of the Van der Pol differential equations formed the basis for training and testing these models. A comparative analysis of the models' performance is summarized in Tables 2 and 3.

Table 2. Comparison of train and test MAE and RMSE for the state of x.

Model	Train MAE	Test MAE	Train RMSE	Test RMSE
LSTM	0.2601	0.2585	0.3002	0.2986
QLSTM	0.1411	0.1397	0.1648	0.1641
GRU	0.1597	0.1591	0.1845	0.1850
QGRU	0.0868	0.0902	0.1013	0.1031

Table 3. Comparison of train and test MAE and RMSE for the state of y.

Model	Train MAE	Test MAE	Train RMSE	Test RMSE
LSTM	0.3224	0.3294	0.3737	0.3828
QLSTM	0.1959	0.1851	0.2498	0.2384
GRU	0.3336	0.3375	0.4319	0.4384
QGRU	0.1473	0.1500	0.1931	0.1943

Table 2 shows that quantum-based models exhibit superior predictive performance on x value over classical models. Among the models compared, the GRU models also outshine the LSTM models in accuracy and QGRU model gives us the most accurate predictions.

Table 3 compares model performances on the y value. It is very interesting to see that the models produced more error on the prediction of the y values compared to the x values. Moreover, GRU models surpass LSTM models in terms of results, with the QGRU model achieving the highest accuracy among the evaluated models.

The example predictive results over epochs by the different models are shown in Figures 6–10.

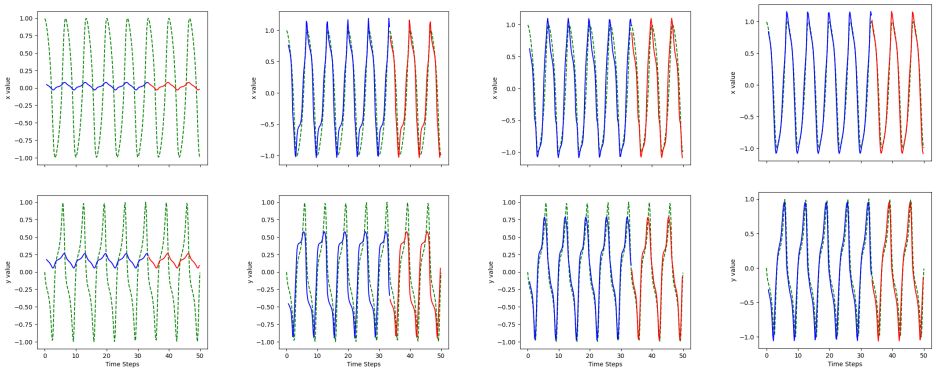

Figure 6. LSTM predictions over 100 epochs for the Van der Pol oscillator. Up: x; down: y; epoch: 5, 50, 70, 100; green dashed line: actual values; blue solid line: training predictions; red solid line: testing predictions.

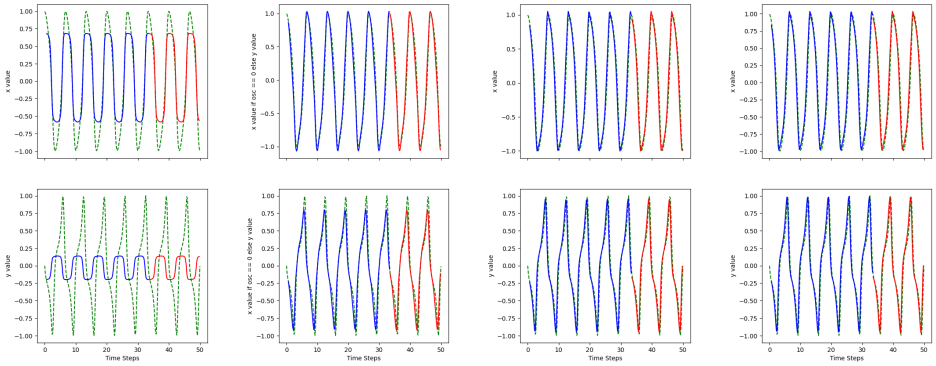

Figure 7. QLSTM predictions over 100 Epochs for the Van der Pol oscillator. Up: x; down: y; epoch: 5, 50, 70, 100; green dashed line: actual values; blue solid line: training predictions; red solid line: testing predictions.

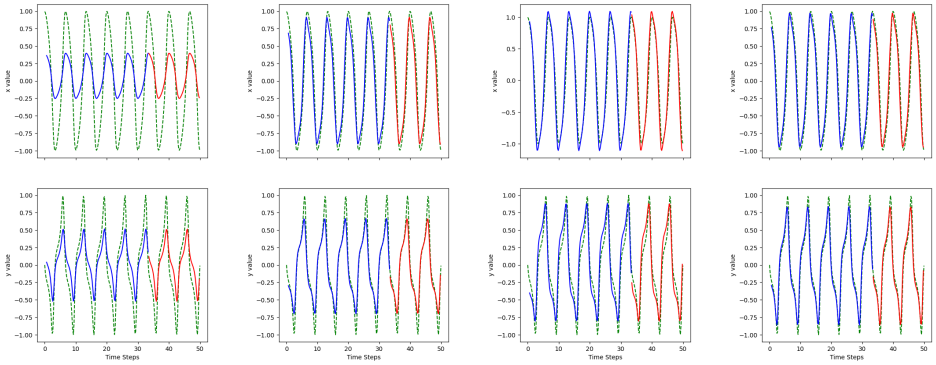

Figure 8. GRU predictions over 100 Epochs for the Van der Pol oscillator. Up: x; down: y; epoch: 5, 50, 70, 100; green dashed line: actual values; blue solid line: training predictions; red solid line: testing predictions.

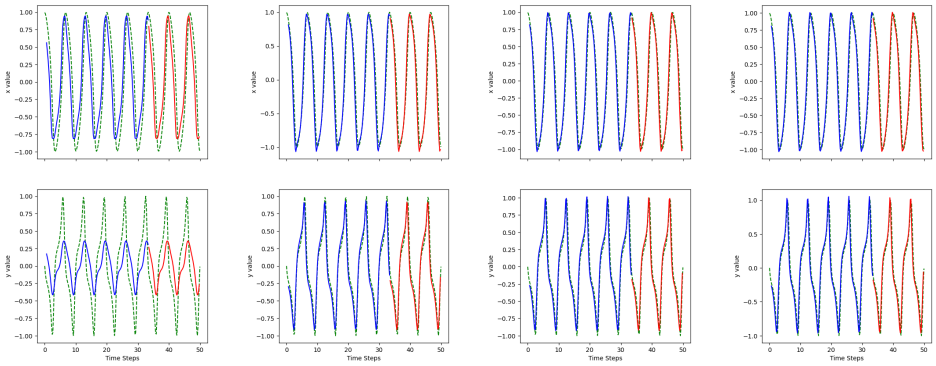

Figure 9. QGRU predictions over 100 Epochs for the Van der Pol oscillator. Up: x; down: y; epoch: 5, 50, 70, 100; green dashed line: actual values; blue solid line: training predictions; red solid line: testing predictions.

