UNIVERSITY OF STUTTGART

MASTER THESIS

Quantum Machine Learning for Time Series Prediction

Author:
Tobias Fellner

Supervisors:
Dr. David Kreplin
Samuel Tovey

First Examiner:
Prof. Dr. Christian Holm
Second Examiner:
Prof. Dr. Jörg Main
A thesis submitted in fulfillment of the requirements for the degree of Master of Science

in the

Institute for Computational Physics

October 13, 2024

Declaration of Authorship

I, Tobias Fellner, declare that this thesis titled, "Quantum Machine Learning for Time Series Prediction" and the work presented in it are my own. I confirm that:

- I have written my thesis independently.
- I have not used any sources other than those indicated and have marked all statements taken verbatim or in spirit from other works as such.
- The thesis submitted has not been the subject of another examination procedure either in full or in substantial parts.
- I have not already published the thesis either in full or in part, unless the examiner has previously approved the publication.
- The electronic copy corresponds to the other copies.
- When using IT/AI-supported writing tools, I have listed these tools in full as aids used with their product name, my source of supply and an overview of the range of functions used in this thesis.
- In the preparation of this thesis, I have worked independently throughout and have controlled the use of IT/AI-supported writing tools.

In this work the following AI-tools were used:

- For spellchecking, paraphrasing and ensuring an easy to understand tone GPT-4o mini as well as GPT-4o (openai.com) in the versions released after July 18, 2024 are used. A section, written by myself, is improved by these models by a prompt similar to the following: *"Please correct grammar and spelling errors, paraphrase when necessary to avoid redundant words and ensure readability."*
- Furthermore, DeepL Write (deepl.com) is used for paraphrasing.
- When writing the code to implement the models, the training procedure and creating the plots, GitHub Co-Pilot (github.com) was used for code improvement and filling out redundant lines of code.

Date:

Signed:

UNIVERSITY OF STUTTGART

Abstract

Institute for Computational Physics

Master of Science

Quantum Machine Learning for Time Series Prediction

by Tobias Fellner

Time series prediction is an essential task in various fields, such as meteorology, finance and healthcare. Traditional approaches to time series prediction have primarily relied on regression and moving average methods, but recent advancements have seen a growing interest in applying machine learning techniques. With the rise of quantum computing, it is of interest to explore whether quantum machine learning can offer advantages over classical methods for time series forecasting. This thesis presents the first large-scale systematic benchmark comparing classical and quantum models for time series prediction. A variety of quantum models are evaluated against classical counterparts on different datasets. A novel quantum reservoir computing architecture is proposed, demonstrating promising results in handling nonlinear prediction tasks. The findings suggest that, for simpler time series prediction tasks, quantum models achieve accuracy comparable to classical methods. However, for more complex tasks, such as long-term forecasting, certain quantum models show improved performance. While current quantum machine learning models do not consistently outperform classical approaches, the results point to specific contexts where quantum methods may be beneficial.

Summary in German

Zeitreihenanalysen sind in vielen Bereichen von zentraler Bedeutung, darunter Meteorologie, Finanzen und Medizin, da präzise Vorhersagen auf Basis historischer Daten wertvolle Einblicke und Prognosen ermöglichen. Während herkömmliche Methoden wie Regression und Moving Average lange Zeit für die Zeitreihenvorhersage verwendet wurden, haben maschinelle Lernverfahren in den letzten Jahrzehnten zunehmend an Bedeutung gewonnen. Mit der Entwicklung der Quantencomputer stellt sich die Frage, ob Quantum Machine Learning (QML) einen Vorteil gegenüber klassischen Modellen bieten kann, insbesondere bei der Vorhersage von Zeitreihen.

Diese Arbeit präsentiert die erste umfassende Benchmark-Studie, die verschiedene Quanten- und klassische Modelle für die Vorhersage von Zeitreihen systematisch miteinander vergleicht. Untersucht werden unterschiedliche Quantenmodelle, darunter Variational Quantum Circuits (VQC), Quantum Long Short-Term Memory (QLSTM) und Quantum Reservoir Computing (QRC). Besonders berücksichtigt wird dabei die Rolle der Quantenverschränkung, die als potenzielle Quelle eines Quanten-Vorteils gilt. Im Rahmen der Arbeit wird eine neuartige QRC-Architektur entwickelt, die vielversprechende Ergebnisse insbesondere für nicht-lineare Vorhersageaufgaben liefert.

Die Ergebnisse der Benchmark-Studie zeigen, dass Quantenmodelle bei einfachen Vorhersagen ähnlich gute Ergebnisse wie klassische Modelle erzielen. Für komplexere Aufgaben, wie die Vorhersage von bis zu 100 Zeitschritten im Mackey-Glass-Datensatz, können einige Quantenmodelle jedoch eine signifikant höhere Genauigkeit erreichen, insbesondere eine VQC-Architektur und eine QRC-Architektur. Diese Ergebnisse deuten darauf hin, dass QML in spezifischen Szenarien Vorteile gegenüber klassischen Methoden bieten kann.

Gleichzeitig zeigt die Arbeit auf, dass in bestimmten Fällen klassische Modelle in der Vorhersagegenauigkeit überlegen sind, insbesondere bei Vorhersagen über kürzere Zeiträume. Zukünftige Forschung sollte sich darauf konzentrieren, die spezifischen Klassen von Zeitreihenproblemen zu identifizieren, in denen Quantenmodelle überlegen sind, und dabei tiefergehende Untersuchungen zu Quantenmechanischen Eigenschaften wie der Verschränkungsentropie und der Quantum Fisher Information durchführen.

Darüber hinaus eröffnet die Arbeit neue Forschungsrichtungen, insbesondere im Hinblick auf das Lernen mit quantenmechanischen Daten. Während der Schwerpunkt dieser Arbeit auf klassischen Daten lag, könnten Quantenmodelle durch ihre intrinsische Quantenstruktur Vorteile bei der Verarbeitung von quantenmechanischen Daten bieten, beispielsweise bei der Klassifikation von Quantenphasen oder der Rekonstruktion von Quantenzuständen. Diese Ansätze könnten das Verständnis der praktischen Anwendbarkeit von QML weiter vertiefen und zukünftig zu konkreten Anwendungen von Quantencomputern führen.

Acknowledgements

I would like to begin by expressing my sincerest gratitude to Professor Christian Holm for his guidance and support throughout the process of completing this Master's thesis. His experience and physical intuition ensured that I remained on track. I'm extremely grateful to David Kreplin, who first piqued my interest in quantum machine learning and provided invaluable guidance throughout the past year, drawing on his extensive experience in the field. Our numerous discussions were instrumental in enhancing the quality of my work and will undoubtedly prove invaluable as I embark on my future endeavors. I would like to express my gratitude to Daniel Fink for providing me with the opportunity to work on this thesis and for ensuring a smooth start. Special thanks to Samuel Tovey for sharing his expertise on machine learning and quantum reservoir computing and for guiding me through the process of publishing research. I would also like to express my gratitude to the entire quantum computing group at the Fraunhofer IPA for their invaluable feedback and support. I am particularly indebted to Simone Blümlein and Frank Huber for their administrative and technical assistance, as well as the entire Institute for Computational Physics for welcoming me so warmly. I am excited to continue learning and working with you all throughout my Ph.D.

Contents

List of Abbreviations

MSE	**M**ean **S**quared **E**rror
MAE	**M**ean **A**bsolute **E**rror
MLP	**M**ulti **L**ayer **P**erceptron
QML	**Q**uantum **M**achine **L**earning
VQA	**V**ariational **Q**uantum **A**lgorithm
VQC	**V**ariational **Q**uantum **C**ircuit
PQC	**P**arameterized **Q**uantum **C**ircuit
QNN	**Q**uantum **N**eural **N**etwork
ELM	**E**xtreme **L**earning **M**achine
RC	**R**eservoir **C**omputing
QRC	**Q**uantum **R**eservoir **C**omputing
RNN	**R**ecurrent **N**eural **N**etwork
LSTM	**L**ong **S**hort-**T**erm **M**emory
DVQC	**D**ressed **V**ariational **Q**uantum **C**ircuit
QRNN	**Q**uantum **R**ecurrent **N**eural **N**etwork
QLSTM	**Q**uantum **L**ong **S**hort-**T**erm **M**emory
LE-QLSTM	**L**inear **E**nhanced **Q**uantum **L**ong **S**hort-**T**erm **M**emory
MAD	**M**edian **A**bsolute **D**eviation
PCC	**P**earson **C**orrelation **C**oefficient

Chapter 1

Introduction

In our daily lives, we frequently encounter sequential data across various domains. Examples include meteorological data, such as weekly temperature trends, financial data, such as annual index values, and medical or sports data, such as heart rate measurements over the duration of training sessions. Over the last century, significant efforts have been dedicated to developing methods for analyzing sequential data. A primary objective is to predict future values in a time series based on historical data [1, 2]. Accurate extrapolation from time series can provide substantial benefits across diverse fields, including the natural and social sciences.

Initially, models for time series prediction predominantly utilized regression and moving averages [3]. However, in the past two decades, there has been a notable shift towards the exploration of machine learning techniques [4–6]. These approaches involve optimizing the parameters of artificial neural networks, enabling them to learn from the provided data and subsequently extrapolate to forecast future data points.