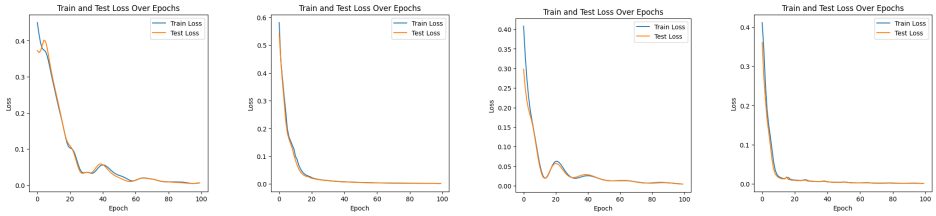

Figure 10. Experiment 1: Train and test losses for the models over 100 epochs. From left to right: LSTM, QLSTM, GRU, QGRU.

It is observable that the Van der Pol oscillator exhibits periodic behavior. It can be seen that all the models caught the patterns well. It can be observed that the QRNN models, especially, learn significantly faster than the classical RNN models from the comparison through epoch 5. Again, the QGRU model shows great capability in learning the dynamics.

By comparing the training and test loss over the epochs, the QRNN models show more stable decrease than the classical RNN models and the QGRU converges faster than QLSTM.

4.3. Two Coupled Damped Harmonic Oscillators Simulation

In this experiment, we study the capability the QRNN in learning the patterns in the system of two coupled harmonic oscillators. It is worth pointing out that this experiment actually is an extension of the prediction on a single damped harmonic oscillator by Chen [34,35]. This experiment extends our understanding not only from a single oscillator, but physical systems by introducing interactions between oscillatory motions. Such systems are paramount in numerous fields, including physics and engineering. The governing differential equations for two coupled damped harmonic oscillators, representing an extended model of the single oscillator, are as follows:

$$\begin{aligned} \frac{d^2\theta_1}{dt^2} &= -\frac{b_1}{m_1}\frac{d\theta_1}{dt} - \frac{g}{l_1}\sin(\theta_1) + \frac{k_c}{m_1}(\theta_2 - \theta_1), \\ \frac{d^2\theta_2}{dt^2} &= -\frac{b_2}{m_2}\frac{d\theta_2}{dt} - \frac{g}{l_2}\sin(\theta_2) + \frac{k_c}{m_2}(\theta_1 - \theta_2). \end{aligned} \tag{44}$$

Here, θ_1 and θ_2 represent the angular displacements of the first and second pendulums, respectively. The constants $g = 9.81$ m/s^2 (gravitational acceleration), $b_1 = b_2 = 0.15$ (damping factors), $l_1 = l_2 = 1.0$ m (lengths of the pendulums), $m_1 = m_2 = 1.0$ kg (masses of the pendulums), and $k_c = 0.05$ (coupling constant) define the system's characteristics. The initial conditions are set with angular displacements $\theta_1 = \theta_2 = 0$ and angular velocities $\dot{\theta_1} = 3.0$, $\dot{\theta_2} = 0.0$ rad/s.

Simulation Results for the Two Coupled Damped Harmonic Oscillators

Just like the Van der Pol example, we present a comparative analysis of the models' performance in predicting the dynamic behavior of both oscillators (again, with MAE and RMSE), which are summarized in Tables 4 and 5.

Table 4. Comparison of train and test MAE and RMSE for Oscillator 1.

Model	Train MAE	Test MAE	Train RMSE	Test RMSE
LSTM	0.5467	0.4371	0.7761	0.4922
QLSTM	0.5284	0.4763	0.6987	0.5374
GRU	0.3648	0.4396	0.4499	0.4941
QGRU	0.2585	0.2411	0.3587	0.2701

Table 5. Comparison of train and test MAE and RMSE for Oscillator 2.

Model	Train MAE	Test MAE	Train RMSE	Test RMSE
LSTM	0.2190	0.3597	0.2814	0.4248
QLSTM	0.1244	0.2591	0.1607	0.2885
GRU	0.1160	0.0858	0.1483	0.1006
QGRU	0.0794	0.0482	0.0949	0.0602

Like the results from the first experiment, in Tables 4 and 5 all the models successfully learn the dynamics, but the QGRU shows excellent predictive results, especially on the prediction of Oscillator 2. We noticed, first, that the predictions on Oscillator 2 are more accurate than Oscillator 1. Second, we noticed that the GRU models outshine LSTM models in general.

The example predictive results over the 100 epochs by the different models are shown in Figures 11–15.

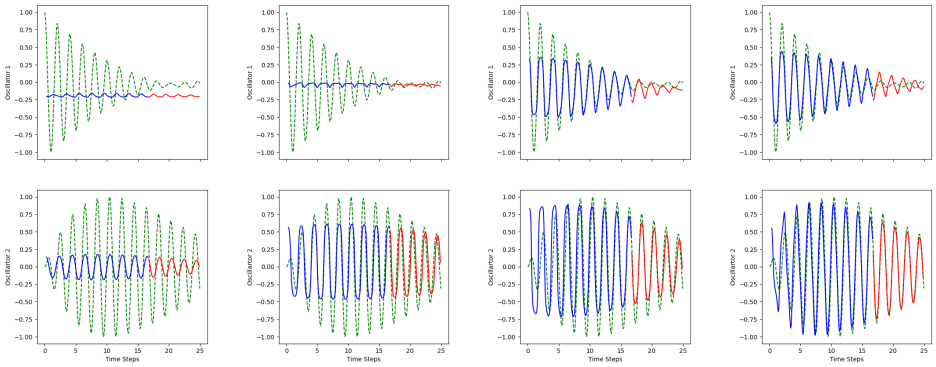

Figure 11. LSTM predictions over 10 epochs. Up: Oscillator 1; down: Oscillator 2; epoch: 5, 50, 70, 100; green dashed line: actual values; blue solid line: training predictions; red solid line: testing predictions.

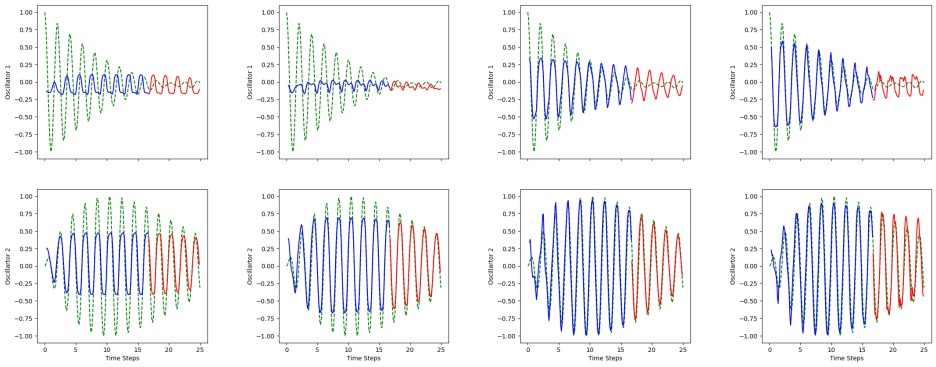

Figure 12. QLSTM Predictions over 10 epochs. Up: Oscillator 1; down: Oscillator 2; epoch: 5, 50, 70, 100; green dashed line: actual values; blue solid line: training predictions; red solid line: testing predictions.