Given the importance of time series prediction, investigating novel methodologies is warranted. Research in quantum computing suggests that quantum computers may surpass classical computers in specific tasks, such as integer factorization [7] and unstructured database searches [8]. Leveraging quantum mechanical properties like superposition and entanglement, quantum computing has the potential to offer computational advantages in handling vast datasets [9, 10].

The last decade has seen the emergence of quantum machine learning (QML), which extends the principles of machine learning to the quantum domain [11, 12]. A core concept involves iterative optimization of parameters within a variational quantum circuit (VQC) to learn a given task. Recent years have witnessed the proposal of various QML models for time series analysis [13–17]. However, it remains unclear whether and in which contexts QML methods offer an advantage over classical approaches [18]. Furthermore, a comprehensive comparison between proposed predictive time series quantum models and its classical counterparts is lacking.

In this work, we present the first large-scale systematic benchmark study comparing different quantum and classical models for time series prediction. The benchmark encompasses a diverse range of learning tasks across different datasets for a wide variety of model architectures. We examine the role of entanglement in the learning processes of quantum models, as this property is often considered the source of potential quantum advantage. Additionally, we propose a novel quantum reservoir computing architecture for time series prediction and demonstrate its potential benefits for quantum hardware computations compared to other proposed models.

For time series prediction tasks that involve forecasting a small number time steps ahead, our results indicate that quantum models, at best, match the prediction accuracy of classical methods. In such cases, classical approaches seem to be better

suited for time series forecasting. However, for more complex prediction challenges, certain quantum machine learning models as well as the proposed quantum reservoir computing architecture, demonstrate promising performance and suggesting potential benefits.

This thesis is organized as follows. Chapter 2 establishes the theoretical foundation, introducing fundamental concepts in machine learning and quantum computing. We provide an introduction to quantum machine learning and an overview of quantum reservoir computing. In Chapter 3, we discuss the methodology of the benchmark study, including the mathematical framework for time series prediction employed in this work. We also cover the classical and quantum models utilized, along with details regarding the benchmark setup, hyperparameter choices, and datasets. Chapter 4 presents an in-depth analysis of the results and their implications, along with arguments supporting our quantum reservoir approach. Finally, Chapter 5 concludes the work and offers an outlook for future research directions.

Chapter 2

Theory and Literature Review

2.1 Machine Learning

Over the last few decades, substantial research efforts have been dedicated to the study of machine learning (ML). With the increasing volume of available data, it is becoming ever more critical to develop algorithms capable of generalizing from data patterns and making predictions in situations with limited information. ML has proven to be an effective tool for tasks such as classifying labeled datasets of images [19–21] or learning patterns in multi-dimensional time series [22, 23]. Such a learning strategy is called supervised learning [24].

We also want to mention the unsupervised learning task [25], where the goal is to learn the probability distribution of a dataset $\mathcal{D} = \{x^1, \ldots, x^M\}$ to generate new samples from it. However, since this thesis focuses on supervised learning, we will only cover supervised ML in the following.

In this chapter, we introduce the fundamentals of ML relevant to this thesis, following chapter two of the textbook "Machine Learning with Quantum Computers" by Maria Schuld [26]. We begin by defining the supervised learning problem and explaining the training process before introducing different machine learning methods.

2.1.1 Supervised Machine Learning

In a supervised learning task, the objective is to establish a relationship between an input domain $\mathcal{X}$ and an output domain $\mathcal{Y}$. Let $\mathcal{D} = \{(x^1, y^1), \ldots, (x^M, y^M)\}$ be a dataset of pairs $(x^m, y^m) \in \mathcal{X} \times \mathcal{Y}$ with inputs x^m and target outputs y^m. The goal is to learn the underlying probability distribution between inputs and target outputs to predict the outputs $y \in \mathcal{Y}$ for unseen inputs $x \in \mathcal{X}$.

We can categorize learning tasks into classification and regression problems, distinguished by the nature of their output domain $\mathcal{Y}$. In classification tasks, $\mathcal{Y}$ is a set of D discrete class labels $\{l_1, \ldots, l_D\}$. For instance, in a task involving the classification of images of cats and dogs, each with $n \times m$ pixels, the input domain might be a matrix in $\mathbb{R}^n \times \mathbb{R}^m$. The output domain on the other hand would be $\mathcal{Y} = \{-1, 1\}$, where -1 represents a cat and 1 represents a dog, allowing for exchanged labeling configurations. Conversely, in regression tasks, the output domain $\mathcal{Y}$ is continuous. For example, in predicting a stock index based on n economic measures, the input domain could be a vector in $\mathbb{R}^n$, and the output domain would be a continuous value in $\mathbb{R}$ representing the stock index's value on the following day. Since time series prediction is a regression problem, this thesis will primarily focus on regression tasks.

To map from the input domain $\mathcal{X}$ to the output domain $\mathcal{Y}$, we define a model as a function

$$f\colon \mathcal{X} \to \mathcal{Y}, \quad f(x^m) = y^m, \quad x^m \in \mathcal{X},\, y^m \in \mathcal{Y}\,, \tag{2.1}$$

that replicates the characteristics of data samples (x^m, y^m) of a dataset

$$\mathcal{D} = \{(x^1, y^1), \ldots, (x^M, y^M)\}\,. \tag{2.2}$$

The model can depend on parameters θ, thereby defining a model family $\{f_\theta\}$. To select the model out of the model family that reproduces the probability distribution of the dataset $\mathcal{D}$ best, we require a measure for the accuracy of the model. This measure can be formalized by defining a loss. For classification tasks this can be based on the accuracy of classifying elements correctly,

$$\text{accuracy} = \frac{\text{number of correctly classified elements}}{\text{total number of elements}}\,. \tag{2.3}$$

The corresponding loss, representing the error rate, is

$$\text{error} = 1 - \text{accuracy}\,. \tag{2.4}$$

In scenarios with a continuously-valued output domain, a continuous-valued measure is necessary. A widely adopted loss function for regression tasks is the Mean-Squared-Error (MSE)

$$L(f_\theta(x), y) = (f_\theta(x) - y)^2\,. \tag{2.5}$$

Alternatively, the Mean-Absolute-Error (MAE) provides another approach,

$$L(f_\theta(x), y) = |f_\theta(x) - y|\,. \tag{2.6}$$

Various other loss functions like the Root-Mean-Square-Error, the Mean-Bias-Error or the Relative-Absolute-Error have been introduced, each suited to different objectives. The choice of the most suitable loss depends on the specific goals of the task at hand. For example, mean-squared-error is more sensitive to large deviations between predicted and actual values compared to mean-absolute-error.

2.1.2 Training a Machine Learning Model

Training a supervised machine learning model entails identifying the optimal model f_θ^* within the model family $\{f_\theta\}$ that minimizes the cost function $\mathcal{L}(f_{\theta^*}(x), y)$

$$\theta^* = \underset{\theta}{\operatorname{argmin}}\, \mathcal{L}(f_{\theta^*}(x), y) = \underset{\theta}{\operatorname{argmin}}\, \frac{1}{M} \sum_{m=1}^{M} L(f_\theta(x^m), y^m) \tag{2.7}$$

for a chosen loss L over M data samples (x^m, y^m) in a dataset $\mathcal{D}$. This process is equivalent to optimizing the parameter set θ^* for the parameterized model family. To determine the optimal solution, the parameters of the cost function $\mathcal{L}(\theta) \equiv \mathcal{L}(f_\theta(x), y)$ can be iteratively updated by

$$\theta^{(t+1)} = \theta^{(t)} - \eta \nabla \mathcal{L}(\theta^{(t)})\,. \tag{2.8}$$

In this method known as gradient descent, the parameters are updated in each iteration t by a term that includes the learning rate η and the gradient of the cost function $\nabla \mathcal{L}(\theta^{(t)})$. This approach drives the cost function towards a minimum $\mathcal{L}(\theta^*)$ within

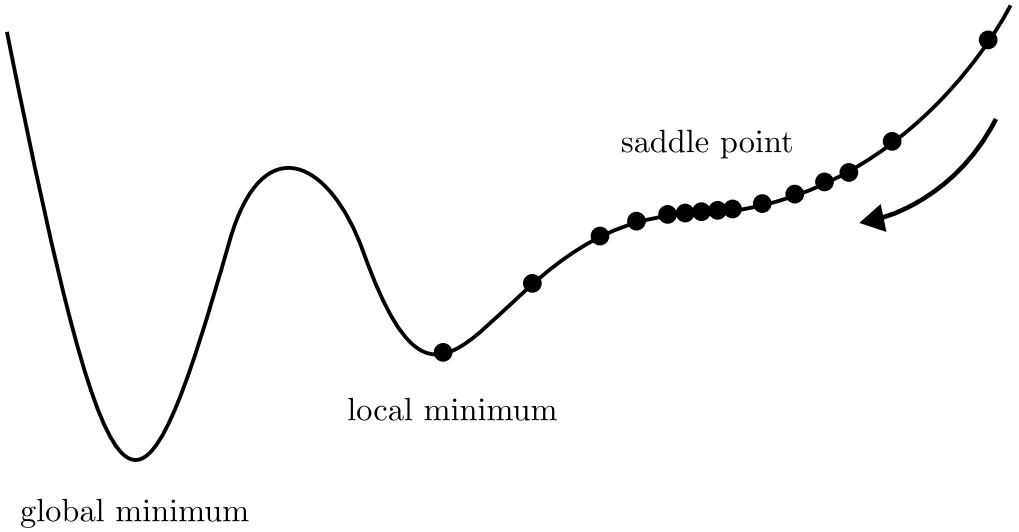

FIGURE 2.1: Training updates within the cost landscape of a one-dimensional parameter space. Gradient descent algorithms may become trapped in local minima, and convergence at saddle points can be slow. This figure is adapted from Figure 2.14 in [26].

the landscape defined by $\mathcal{L}(\theta)$. In order to efficiently compute the gradient of the cost function with respect to each parameter, a method known as backpropagation is typically employed. However, gradient descent can result in the cost function becoming stuck in a local minimum or experiencing slow convergence at saddle points, as shown in Figure 2.1.

To mitigate these issues, one can extend the concept of gradient descent to stochastic gradient descent. This method introduces stochastic, or dynamic, elements into the parameter update process. For example, by selecting a subset of randomly sampled data to perform a single parameter update, the algorithm can potentially escape local minima due to the inherent stochasticity of the data sampling process [27]. These subsets of data are referred to as batches, and a full cycle of parameter updates for all data in training is called an epoch. A related and widely used optimization algorithm is the Adam optimizer [28], which dynamically adjusts the learning rate and bases the update of individual parameters on changes made in previous steps. The Adam optimizer will be predominantly utilized throughout this thesis.

One fundamental problem that can occur during training is overfitting, where a model captures the data used in training but fails to generalize to new data from the underlying probability distribution. This phenomenon is demonstrated in Figure 2.2 (a) for a regression task. To address overfitting, the dataset is typically split into three distinct sets: training data used for model parameter updates, validation data used to monitor performance during training, and test data used to evaluate the final model's generalizability. A common strategy is to halt training when the validation error begins to rise, as illustrated in Figure 2.2 (b), indicating that the model is overfitting to the training data and losing its ability to generalize effectively.

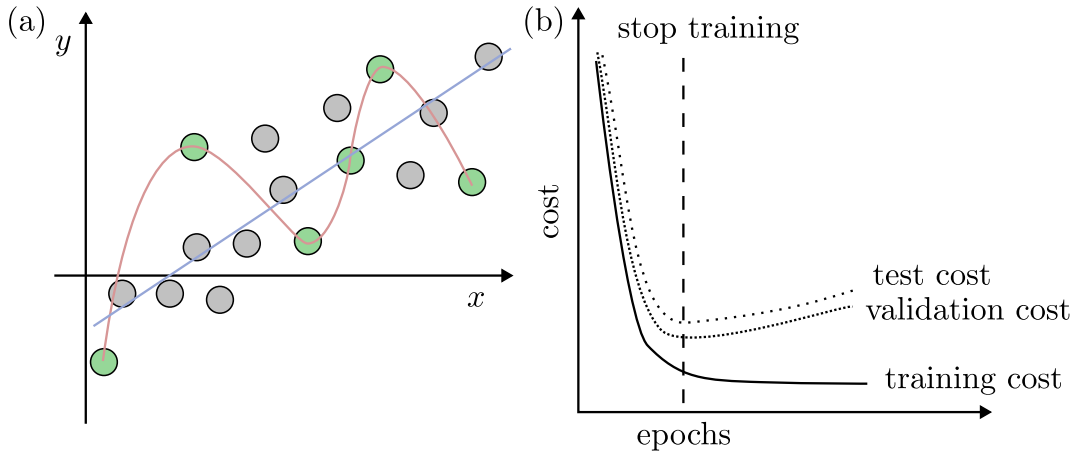

FIGURE 2.2: (a): Overfitting in a regression task. The data points highlighted in green are used for training. The trained model (red line) fits the training data perfectly but fails to generalize to unseen data. The entire dataset is better represented by the linear trend (blue line) than by the trained model. (b): Training, validation, and test costs over the epochs of training. In the event of overfitting, the training loss decreases further while the validation loss begins to increase. When this occurs, training should be stopped to prevent further overfitting. These figures are adapted from Figures 2.11 and 2.15 in [26].

2.1.3 Methods in Machine Learning

Having established how supervised machine learning models are trained, we now shift our attention to understanding the underlying structures of these models.

A linear model $f_{\mathbf{w}}$ is a simple parameterized function mapping an input domain $\mathcal{X}$ to an output domain $\mathcal{Y}$. In the context of real-valued vectors, this model takes an input $\mathbf{x} \in \mathbb{R}^N$ and transforms it using a weight vector $\mathbf{w} \in \mathbb{R}^N$ by

$$f_{\mathbf{w}}(\mathbf{x}) = \mathbf{w}^T\mathbf{x}\,. \tag{2.9}$$

We note that the linear model can also incorporate a scalar bias $b \in \mathbb{R}$. However, this bias can be absorbed into the notation by extending $\mathbf{w}$ with an additional dimension and adding an additional value $x_0 = 1$ to $\mathbf{x}$. The weight vector $\mathbf{w}$ is optimized during training to minimize the associated cost. However, linear models can only capture linear dependence in the data. For many problems, however, a nonlinear description is relevant. A commonly used nonlinear model is a neural network. They offer greater flexibility and complexity by combining multiple perceptrons, denoted as $\phi(a)$. Each perceptron receives a weighted input $a = \mathbf{w}^T\mathbf{x}$ and then applies an activation function ϕ. The concept of a perceptron is illustrated in Figure 2.3 (a). Different activation functions can be used, the most common ones include the sigmoid function

$$\phi(a) = \frac{1}{1+e^{-a}}\,, \tag{2.10}$$

the hyperbolic tangent

$$\phi(a) = \tanh a \tag{2.11}$$

or the rectified linear units

$$\phi(a) = \begin{cases} a & \text{if } a > 0 \\ 0 & \text{else.} \end{cases} \tag{2.12}$$

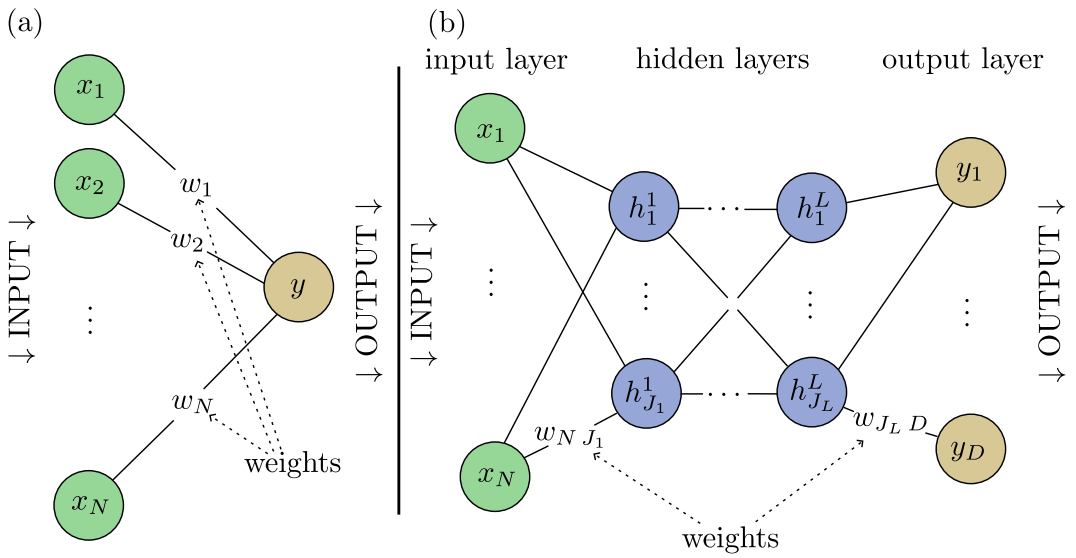

FIGURE 2.3: (a): Graphical representation of a perceptron. The perceptron receives a weighted input, denoted as $\mathbf{w}^T\mathbf{x}$, and outputs a value y. (b): By stacking multiple perceptrons, a neural network with layered perceptrons can be constructed. In this configuration, the output of perceptrons in one layer serves as the input for the perceptrons in the subsequent layer. The figures are adapted from Figures 2.17 and 2.18 in [26].

These activation functions add nonlinearity to the model.

By stacking multiple perceptrons in a layered structure, we construct a neural network, with the output of multiple perceptrons serving as the weighted input for the next. The set of perceptrons within layer i is denoted as $\boldsymbol{\phi}_i$ in the following. A prime example is the feed-forward neural network, represented as

$$f_{\mathbf{W}_1,\mathbf{W}_2,\ldots,\mathbf{W}_L}(\mathbf{x}) = \boldsymbol{\phi}_L(\mathbf{W}_L \cdots \boldsymbol{\phi}_2(\mathbf{W}_2\boldsymbol{\phi}_1(\mathbf{W}_1\mathbf{x}))\cdots)\,. \tag{2.13}$$

Here, the input vector $x \in \mathbb{R}^N$ is processed through L layers $l = 1,\ldots,L$. Each layer contains J_l perceptrons with weights $\mathbf{W_l}$ summarized in $\mathbb{R}^{J_l \times J_l}$ matrices. The layered structure has led to the alternative name "Multi-Layer Perceptron" (MLP). As shown in Figure 2.3 (b), data from the input domain is passed through hidden layers of perceptrons $h_1^l, \cdots, h_{J_l}^l$ before reaching the output domain $\mathcal{Y}$. The MLP, being a fundamental architecture, has paved the way for a wide range of neural network architectures tailored to specific tasks. For sequential data processing, prominent examples include Recurrent Neural Networks (RNNs) [29, 30] and Long Short-Term Memory (LSTM) networks [22], which are discussed in Chapter 3.