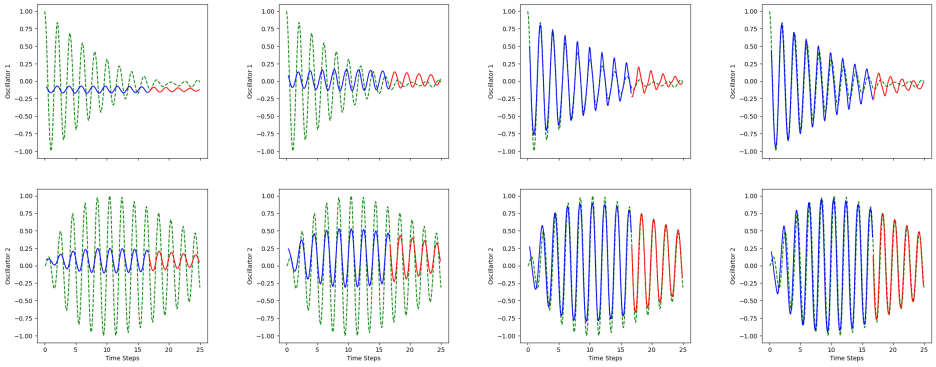

Figure 13. GRU Predictions over 10 epochs. Up: Oscillator 1; down: Oscillator 2; epoch: 5, 50, 70, 100; green dashed line: actual values; blue solid line: training predictions; red solid line: testing predictions.

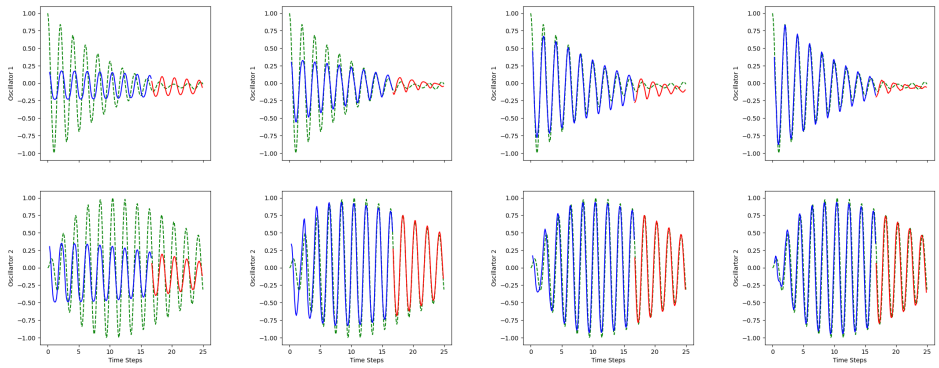

Figure 14. QGRU Predictions over 10 epochs. Up: Oscillator 1, down: Oscillator 2 epoch: 5, 50, 70, 100; green dashed line: actual values; blue solid line: training predictions; red solid line: testing predictions.

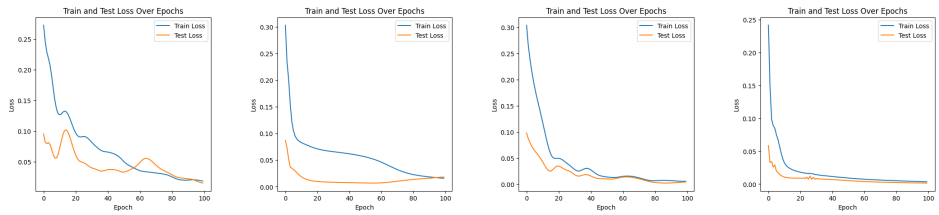

Figure 15. Experiment 2: train and test losses for the models over 100 epochs. From left to right: LSTM, QLSTM, GRU, QGRU.

Similar to the Van der Pol case, the GRU models converge faster than the LSTM models and the quantum-based models learn the pattern faster than the classical models (which can be seen by the comparison on the results on epoch 5). The quantum-based RNNs also show stabler decrease in loss. Here, we observe several points of interest: Firstly, there is a minor increase in the test loss in LSTM models from epoch 80 to 100, potentially indicative of mild overfitting. Although adjusting the learning rate could mitigate this, we maintain a constant learning rate to achieve a straightforward comparison between the models. Secondly, spikes are observed in the LSTM model's test loss. Furthermore, at epoch 5, the LSTM model exhibits an undershot in performance, which is not seen in the results of the other models.

From the previous two numerical experiments, the QRNN models demonstrate exceptional capability in learning patterns by system dynamics. In the last experiment, we apply the models to chaotic systems using a different approach.

4.4. System of Lorenz Equations

The Lorenz equations, fundamental in chaos theory, model the dynamics of atmospheric convection and are characterized by their chaotic nature for certain parameter values. The system is described by the following set of differential equations:

$$\begin{aligned} \frac{dx}{dt} &= \sigma(y - x), \\ \frac{dy}{dt} &= x(\rho - z) - y, \\ \frac{dz}{dt} &= xy - \beta z. \end{aligned} \tag{45}$$

In Equation (45), x, y, and z represent the system states, and the parameters σ, ρ, and β are crucial for the system's behavior. The chosen values $\sigma = 10.0$, $\beta = \frac{8}{3}$, and $\rho = 28.0$ are known to induce chaotic dynamics in the Lorenz system. These parameters represent the Prandtl number, normalized Rayleigh number, and certain physical dimensions of the convective cells, respectively. Their specific values make the system a classic example of

chaos, making it a compelling subject for studying the predictive capabilities of quantum-enhanced and classical neural networks. Particularly, the research on the utilization of classical neural networks is discussed in [49].

Data Preparation and Hyperparameter Configuration

Unlike the previous two experiments, we generated the data differently in the application of RNN models on Lorenz equations. We first randomly generated 10 datasets; we used 67% of them for training and the remaining 33% for testing. The shapes of the training and test datasets are as follows:

- Training dataset shapes:
  - Features: `torch.Size([3120, 10, 3])`;
  - Labels: `torch.Size([3120, 3])`.
- Testing dataset shapes:
  - Features: `torch.Size([780, 10, 3])`;
  - Labels: `torch.Size([780, 3])`.

Regarding the hyperparameter aspect, the previous two experiments demonstrate that QRNN models rapidly catch the trend of the dataset. Consequently, we aim to explore more about these models' proficiency in learning features with shorter sequence lengths, reduced data points, and fewer epochs.