2.2 Quantum Computing

In the following, we introduce the fundamental elements of quantum computing. We begin with the computational unit of quantum computing, the qubit, and move on to basic quantum operations known as quantum gates. Next, we discuss quantum measurements as well as the quantum circuit, which is the computational unit for describing quantum algorithms. The following sections present a general introduction to quantum computing and its commonly used notations. A more detailed

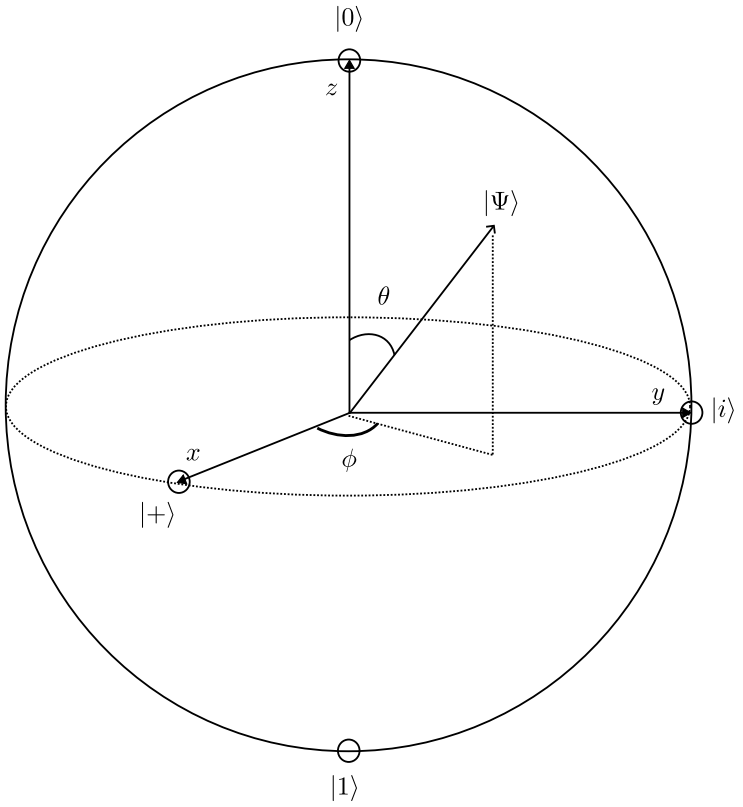

FIGURE 2.4: The Bloch sphere representation of the qubit state $|\Psi\rangle$ illustrates the state as a unit vector on the sphere, defined by two angles, θ and ϕ. The computational states $|0\rangle$ and $|1\rangle$ are located at the north and south pole of the sphere, respectively. This figure is inspired by Figure 1.3 in [31].

introduction can be found in the textbook "Quantum Computation and Quantum Information" by Nielsen and Chuang [31] from which we adapted our notation.

2.2.1 Qubits

The fundamental computational unit in classical computation is a bit, which can either be in state 0 or 1. In the context of quantum computing, this fundamental unit is extended to the quantum bit, or qubit for short. A qubit can be understood as a physical quantum two-level system, such as a spin, which can be in a spin-up or spin-down state. Utilizing bra-ket notation, we can, without loss of generality, designate the state with lower energy as $|0\rangle$ and the state with higher energy as $|1\rangle$. In the quantum-classical analogy, these states correspond to the classical computational states 0 and 1. In contrast to classical states, quantum states allow for superposition, whereby a qubit can exist simultaneously in states $|0\rangle$ and $|1\rangle$:

$$|\Psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle\,, \tag{2.14}$$

where $\alpha, \beta \in \mathbb{C}$ and $|\alpha|^2 + |\beta|^2 = 1$. This property is a fundamental feature for a potential advantage of quantum computing over classical computing. The normalization condition must be considered because α and β can be interpreted as probability amplitudes. Measuring the quantum state $|\Psi\rangle$ results in $|0\rangle$ with probability

$|\alpha|^2$ and $|1\rangle$ with probability $|\beta|^2$. Consequently, the qubit state is mathematically represented as a unit vector in the two-dimensional complex Hilbert space H_2. An illustrative representation of a qubit is achieved by interpreting its quantum state as a unit vector on the Bloch sphere, as depicted in Figure 2.4. The state can be expressed in terms of spherical coordinates:

$$|\Psi\rangle = \cos\frac{\theta}{2}|0\rangle + e^{i\phi}\sin\frac{\theta}{2}|1\rangle . \tag{2.15}$$

Here, the overall phase of the state is chosen in a way that the amplitude of $|0\rangle$ is real. A unit vector aligned along the z-axis corresponds to the $|0\rangle$ state, while a unit vector pointing in the opposite z-direction corresponds to the $|1\rangle$ state. States where the unit vector aligns with other axes are commonly referred to in the literature as:

$$\begin{aligned} |+\rangle &= \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle), \\ |-\rangle &= \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle), \\ |i\rangle &= \frac{1}{\sqrt{2}}(|0\rangle + i|1\rangle), \\ |-i\rangle &= \frac{1}{\sqrt{2}}(|0\rangle - i|1\rangle). \end{aligned} \tag{2.16}$$

2.2.2 Quantum Gates

In order to perform computations on the qubits, we have to define operations on the quantum state. Such operations are called quantum gates. We first discuss the mathematical formalism for gates on only one qubit before moving on the multi-qubit gates.

As the quantum state of a single qubit can mathematically be described as a unit-vector in the complex two dimensional Hilbert space H_2, operations on the state have to be unitary in order to preserve the norm. The single-qubit Pauli gates

$$\begin{aligned} X &= |0\rangle\langle 1| + |1\rangle\langle 0| \\ Y &= -i|0\rangle\langle 1| + i|1\rangle\langle 0| \\ Z &= |0\rangle\langle 0| - |1\rangle\langle 1| \end{aligned} \tag{2.17}$$

can be understood as π rotations of the quantum state around the corresponding axis in the Bloch sphere picture up to a phase. Following Dirac-Notation, $\langle\cdot|$ is the conjugate transpose of $|\cdot\rangle$. The X gate is the analog to the classical NOT operation to the states $|0\rangle$ and $|1\rangle$:

$$X(\alpha|0\rangle + \beta|1\rangle) = \beta|0\rangle + \alpha|1\rangle . \tag{2.18}$$

The Z gate adds a phase to the state $|1\rangle$:

$$Z(\alpha|0\rangle + \beta|1\rangle) = \alpha|0\rangle - \beta|1\rangle . \tag{2.19}$$

Finally, the Y gate flips the amplitudes of $|0\rangle$ and $|1\rangle$ and adds a phase:

$$Y(\alpha|0\rangle + \beta|1\rangle) = -i\beta|0\rangle + i\alpha|1\rangle . \tag{2.20}$$

Another important single-qubit gate is the Hadamard gate H, which creates equal superpositions when applied to the states $|0\rangle$ and $|1\rangle$:

$$\begin{aligned} H|0\rangle &= \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) = |+\rangle \ , \\ H|1\rangle &= \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) = |-\rangle \ . \end{aligned} \tag{2.21}$$

Furthermore, rotations with an arbitrary angle θ on the Bloch sphere around a Cartesian axis can be archived by applying the rotation gates

$$\begin{aligned} R_x(\theta) = \exp\left(-i\frac{\theta}{2}X\right) &= \cos\frac{\theta}{2}|0\rangle\langle 0| - i\sin\frac{\theta}{2}|0\rangle\langle 1| \\ &\quad - i\sin\frac{\theta}{2}|1\rangle\langle 0| + \cos\frac{\theta}{2}|1\rangle\langle 1| \ , \\ R_y(\theta) = \exp\left(-i\frac{\theta}{2}Y\right) &= \cos\frac{\theta}{2}|0\rangle\langle 0| - \sin\frac{\theta}{2}|0\rangle\langle 1| \\ &\quad + \sin\frac{\theta}{2}|1\rangle\langle 0| + \cos\frac{\theta}{2}|1\rangle\langle 1| \ , \\ R_z(\theta) = \exp\left(-i\frac{\theta}{2}Z\right) &= e^{i\frac{\theta}{2}}|0\rangle\langle 0| + e^{i\frac{\theta}{2}}|1\rangle\langle 1| \ . \end{aligned} \tag{2.22}$$

To let different qubits in the same computational process interact with each other, one needs to introduce multi-qubit gates. The quantum state of a n-qubit system is described by a unit-vector in the 2^n dimensional Hilbert space H_{2^n}. For example for $n = 2$, let $|\Psi_1\rangle$ and $|\Psi_2\rangle$ be the quantum state of two single qubits:

$$\begin{aligned} |\Psi_1\rangle &= \alpha|0\rangle + \beta|1\rangle \ , \\ |\Psi_2\rangle &= \gamma|0\rangle + \delta|1\rangle \ . \end{aligned} \tag{2.23}$$