We pick sequence length 10 for all the models and all the models will be running over 20 epochs instead of the 100 epochs in the previous two experiments.

Hyperparameter configurations for LSTM, QLSTM, GRU, and QGRU models are presented in Table 6:

Table 6. Hyperparameter configuration for classic models and quantum models.

Hyperparameter	Quantum	Classic
Optimizer	RMSprop	Adam
Loss Function	MSE	MSE
Backend	default.qubit	-
Number of Qubits	4	-
VQC Layer	4	-
Percentage of Train Set	67%	67%
Percentage of Test Set	33%	33%
Epochs	20	20
Learning Rate	0.01	0.01

4.5. Results Analysis

The performances of each model on the Lorenz system data are summarized in Tables 7 and 8. These tables detail the Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) for training and testing datasets across three dimensions.

According to the error metrics, quantum-based RNNs exhibit a marked increase in accuracy, particularly the QGRU model in comparison to the classical GRU. Moreover, the LSTM model demonstrates notably poorer performance compared to the other models, potentially caused by its slower rate of convergence across epochs (again, only 20 epochs were used in this case).

Table 7. Mean Absolute Error (MAE) for each model on test set.

Model	X	Y	Z
LSTM	2.0361	2.1232	2.2119
QLSTM	0.8820	0.9473	1.0206
GRU	1.1684	1.1719	1.1710
QGRU	0.4864	0.4723	0.4555

Table 8. Root Mean Square Error (RMSE) for each model on test set.

Model	X	Y	Z
LSTM	2.1736	2.2718	2.3714
QLSTM	1.2234	1.2723	1.3348
GRU	1.1783	1.1748	1.1740
QGRU	0.4971	0.4846	0.4745

To further present the models' predictions, we visualize their predictive outcomes across three dimensions, the trajectories and the losses over the epochs, in Figures 16–24.

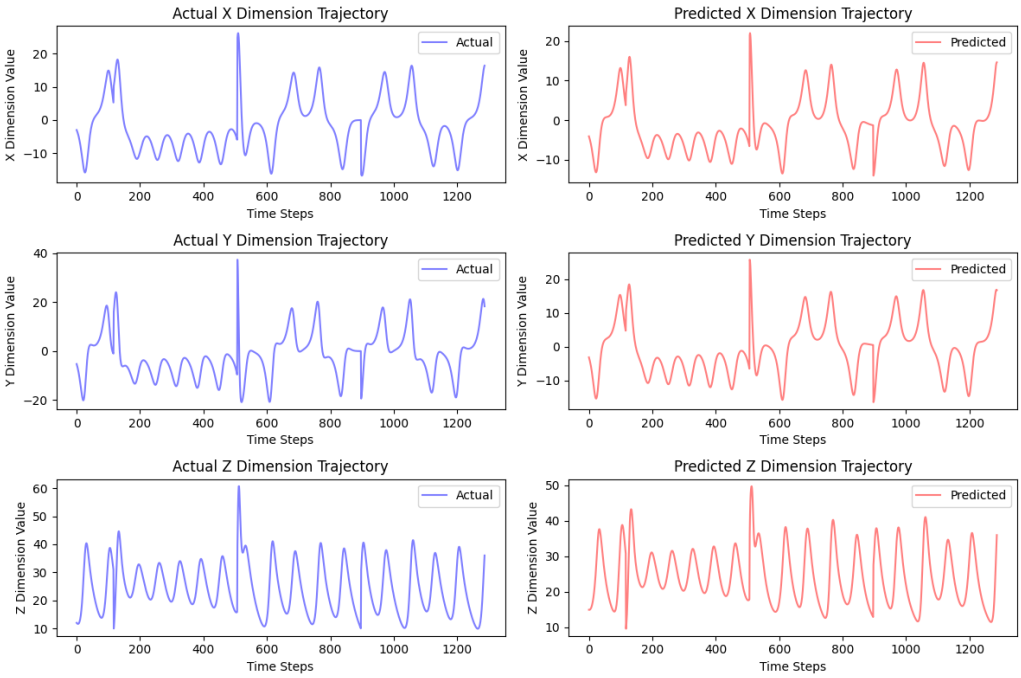

Figure 16. LSTM predictions for dimensions X (**up**), Y (**middle**), and Z (**bottom**).

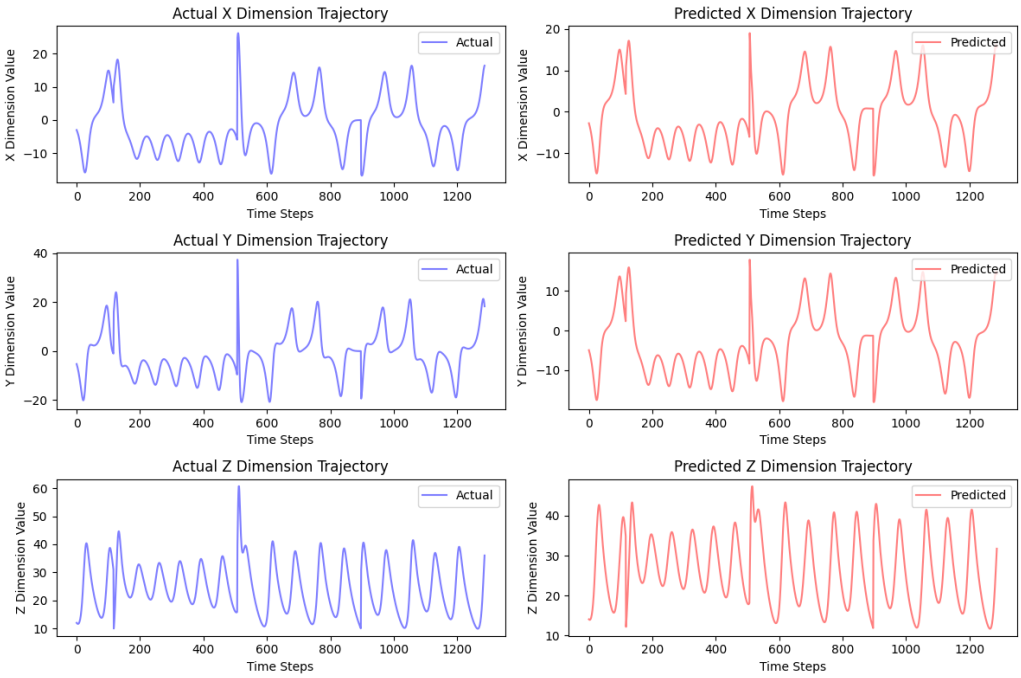

Figure 17. QLSTM predictions for dimensions X (**up**), Y (**middle**), and Z (**bottom**).

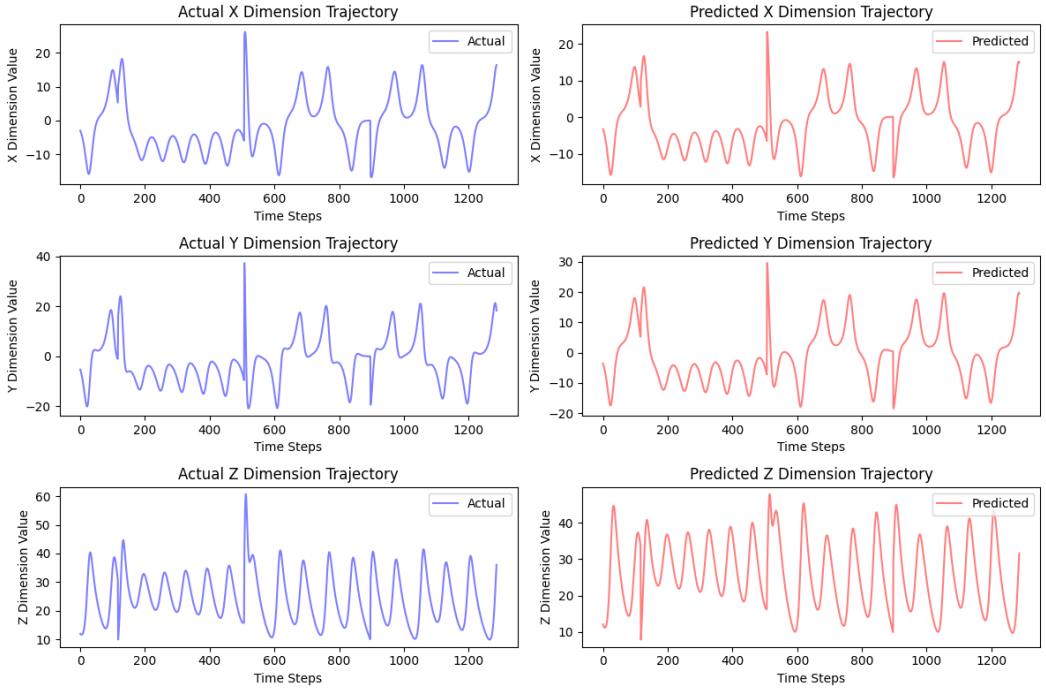

Figure 18. GRU predictions for dimensions X (**up**), Y (**middle**), and Z (**bottom**).

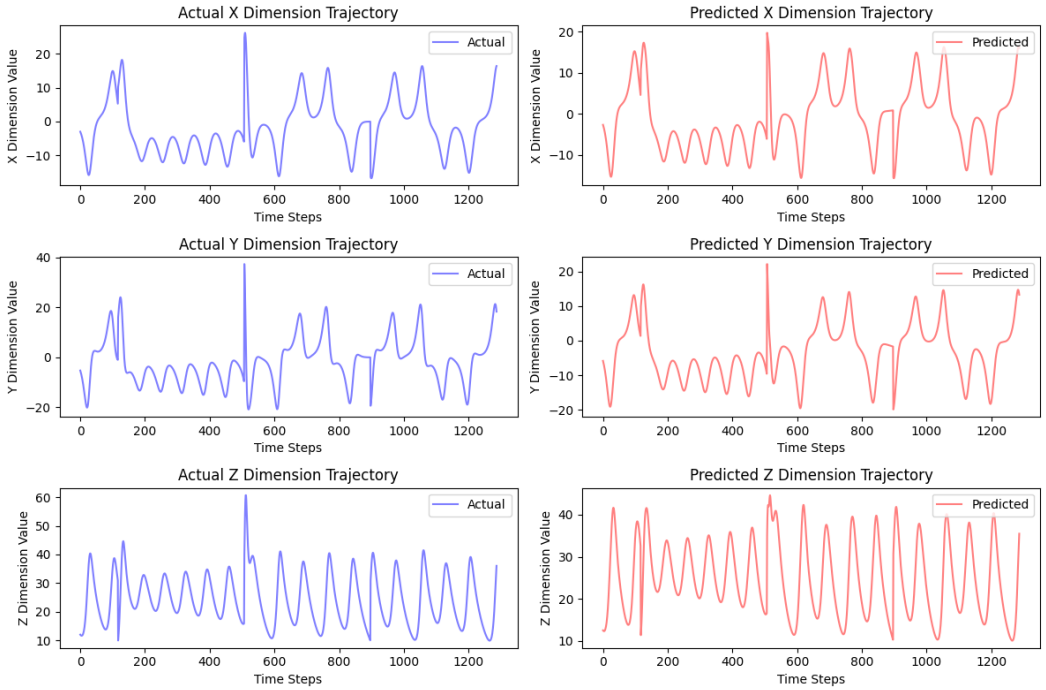

Figure 19. QGRU predictions for dimensions X (**up**), Y (**middle**), and Z (**bottom**).

Firstly, regarding to the three-dimensional predictions, it is evident that all models correctly capture the dataset's trend. Nonetheless, a closer look at the figure scales reveals difficulties in predicting the spike at the figure's center for all the models (for LSTM in y and z dimensions, QLSTM in z, GRU in y and z, and QGRU in z). We believe this challenge could be solved by increasing the number of epochs, though the performance is deemed satisfactory for the current study.

Furthermore, the QGRU outperforms other models in trajectory plotting, showing a consistent decline in both training and test losses across epochs. In contrast, the classical RNNs display slight fluctuations in their loss plots. Although a spike is observed in the

QLSTM's loss plot, quantum-based RNNs, in general, tend to show a smoother and more rapid reduction in losses than the classical models.

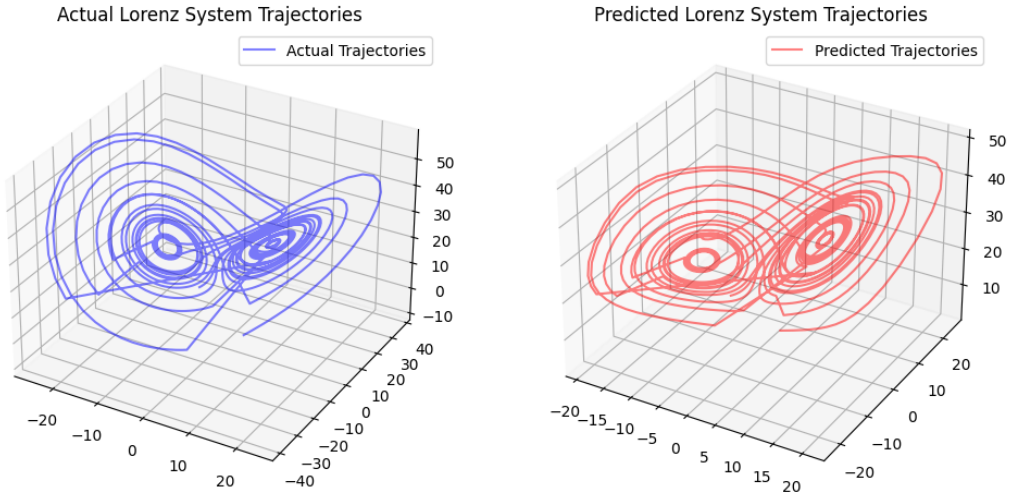

Figure 20. Predicted Lorenz system trajectory by LSTM model.