In this case, the quantum state of the two qubit system $|\Psi\rangle$ consisting of the two qubits in states $|\Psi_1\rangle$ and $|\Psi_2\rangle$ can be constructed by the tensor product of the single-qubits states

$$|\Psi\rangle = |\Psi_1\rangle \otimes |\Psi_2\rangle = \alpha\gamma|0\rangle \otimes |0\rangle + \alpha\delta|0\rangle \otimes |1\rangle + \beta\gamma|1\rangle \otimes |0\rangle + \beta\delta|1\rangle|1\rangle \ . \tag{2.24}$$

To simplify the notation, the tensor product symbol $\otimes$ is omitted in the following: $|i\rangle \otimes |j\rangle = |ij\rangle$. Multi-qubit gates are unitary operations on H_{2^n}. They play a crucial role in quantum computing as they can lead to entangled quantum states. We call a n-qubit quantum system in state $|\Psi\rangle$ entangled if the quantum state can not be expressed as a product of single-qubit states $|\Psi_i\rangle$

$$|\Psi\rangle = |\Psi_i\rangle \otimes |\Psi_2\rangle \otimes \cdots \otimes |\Psi_n\rangle \ . \tag{2.25}$$

Entanglement is a crucial element in the field of quantum computing, as it gives access to all possible normalized state configurations within the Hilbert space. Alongside superposition, it is a fundamental property for a potential computational advantage of quantum algorithms over their classical counterparts. An important two-qubit gate is the quantum controlled NOT ($CNOT$) gate:

$$CNOT = |00\rangle\langle 00| + |01\rangle\langle 01| + |10\rangle\langle 11| + |11\rangle\langle 10| \ . \tag{2.26}$$

It is the quantum counterpart to the classical $CNOT$ gate. The first qubit is called the control qubit, the second qubit is called the target qubit. Only if the control qubit is in state $|1\rangle$, the target qubit is flipped:

$$\begin{aligned} CNOT\,|00\rangle &= |00\rangle\ , \\ CNOT\,|01\rangle &= |01\rangle\ , \\ CNOT\,|10\rangle &= |11\rangle\ , \\ CNOT\,|11\rangle &= |10\rangle\ . \end{aligned} \tag{2.27}$$

Based on the state of the control qubit, a Pauli-X rotation on the target qubit is performed. Thus, the $CNOT$ gate is also called the controlled X-gate (CX). In general, we can define a controlled two qubit gates CU for an arbitrary unitary operation U on the target qubit.

We want to finalize our discussion of quantum gates by demonstrating that controlled multi-qubit gates can lead to entangled states. We consider a two qubit system and a $CNOT$ gate. Let the first qubit be initially in state $|+\rangle$ and the second qubit is in state $|0\rangle$. The quantum state of the two qubit system is

$$|\Psi\rangle = |+\rangle \otimes |0\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle + |1\rangle\right) \otimes |0\rangle = \frac{1}{\sqrt{2}}\left(|00\rangle + |10\rangle\right)\ . \tag{2.28}$$

Applying the $CNOT$ gate to the state gives

$$|\Psi'\rangle = CNOT\,|\Psi\rangle = \frac{1}{\sqrt{2}}\left(|00\rangle + |11\rangle\right)\ . \tag{2.29}$$

It is impossible to write the state $|\Psi'\rangle$ as a product of single-qubit states $|\Psi_1\rangle \otimes |\Psi_2\rangle$. Therefore the state $|\Psi'\rangle$ is entangled.

2.2.3 Quantum Measurements

To obtain information about an n-qubit quantum state, one must perform quantum measurements, which are defined as a set $\{M_m\}$ of measurement operators acting on the complex Hilbert space H_{2^n}. The index m denotes the different possible outcomes of the quantum measurement. Given a quantum state $|\Psi\rangle$ prior to measurement, the probability of observing outcome m is expressed as

$$p(m) = \langle\Psi|\, M_m^\dagger M_m\, |\Psi\rangle\ . \tag{2.30}$$

The measurement operators M_m must satisfy the completeness relation

$$\sum_m M_m^\dagger M_m = I\,, \tag{2.31}$$

ensuring that the probabilities of all possible outcomes sum to one:

$$1 = \sum_m p(m) = \sum_m \langle\Psi|\, M_m^\dagger M_m\, |\Psi\rangle\ . \tag{2.32}$$

A special case of quantum measurements are projective measurements. They are defined by a hermitian operator M acting on H_{2^n}. This operator can be decomposed as

$$M = \sum_\lambda P_\lambda\,, \tag{2.33}$$

where P_λ is the projector onto the eigenspace of M with eigenvalue λ. The probability of obtaining the result λ is given by

$$p(\lambda) = \langle \Psi | \, P_\lambda \, | \Psi \rangle \ . \tag{2.34}$$

Let λ be the outcome of a measurement, the state of the quantum system after measuring M is given by

$$\frac{P_\lambda \, |\Psi\rangle}{\sqrt{p(\lambda)}} \tag{2.35}$$

In general, measuring causes the wavefunction corresponding to the quantum state before measurement to collapse, reducing the quantum state $|\Psi\rangle$ to the eigenstate associated with the measured eigenvalue λ.

The advantageous property of projective measurements is that it is easy to calculate the average value of a measurement, called the expectation value

$$\begin{aligned} \mathbb{E}_{|\Psi\rangle}(M) &= \sum_\lambda \lambda p(\lambda) \\ &= \sum_\lambda \lambda \, \langle \Psi | \, P_\lambda \, | \Psi \rangle \\ &= \langle \Psi | \left(\sum_\lambda \lambda P_\lambda \right) | \Psi \rangle \\ &= \langle \Psi | \, M \, | \Psi \rangle \ . \end{aligned} \tag{2.36}$$

In quantum computing, a crucial set of measurements involves the Pauli operators X, Y, and Z, as defined in Equation (2.17). The eigenvalues of these operators are $\lambda \in \{-1, 1\}$, thus the expectation value is constrained within the interval $[-1, 1]$. For multi-qubit systems, it is feasible to measure an operator that is the tensor product of multiple Pauli operators. For instance, in a three-qubit system, one might measure

$$Z \otimes X \otimes Z \equiv ZXZ \, , \tag{2.37}$$

where ZXZ is referred to as the Pauli string representation of the measurement.

2.2.4 Quantum Circuit

Combining the elements explored in the previous sections, we arrive at quantum circuits, the fundamental building blocks of quantum computation. A quantum circuit consists of multiple qubits, typically initialized in their ground state $|0\rangle$. Various quantum gates can be applied to manipulate this multi-qubit state. Finally one can perform measurements on the resulting quantum state. This process can be illustrated by a circuit diagram as shown in Figure 2.5 (a). Each qubit is represented by a horizontal line. The corresponding circuit diagram symbols for the quantum gates introduced above are shown in Figure 2.5 (b). For single-qubit gates, the operation is indicated within a box. For two-qubit gates, a dot marks the control qubit, while the operation applied to the target qubit is shown in the connected box. This visual representation provides a clear overview of the operations within the circuit.

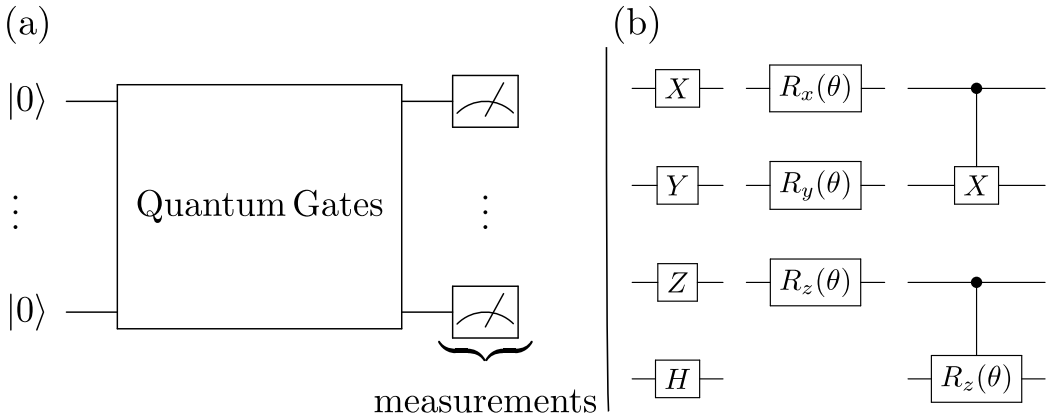

FIGURE 2.5: The circuit diagram of a quantum circuit is shown in (a). Single qubits are represented by horizontal lines. Quantum gates correspond to boxes on and between different lines. Different quantum single and two qubit gates are drawn in (b). Measurements are indicated by a measurement symbol.