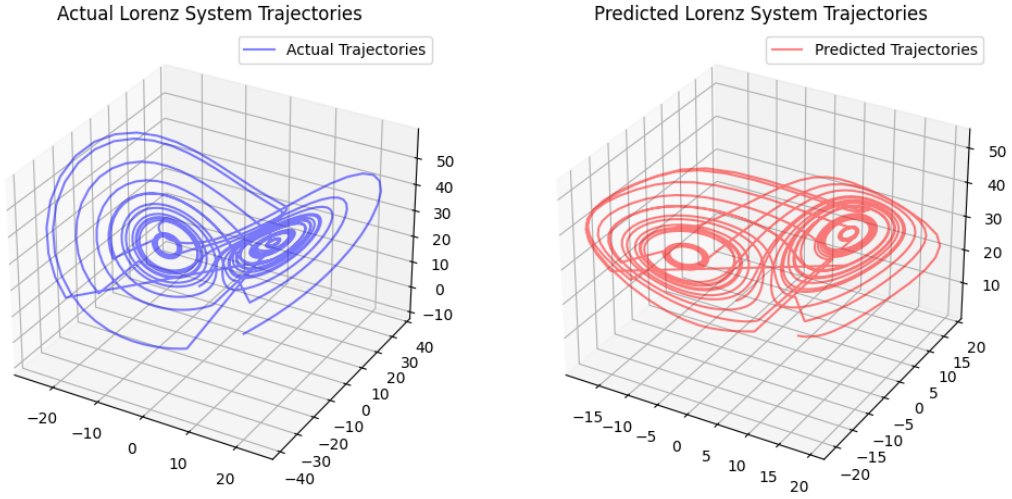

Figure 21. Predicted Lorenz system trajectory by QLSTM model.

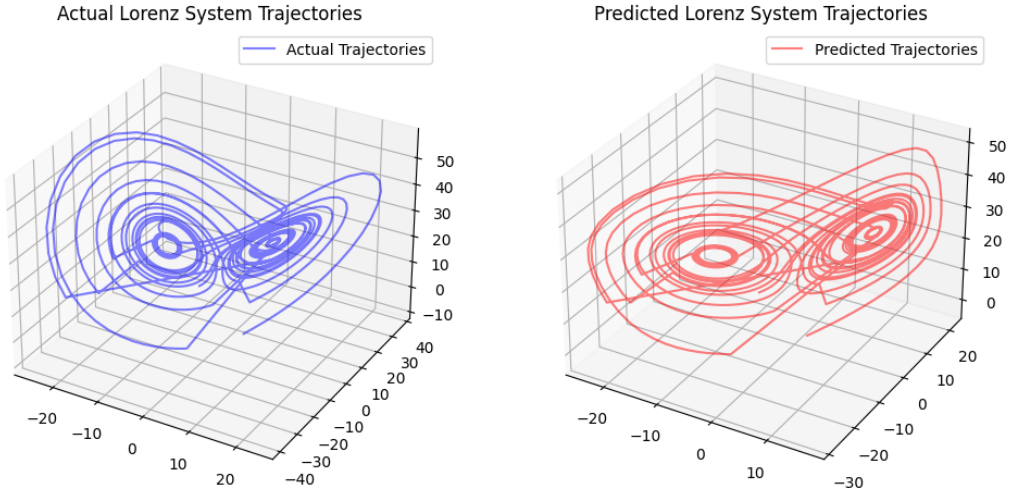

Figure 22. Predicted Lorenz system trajectory by GRU model.

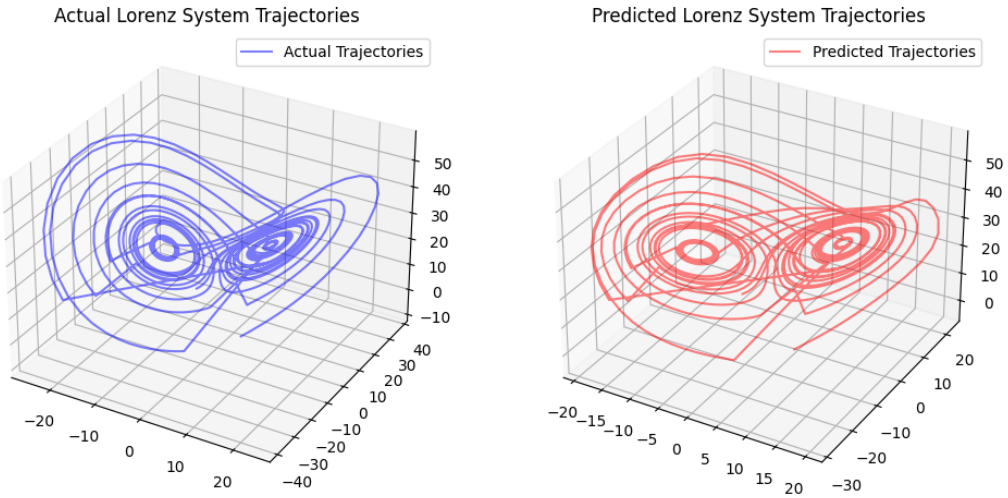

Figure 23. Predicted Lorenz system trajectory by QGRU model.

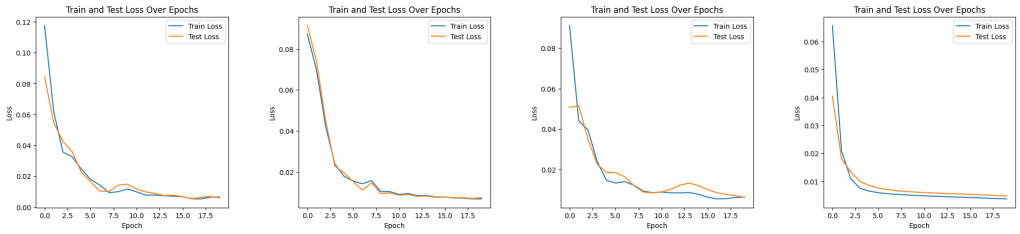

Figure 24. Experiment 3: Train and test losses for the models over 20 epochs. From left to right: LSTM, QLSTM, GRU, QGRU.

5. Discussion

This study explores the efficacy of quantum-enhanced models, specifically the quantum Long Short-Term Memory (QLSTM) and the Quantum Gated Recurrent Unit (QGRU), in predicting the dynamics of systems governed by ordinary differential equations. These systems include Van der Pol oscillators, two damped coupled oscillators, and the Lorenz system. We benchmarked the performance of these quantum models against traditional Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) models by evaluating their accuracy through MAE and RMSE, alongside observing their training and testing loss trajectories over epochs.

The results from the three numerical experiments showcase that quantum-based models, particularly the QGRU, significantly outperform their classical counterparts in terms of predictive accuracy. This discovery highlights the capability of quantum computing techniques to improve the predictive performance of recurrent neural networks, particularly in the complex dynamics of systems represented by ordinary differential equations. However, it is important to recognize that these quantum-based models require greater computational resources, primarily because the operations of the Variational Quantum Circuit (VQC) are simulated on classical computing hardware. The same challenge is observed across numerous studies in the QRNN field, as highlighted in [50,51]. To address these computational challenges and enhance efficiency, we implemented batch sizes of 64 in this study, which significantly accelerated the processing time. Additionally, a critical aim of our ongoing research is to test the performance of these models on actual quantum hardware. This will not only confirm their theoretical benefits but also evaluate their practical utility and efficiency within a real quantum computing context.

The example code for *classQLSTM* and *classQGRU* can be found in the following Github link: https://github.com/YuanChenmt/QRNN-Examples (accessed on 16 April 2024).

6. Conclusions

We have validated the superior performance of quantum-based recurrent neural networks (QRNNs), particularly the Quantum Gated Recurrent Unit (QGRU), over traditional RNN models in predicting the outcomes of systems described by ordinary differential equations in terms of model accuracy, efficiency, and stability. This study aims to contribute to the understanding of QRNNs' predictive capabilities, suggesting potential pathways for their application in complex, real-world scenarios. It modestly seeks to add to the ongoing discourse in computational science, hoping to inspire further research and exploration in the field.

Author Contributions: Conceptualization, Y.C.; methodology, Y.C. and A.K.; software, Y.C.; validation, A.K.; formal analysis, Y.C. and A.K.; investigation, A.K.; resources, A.K.; data curation, Y.C.; writing—original draft preparation, Y.C.; writing—review and editing, A.K.; visualization, Y.C.; supervision, A.K.; project administration, A.K.; funding acquisition, A.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: This paper used manual data, no new data were created.

Acknowledgments: We would like to express our sincere gratitude to the reviewers for their insightful comments and constructive feedback, which significantly improved the quality and clarity of this manuscript. Their expertise and valuable insights have been invaluable in shaping the direction of our research and enhancing the overall impact of our work.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Zakwan, M.; Di Natale, L.; Svetozarevic, B.; Heer, P.; Jones, C.N.; Ferrari Trecate, G. Physically Consistent Neural ODEs for Learning Multi-Physics Systems. *arXiv* **2022**, arXiv:2211.06130.
2. Städter, P.; Schälte, Y.; Schmiester, L.; Hasenauer, J.; Stapor, P.L. Benchmarking of numerical integration methods for ODE models of biological systems. *Sci. Rep.* **2021**, *11*, 2696. [CrossRef] [PubMed]
3. Yazdani, A.; Lu, L.; Raissi, M.; Karniadakis, G.E. Systems biology informed deep learning for inferring parameters and hidden dynamics. *PLoS Comput. Biol.* **2020**, *16*, e1007575. [CrossRef] [PubMed]
4. Jorge, M. An Application of Ordinary Differential Equations in Economics: Modeling Consumer's Preferences Using Marginal Rates of Substitution. *Math. Methods Sci. Mech.* **2014**, *33*. [CrossRef]
5. Bashforth, F.; Adams, J.C. *An Attempt to Test the Theories of Capillary Action by Comparing the Theoretical and Measured Forms of Drops of Fluid*; University Press: Cambridge, MA, USA, 17 October 2007; (1883) Paperback.
6. Dahlquist, G. A special stability problem for linear multistep methods. *BIT* **1963**, *3*, 27–43. [CrossRef]
7. Runge, C. Über die numerische Auflösung von Differentialgleichungen. *Math. Ann.* **1895**, 46, 167–178. [CrossRef]
8. Wilhelm, K. Beitrag zur näherungsweisen Integration totaler Differentialgleichungen. *Zeitschrift für Mathematik und Physik* **1901**, *46*, 435–453.
9. Sottas, G. *Rational Runge-Kutta Methods Are Not Suitable for Stiff Systems of ODE's*; Report SFB 123, Number 215; University of Heidelberg: Heidelberg, Germany, 1983.
10. Ahmed, D.M.; Hassan, M.M.; Mstafa, R.J. A Review on Deep Sequential Models for Forecasting Time Series Data. *Appl. Comput. Intell. Soft Comput.* **2022**, *2022*, 6596397. [CrossRef]
11. Lindemann, B.; Müller, T.; Vietz, H.; Jazdi, N.; Weyrich, M. A survey on long short-term memory networks for time series prediction. *Procedia CIRP* **2021**, *99*. [CrossRef]
12. Weerakody, P.B.; Wong, K.W.; Wang, G.; Ela, W. A review of irregular time series data handling with gated recurrent neural networks. *Neurocomputing* **2021**, *441*, 161–178. [CrossRef]
13. Sutskever, I.; Vinyals, O.; Le, Q.V. Sequence to sequence learning with neural networks. In Proceedings of the Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, Montreal, QC, Canada, 8–13 December 2014; pp. 3104–3112.
14. Graves, A.; Jaitly, N.; Mohamed, A.-R. Hybrid speech recognition with deep bidirectional LSTM. In Proceedings of the 2013 IEEE Workshop on Automatic Speech Recognition and Understanding, Olomouc, Czech Republic, 8–12 December 2013; pp. 273–278.
15. Flurin, E.; Martin, L.S.; Hacohen-Gourgy, S.; Siddiqi, I. Using a recurrent neural network to reconstruct quantum dynamics of a superconducting qubit from physical observations. *Phys. Rev. X* **2020**, *10*, 011006. [CrossRef]