2.3 Quantum Machine Learning

In the previous two chapters, we introduced the research areas of machine learning and quantum computing. Each of these fields has the potential to solve specific problems efficiently. Combining them, therefore, presents an opportunity to leverage the strengths of both. This emerging interdisciplinary field is known as quantum machine learning (QML). The fundamental concept in this area of research is to utilize quantum circuits for specific learning tasks. Over the past few years, various QML algorithms have been proposed, including variational quantum algorithms (VQAs) [32] and quantum kernel methods [33]. In this thesis, we focus on the former, as they provide a general framework for solving various tasks including time series analysis. In this chapter, we present the foundational concepts of VQAs, discuss methods for encoding data into quantum states, and explain how to train VQAs. Unless otherwise specified, we follow chapters three and five of the textbook "Machine Learning with Quantum Computers" by Maria Schuld [26].

2.3.1 Variational Quantum Algorithms

The concept of a VQA involves using a parameterized quantum system to execute a learning task. When the quantum system is a quantum circuit, these algorithms are referred to as variational quantum circuits (VQCs) or parameterized quantum circuits (PQCs). These can be viewed as quantum analogs to classical neural networks, as discussed previously in 2.1, and are thus also often called quantum neural networks (QNNs). The idea of using a VQC for a learning task is illustrated in Figure 2.6. The central component is a quantum circuit $U(x, \theta)$, where data x can be encoded, and it contains parameters θ. In the analogy between neural networks and VQCs, these parameters correspond to weights in a classical neural network. The parameters can be realized on a quantum circuit by parameterized rotation gates as introduced in Equation (2.22). Let $|\Psi(x, \theta)\rangle$ represent the quantum state prepared by applying $U(x, \theta)$ to $|0\rangle$. We can measure the expectation value of some chosen observable M to gain insights into the prepared quantum state. This process defines a function

$$f_\theta(x) = \langle \Psi(x, \theta)| M |\Psi(x, \theta)\rangle \tag{2.38}$$

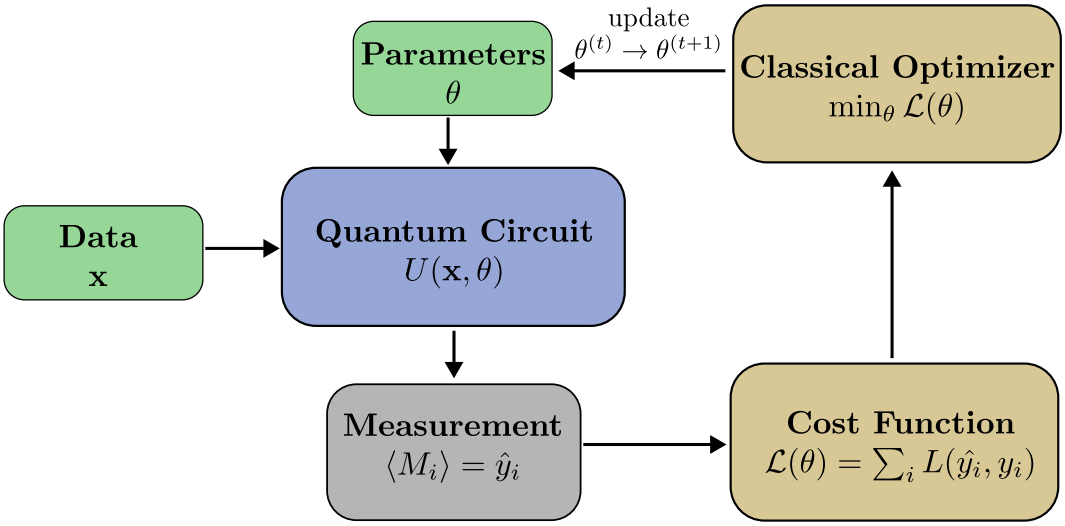

FIGURE 2.6: Variational quantum circuits involve encoding data into a parameterized quantum circuit. By measuring the expectation values of observables on the quantum state, one can calculate a cost function. The objective of the learning task is to minimize this cost by adjusting the parameters of the quantum circuit accordingly. This figure is inspired by Figure 1 in [32].

which characterizes a variational quantum model. Similar to classical supervised machine learning, the outputs for different input data are used to compute a cost function. This cost function can be minimized by updating the parameters θ using optimization techniques, as in classical ML described in Section 2.1.

For the quantum circuit $U(x, \theta)$ itself, various architectures have been proposed. A specific arrangement of quantum gates on the quantum circuit is called an ansatz. Choosing an appropriate ansatz is a complex task and depends heavily on the learning task at hand. Nevertheless, we aim to provide an overview of the fundamental building blocks of quantum circuits used in VQAs. One can distinguish between data encoding blocks $E(x)$ and variational blocks $V(\theta)$, as illustrated in Figure 2.7 (a). The former is responsible for encoding data into a quantum state. We will discuss different encoding strategies in the subsequent section. The variational block, on the other hand, contains the parameters that can be optimized during the training process. It has been demonstrated that a layered structure, repeating these two blocks, enhances the model's expressivity [34]. This approach is known as data re-uploading and is depicted in Figure 2.7 (b). For the structure of the variational blocks $V(\theta)$, one can distinguish between single-qubit gate blocks and entangling blocks. Single-qubit gate blocks introduce trainable parameters through parameterized single-qubit rotations. For instance, one could employ a block of Pauli rotation gates as defined in Equation (2.22). An example of such a structure is shown in Figure 2.7 (c). The purpose of an entangling block is to connect individual qubits using multi-qubit gates. This approach allows the quantum circuit's quantum state to explore the full n-qubit Hilbert space of dimension $\dim(H_{2^n}) = 2^n$, rather than being confined to the product space of n single-qubit Hilbert spaces of dimension $\dim(nH_2) = 2n$. An example of an entangling block is shown in Figure 2.7 (d), where $CNOT$ gates, as defined in Equation (2.26), are applied to all qubits in a nearest-neighbor fashion. However, various entangling architectures and gates can

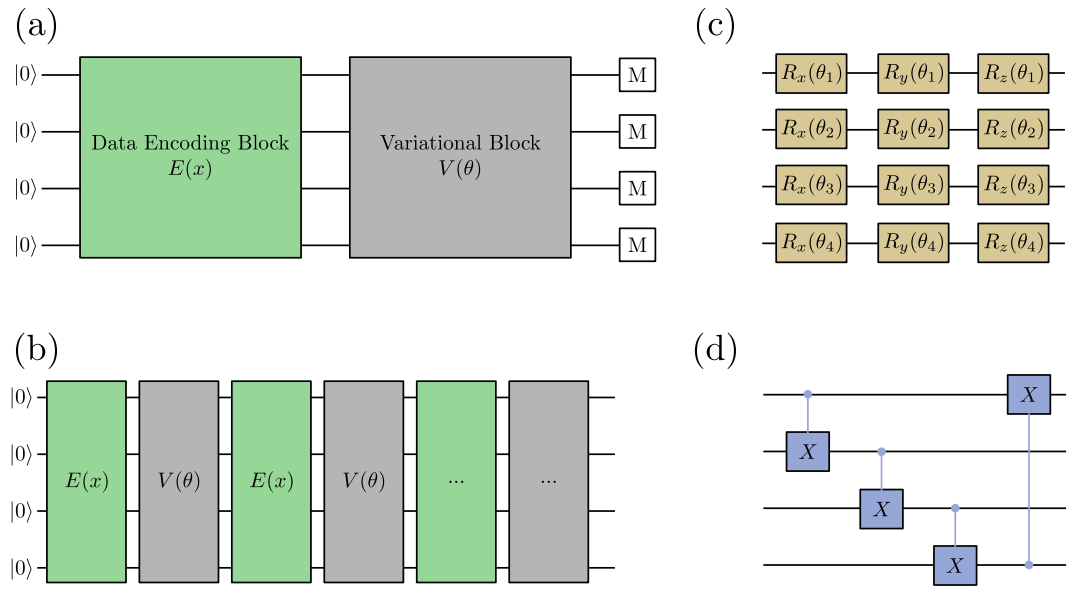

FIGURE 2.7: (a): Basic structure of a variational quantum circuit. In a data encoding block, $E(x)$, data is encoded into a quantum state, whereas a variational block, $V(\theta)$, includes trainable parameters. (b): The data re-uploading scheme. It has been shown that quantum models with this ansatz exhibit higher expressivity [34]. (c): Example of a single-qubit gate block featuring Pauli rotation gates. The optimization of the rotation gate angles occurs during the training process. (d): Entangling Block, where individual qubits are coupled via $CNOT$ gates between nearest neighbors.

be employed. For example, controlled rotation gates can be used, making the entangling block explicitly parameter-dependent. While these architectures are widely used in the literature, it is important to emphasize that selecting a specific ansatz is a nontrivial task, with many variations available [35, 36]. Nevertheless, throughout this thesis, we will primarily adhere to the structures presented in Figure 2.7.

2.3.2 Data Encoding Strategies

Efficiently encoding data into a quantum state is of crucial importance for successfully employing quantum models for learning tasks and are seen as a key bottleneck for QML [34, 37]. In the following we introduce the most important encoding strategies and discuss their individual advantages and disadvantages.

Basis Encoding

One straightforward way to encode data into a quantum state is by assigning an n-bit representation of a number to a computational basis state of an n-qubit quantum system; for example, $|5\rangle = |0101\rangle$. Similar to classical computers, binary representations can be defined for all numbers $x \in \mathbb{C}$. However, the bit length scales with the precision required to represent the number. Therefore, encoding can require a large number of qubits, which is infeasible for most quantum machine learning tasks.