16. August, M.; Ni, X. Using recurrent neural networks to optimize dynamical decoupling for quantum memory. *Phys. Rev. A* **2017**, *95*, 012335. [CrossRef]
17. Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef] [PubMed]
18. Chung, J.; Gulcehre, C.; Cho, K.; Bengio, Y. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv* **2014**, arXiv:1412.3555.
19. Gajamannage, K.; Jayathilake, D.I.; Park, Y.; Bollt, E.M. Recurrent neural networks for dynamical systems: Applications to ordinary differential equations, collective motion, and hydrological modeling. *Chaos* **2023**, *33*, 013109. [CrossRef] [PubMed]
20. Bai, S.; Kolter, J.Z.; Koltun, V. An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling. *arXiv* **2018**, arXiv:1803.01271.
21. Fu, Y.; Saab, S., Jr.; Ray, A.; Hauser, M. A Dynamically Controlled Recurrent Neural Network for Modeling Dynamical Systems. *arXiv* **2019**, arXiv:1911.00089.
22. Niu, M.Y.; Horesh, L.; Chuang, I. Recurrent Neural Networks in the Eye of Differential Equations. *arXiv* **2019**, arXiv:1904.12933.
23. de la Fraga, L.G.; Ovilla-Martínez, B.; Tlelo-Cuautle, E. Echo state network implementation for chaotic time series prediction. *Microprocess. Microsyst.* **2023**, *103*, 104950. [CrossRef]
24. Prokhorov, D. Echo state networks: Appeal and challenges. In Proceedings of the 2005 IEEE International Joint Conference on Neural Networks, Montreal, QC, Canada, 31 July–4 August 2005; Volume 3, pp. 1463–1466. [CrossRef]
25. Vlachas, P.R.; Byeon, W.; Wan, Z.Y.; Sapsis, T.P.; Koumoutsakos, P. Data-driven forecasting of high-dimensional chaotic systems with long short-term memory networks. *Proc. R. Soc. A* **2018**, *474*, 20170844. [CrossRef] [PubMed]
26. Meng, X.; Yang, T. Entanglement-Structured LSTM Boosts Chaotic Time Series Forecasting. *Entropy* **2021**, *23*, 1491. [CrossRef] [PubMed]
27. Cross, A. The IBM Q experience and Qiskit open-source quantum computing software. *APS Meet. Abstr.* **2018**, *2018*, L58.
28. Arute, F.; Arya, K.; Babbush, R.; Bacon, D.; Bardin, J.C.; Barends, R.; Biswas, R.; Boixo, S.; Brandao, F.G.S.L.; Buell, D.A.; et al. Quantum supremacy using a programmable superconducting processor. *Nature* **2019**, *574*, 505–510. [CrossRef] [PubMed]
29. Bergholm, V.; Izaac, J.; Schuld, M.; Gogolin, C.; Ahmed, S.; Ajith, V.; Alam, M.S.; Alonso-Linaje, G.; Narayanan, B.A.; Asadi, A.; et al. PennyLane: Automatic differentiation of hybrid quantum-classical computations. *arXiv* **2022**, arXiv:1811.04968.
30. Lanting, T.; Przybysz, A.J.; Smirnov, A.Y.; Spedalieri, F.M.; Amin, M.H.; Berkley, A.J. Entanglement in a quantum annealing processor. *Phys. Rev. X* **2014**, *4*, 021041. [CrossRef]
31. Preskill, J. Quantum computing in the NISQ era and beyond. *Quantum* **2018**, *2*, 79. [CrossRef]
32. Gottesman, D. Stabilizer codes and quantum error correction. *arXiv* **1997**, arXiv:quant-ph/9705052.
33. Gottesman, D. Theory of fault-tolerant quantum computation. *Phys. Rev. A* **1998**, *57*, 127. [CrossRef]
34. Mitarai, K.; Negoro, M.; Kitagawa, M.; Fujii, K. Quantum circuit learning. *Phys. Rev. A* **2018**, *98*, 032309. [CrossRef]
35. Du, Y.; Hsieh, M.-H.; Liu, T.; Tao, D. The expressive power of parameterized quantum circuits. *arXiv* **2018**, arXiv:1810.11922.
36. Qi, J.; Yang, C.-H.; Chen, P.-Y. QTN-VQC: An end-to-end learning framework for quantum neural networks. *Phys. Scr.* **2024**, *99*, 015111. [CrossRef]
37. Schuld, M.; Bocharov, A.; Svore, K.; Wiebe, N. Circuit-centric quantum classifiers. *arXiv* **2018**, arXiv:1804.00633.
38. Havlíček, V.; Córcoles, A.D.; Temme, K.; Harrow, A.W.; Kandala, A.; Chow, J.M.; Gambetta, J.M. Supervised learning with quantum-enhanced feature spaces. *Nature* **2019**, *567*, 209–212. [CrossRef] [PubMed]
39. Dallaire-Demers, P.-L.; Killoran, N. Quantum generative adversarial networks. *Phys. Rev. A* **2018**, 98, 012324. [CrossRef]
40. Chen, S.Y.-C.; Yang, C.-H.H.; Qi, J.; Chen, P.-Y.; Ma, X.; Goan, H.-S. Variational quantum circuits for deep reinforcement learning. *IEEE Access* **2020**, *8*, 141007–141024. [CrossRef]
41. Schuld, M.; Petruccione, F. *Supervised Learning with Quantum Computers*; Springer: Berlin/Heidelberg, Germany, 2018; Volume 17.
42. Chen, S.Y.-C.; Yoo, S. Federated Quantum Machine Learning. *Entropy* **2021**, *23*, 460. [CrossRef] [PubMed]
43. Wu, S.L.; Chan, J.; Guan, W.; Sun, S.; Wang, A.; Zhou, C.; Livny, M.; Carminati, F.; Di Meglio, A.; Li, A.C.Y. Application of quantum machine learning using the quantum variational classifier method to high energy physics analysis at the LHC on IBM quantum computer simulator and hardware with 10 qubits. *J. Phys. Nucl. Part. Phys.* **2021**, *48*, 12. [CrossRef]
44. Biamonte, J.; Wittek, P.; Pancotti, N.; Rebentrost, P.; Wiebe, N.; Lloyd, S. Quantum Machine Learning. *Nature* **2017**, *549*, 195–202. [CrossRef] [PubMed]
45. Dunjko, V.; Briegel, H.J. Machine Learning & Artificial Intelligence in the Quantum Domain: A Review of Recent Progress. *Rep. Prog. Phys.* **2018**, *81*, 074001. [PubMed]
46. Zaman, K.; Marchisio, A.; Hanif, M.A.; Shafique, M. A Survey on Quantum Machine Learning: Current Trends, Challenges, Opportunities, and the Road Ahead. *arXiv* **2023**, arXiv:2310.10315.
47. Chen, S.Y.-C.; Yoo, S.; Fang, Y.-L.L. *Quantum Long Short-Term Memory*; Computational Science Initiative, Brookhaven National Laboratory: Upton, NY, USA, 2020.
48. Chen, S.Y.-C.; Fry, D.; Deshmukh, A.; Rastunkov, V.; Stefanski, C. Reservoir Computing via Quantum Recurrent Neural Networks. *arXiv* **2020**, arXiv:2211.02612v1.
49. Malvern, M.; Gibbons, T.E. Learning and Modeling Chaos Using LSTM Recurrent Neural Networks. 2018. Available online: https://api.semanticscholar.org/CorpusID:212631658 (accessed on 3 March 2024).

50. Khan, S.Z.; Muzammil, N.; Zaidi, S.M.H.; Aljohani, A.J.; Khan, H.; Ghafoor, S. Quantum Long Short-Term Memory (QLSTM) vs Classical LSTM in Time Series Forecasting: A Comparative Study in Solar Power Forecasting. *arXiv* **2023**, arXiv:2310.17032v2.
51. Siemaszko, M.; Buraczewski, A.; Le Saux, B. Rapid training of quantum recurrent neural networks. *Quantum Mach. Intell.* **2023**, *5*, 31. [CrossRef]