Amplitude Encoding

A different encoding strategy assigns classical data to quantum amplitudes in some basis. For example, a normalized vector $\mathbf{x} \in \mathbb{C}^{2^n}$, where $\sum_k |x_k|^2 = 1$, can be expressed by the amplitudes of a quantum state

$$\mathbf{x} = \begin{pmatrix} x_1 \\ \vdots \\ x_{2^n} \end{pmatrix} \leftrightarrow |\psi_x\rangle = \sum_{j=0}^{2^n-1} x_j |j\rangle \ , \tag{2.39}$$

where the states $|j\rangle$ form a basis. For instance, a normalized vector $\mathbf{x} = (x_1, x_2, x_3, x_4)$ can be represented by a two-qubit system as

$$|\Psi_x\rangle = x_1 |00\rangle + x_2 |01\rangle + x_3 |10\rangle + x_4 |11\rangle \ . \tag{2.40}$$

Note that the number of qubits needed to encode n complex values scales with $\log_2(n)$, making this method more efficient in terms of qubit usage compared to basis encoding. However, there are limitations to this encoding strategy. First, due to the required normalization condition, vectors $\mathbf{u}$ and $\mathbf{v}$ that are scalar multiples of each other, $\mathbf{u} = \alpha\mathbf{v}$ with $\alpha \in \mathbb{C}$, are effectively mapped to the same normalized vector. Thus, an additional dimension is needed to capture the normalization factor to retain information about the actual length of the vector. Second, and more severe, preparing a quantum state according to the amplitude vectors can be very complex and require many quantum gates on actual quantum hardware. While this encoding strategy has been explored in the context of quantum machine learning tasks [38, 39], a more hardware-efficient strategy is widely studied and applied in quantum machine learning tasks. We will introduce this concept in the following section.

Angle Encoding

In angle encoding, a scalar value $x \in \mathbb{R}$ corresponds to the time t of a unitary evolution governed by a Hamiltonian H

$$U(x) = e^{-ixH} \, . \tag{2.41}$$

The resulting quantum state after applying the unitary transformation is $|\Psi(x)\rangle = U(x)|\Psi_0\rangle$. This state depends on the value of x, with the dependence determined by the Hamiltonian H. To encode a single value onto one qubit, a common choice for the Hamiltonian is $H = \frac{1}{2}\mathcal{O}$, where $\mathcal{O} \in \{X, Y, Z\}$ represents one of the Pauli gates, as described in Equation (2.17). With Equation (2.21), the unitary encoding becomes identical to the single-qubit rotation gates $R_x(x)$, $R_y(x)$, and $R_z(x)$. These gates rotate the single-qubit quantum state by an angle x on the Bloch sphere around the respective axis. In this scheme, all values must be scaled into the interval $x \in [0, 2\pi[$ due to the 2π periodicity of the rotation. For example, using R_x rotation gates to encode a vector $\mathbf{x} = (x_1, x_2, x_3, x_4)$ into a quantum state leads to

$$\begin{aligned} |\Psi_\mathbf{x}\rangle = &\left(\cos\frac{x_1}{2}|0\rangle - i\sin\frac{x_1}{2}|1\rangle\right) \otimes \left(\cos\frac{x_2}{2}|0\rangle - i\sin\frac{x_2}{2}|1\rangle\right) \\ &\otimes \left(\cos\frac{x_3}{2}|0\rangle - i\sin\frac{x_3}{2}|1\rangle\right) \otimes \left(\cos\frac{x_4}{2}|0\rangle - i\sin\frac{x_4}{2}|1\rangle\right) \, . \end{aligned} \tag{2.42}$$

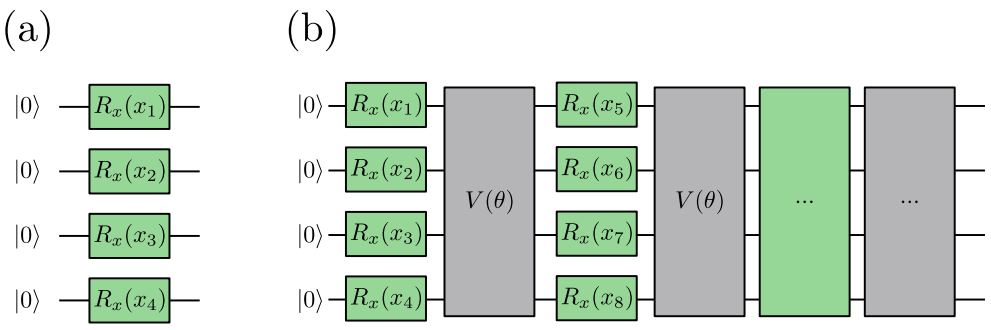

FIGURE 2.8: (a): Pauli rotations gates R_x are used to encode a vector $\mathbf{x} \in \mathbb{R}^4$. (b): By employing a layered structure as in the data re-uploading scheme, different values can be encoded onto the same qubit.

The circuit diagram for this encoding is shown in Figure 2.8 (a). The number of qubits involved equals the number of values to be encoded. However, by using a data re-uploading structure, as introduced in the previous section, multiple values can be encoded onto the same qubit in a layered structure, as shown in Figure 2.8 (b). For a fixed layered structure, the number of qubits required scales linearly with the number of values to be encoded. Due to the efficient scaling of qubits and the ease of implementing this encoding strategy on real quantum hardware, this design is favored by many quantum machine learning researchers [34, 40]. In this thesis, we will predominantly employ this encoding strategy.

2.3.3 Current Challenges in Quantum Machine Learning

After introducing the core concepts of QML, we conclude this section with a summary of the current challenges in the field [40]. First, we address the challenge of selecting appropriate architectures for VQAs. We then examine the limitations of implementing QML methods on current quantum hardware. Finally, we provide an overview of the phenomenon of barren plateaus.

Architecture design of Variational Quantum Algorithms

As previously indicated in Figure 2.7, there is a massive number of potential configurations for the ansatz of VQAs. The arrangement of single and multi-qubit gates is arbitrary. Identifying the optimal ansatz is a highly challenging process that depends on the specific problem being considered [35, 36]. In light of the vast number of potential configurations, identifying an optimal one may necessitate exponentially increasing resources, which could ultimately impede the potential for quantum acceleration. It is therefore imperative to investigate the impact of the ansatz on the model capabilities. The strategies that have been investigated thus far are predominantly based on classical machine learning methods. For instance, in the work [41, 42] reinforcement learning to identify an optimal ansatz for a given problem is utilized. In [43] , a supervised learning technique was employed to identify an appropriate ansatz. However, it should be noted that these methods still necessitate additional resources, as the architecture of a quantum model must be optimized even before training the model itself. The theoretical investigation of the influence of ansatz design on model performance remains an ongoing research question.

Computation on Quantum Hardware

The training of VQAs on quantum computers presents a series of challenges. It is notable that the current generation of quantum computers exhibits hardware noise, which refers to the corruption of gate operations or the quantum state itself by external sources such as electromagnetic influences or thermal fluctuations [44]. While considerable effort has been dedicated to minimizing errors and developing error-correcting schemes [45, 46], it is imperative to consider the constraints imposed by hardware when evaluating the current capabilities of quantum machine learning models. On the one hand, noise does impact the accuracy of quantum machine learning models. On the other hand, it can impact the trainability of the model as well, [47]. When the loss landscape spanned by the trainable parameters θ and defined by the loss function $\mathcal{L}(\theta)$ gets noisy, it becomes more challenging to navigate through the loss landscape to a minimum.

Secondly, as quantum mechanical measurements are a statistical process, it is necessary to execute a quantum circuit with subsequent measurement multiple times and then to average the resulting measurement data in order to obtain an estimate of the property being measured. Each run is referred to as a shot. Due to the dependence on multiple circuit evaluations, noise resulting from finite sampling is introduced. The uncertainty of a measured expectation value scales with $1/\sqrt{N}$, where N is the number of shots used to estimate the expectation value [48]. Therefore, a sufficient number of shots must be performed to estimate an expectation value with sufficient accuracy, in order to enable the training of a quantum machine learning model.

A further challenge in applying quantum machine learning models to quantum hardware is the calculation of the gradients of the trainable parameters during the training procedure. In contrast to classical ML, it is not feasible to directly calculate the gradient of each trainable parameter via backpropagation. In order to calculate the gradient of parameters in VQCs, the parameter shift rule can be applied [49, 50]. This method allows for the calculation of the gradient of the expectation value $f(x, \theta)$ with respect to each parameter θ_i within the set of parameters θ. When a parameter θ_i is shifted by some value $h \in \mathbb{R}$, the gradient in direction $\hat{e}_i$ can be determined by

$$\frac{\partial f(x,\theta)}{\partial \theta_i} = \frac{f(x, \theta + \hat{e}_i h) - f(x, \theta - \hat{e}_i h)}{2\sin(h)} \,. \tag{2.43}$$

It should be noted that in order to train a quantum machine learning model, it is necessary to determine the expectation values $f(x, \theta + \hat{e}_i h)$ and $f(x, \theta - \hat{e}_i h)$ for each parameter in the set of parameters, which requires the use of quantum computing resources that scale linearly with the number of parameters in the model.

Barren Plateaus

Barren plateaus are a phenomenon observed in VQAs, where the loss landscape, defined by the loss function $\mathcal{L}(\theta)$ and spanned by the quantum parameters θ, becomes exponentially flat as the system size increases [51]. This presents a challenge, as exponentially more resources, like exponentially more shots due to noise of finite sampling, are required during training to find an optimal solution. Since large system sizes with many qubits, and consequently with an exponentially large Hilbert spaces, are of interest, barren plateaus could undermine the potential quantum speedup. In recent years, significant research has been conducted on the causes

leading to barren plateaus [52–54]. For instance, a relationship between model expressivity and the occurrence of barren plateaus has been established [55]. Additionally, a algorithm has been developed to identify architectures that avoid barren plateaus [56]. However, recent findings, published during the course of this thesis, suggest that models processing classical data and avoiding barren plateaus are classically simulable [57]. This implies that such quantum models can be simulated with classical resources in polynomial time, challenging the advantage of using VQAs for classical data learning. This insight encourages exploration of quantum models beyond VQAs, such as quantum reservoirs, which will be introduced in the following section.

2.4 Reservoir Computing

In the previous chapters, we introduced the concepts of ML and its advanced quantum variant. Training these models can become computationally expensive, particularly when they involve a large number of trainable parameters. This challenge is especially true for VQAs, where numerous iterations are required to calculate weight updates. To mitigate these computational demands, one can, in appropriate cases, employ methods such as extreme learning machines (ELMs) [58] or reservoir computing (RC) [59, 60]. These approaches aim to map data $\mathbf{x}$ into a high-dimensional representation, where a readout layer is trained to associate this high-dimensional state with the corresponding labels $\mathbf{y}$. Reservoir computing is a subset of extreme learning machines and involves an internal reservoir that can process sequential input. The dimension of the quantum state of a quantum circuit scales exponentially with the number of qubits, making it reasonable to use quantum circuits as reservoirs. This allows data to be projected into a system with inherently large dimensionality. This concept is known as quantum reservoir computing (QRC).

In the following, we begin by formally introducing the concept of RC and explore potential physical systems that can function as reservoirs. We then proceed to outline the principles of QRC, positioning this approach within the broader context of QML, as introduced in the previous section. Notation und Introduction in the following section adhere to the textbook "Reservoir Computing: Theory, Physical Implementations and Applications" by Nakajima and Fischer [61].

2.4.1 Principles of Reservoir Computing

Reservoir computing is a framework designed to process sequential input data $\mathbf{x} = \{x_t\}_{t=1}^{l}$ with corresponding target outputs $\mathbf{y} = \{y_t\}_{t=1}^{l}$. The input data is fed into a high-dimensional representation in a recurrent manner. The evolution of the reservoir state s_t at time t is governed by the function $\mathcal{F}$. The progression of the reservoir state is expressed as

$$s_t = \mathcal{F}(s_{t-1}, x_t)\,. \tag{2.44}$$

To map the reservoir state to the target output $\{y_t\}_{t=1}^{l}$, measurements of the reservoir state $m(s_t)$ are first conducted. These measurements are then processed by a neural network, which often is just a simple linear layer. In the case of a linear neural network with weights W_{out}^{T}, this results in

$$\hat{y}_t = W_{out}^{T} m(s_t)\,. \tag{2.45}$$

The neural network is trained such that that the outputs $\hat{y}_t$ approximate the target outputs y_t.

Applications for reservoir computing range from pattern classification and generation over sequence filtering and control to time series forecasting [62].

Reservoir computing can be implemented on classical computers, where the high-dimensional reservoir is represented by a large vector. However, the pursuit of more powerful or energy-efficient computing resources has led to the exploration of physical systems as reservoirs. In recent decades, systems such as electronic circuits [63, 64], particle swarms [65, 66], and photonic systems like lasers [67, 68] have been investigated as potential candidates for physical reservoirs. In this context, QRC has been proposed [69, 70], which we will explore in the following.

2.4.2 Quantum Reservoir Computing

In the context of QRC, sequential input data $\mathbf{x} = \{x_t\}_{t=1}^{l}$ with corresponding target outputs $\mathbf{y} = \{y_t\}_{t=1}^{l}$ is encoded into a quantum state. This encoding can be achieved using methods such as amplitude or angle encoding, as explained in the previous section. The quantum reservoir evolves according to a unitary operation $U(x_t)$

$$|\Psi_t\rangle = U(x_t)\,|\Phi_{t-1}\rangle\ , \tag{2.46}$$

where $|\Psi_t\rangle$ represents the quantum state of the reservoir at time t. To map this state to the target outputs $\mathbf{y} = \{y_t\}_{t=1}^{l}$, measurements $m(|\Psi_t\rangle)$ are performed on the quantum state. Similar to general RC, these measurements are passed through a neural network. In the case of a linear layer with weights W_{out}^T this leads to

$$\hat{y}_t = W_{out}^T m(|\Psi_t\rangle)\,. \tag{2.47}$$

The neural network is trained to ensure that the outputs $\hat{y}_t$ approximate the target outputs y_t. Due to the exponential scaling of the quantum state $|\Psi_t\rangle$ with the number of qubits, quantum systems are inherently promising candidates for physical reservoirs. Even with a small number of qubits, data can be projected into a high-dimensional state, allowing the readout layer to extract complex features.

Unlike quantum machine learning algorithms such as VQAs discussed in the previous section, QRC does not involve optimizing weights within the quantum system itself. This significantly accelerates the training process, as the quantum system is only engaged in the encoding and processing of data, and is subsequently measured. Training a quantum reservoir requires only the measurement results of the reservoir state. The optimization of weights requires classical resources, making it computational efficient. In contrast, VQAs necessitate numerous preparations and executions of the quantum system to update quantum weights. Furthermore, QRC may avoid the issue of barren plateaus, as no parameters in the quantum circuit need optimization, preventing the emergence of an exponentially vanishing loss landscape.

These practical advantages justify the further exploration of QRC, as undertaken in this thesis.

Chapter 3

Methodology of Benchmarking Time Series (Quantum) Models

3.1 Motivation

Over recent years, a variety of quantum machine learning models for time series analysis have been introduced [13–17]. These models are often inspired by classical ML methods and typically have a classical counterpart. While it has been demonstrated that these approaches can successfully predict time series data, a comprehensive comparison of these models against each other is still lacking. Moreover, an extensive benchmark of these models is particularly important for evaluating the capabilities of quantum models relative to classical methods. Such a benchmark is valuable as it can reveal potential advantages of employing quantum models over classical ones.

In this work, we align with the recent trend in quantum machine learning research, where existing models are systematically compared with their classical counterparts. While such comparisons have been conducted for classification tasks [71], this work aims to benchmark the time series prediction capabilities of quantum machine learning models.

This chapter is structured as follows: We begin by introducing the method of time series prediction employed in this thesis. Second, we review the datasets utilized for the benchmark. Next, we present the classical and quantum models for time series prediction under investigation. Finally, we discuss the technical aspects of the benchmark, including the measures of prediction accuracy and the hyperparameters used.

3.2 Concept of Time Series Prediction

In the following we introduce the method of time series prediction employed in this thesis. Throughout this work, we address discrete time series datasets of length n

$$\{\mathbf{x}_i\}, \quad i = 1, 2, \ldots, n. \tag{3.1}$$

The values at each point in time are generally d-dimensional, $\mathbf{x}_i \in \mathbb{R}^d$. To train a supervised machine learning model, as introduced in Chapter 2, we divide the dataset into sequences of length l consisting of consecutive data points

$$\{[\mathbf{x}_1, \ldots, \mathbf{x}_{l-1}], [\mathbf{x}_2, \ldots, \mathbf{x}_l], \ldots, [\mathbf{x}_{n-l+1}, \ldots, \mathbf{x}_n]\} \ . \tag{3.2}$$

This approach is also known as sliding window method [72]. The models are trained to learn the mapping f of these sequences $[\mathbf{x}_{t-l+1}, \ldots, \mathbf{x}_t]$ onto a data point $\mathbf{x}_{t+k}$,

which is k steps ahead in the sequence

$$f([\mathbf{x}_{t-l+1}, \ldots, \mathbf{x}_t]) = \mathbf{x}_{t+k}\,. \tag{3.3}$$

It is important to note that the difficulty of the learning task can be adjusted by varying the prediction step k. For small values of k, the function to be learned is typically highly linear. Increasing k introduces more nonlinearity into the problem, thereby increasing the complexity, which is crucial for finding a adequate difficulty for benchmarking different models. Introducing a prediction step requires truncating the sequences at the end of the consecutive sequences in Equation (3.2), as otherwise, the data point to be predicted would fall outside the dataset. For training and evaluating the machine learning models, we ultimately obtain tuples consisting of sequences as inputs and their corresponding labels as outputs of the model

$$\hat{X} = \{([\mathbf{x}_1, \ldots, \mathbf{x}_{l-1}], \mathbf{x}_{l-1+k}), \ldots, ([\mathbf{x}_{n-k-l+1}, \ldots, \mathbf{x}_{n-k}], \mathbf{x}_n)\}\,. \tag{3.4}$$

3.3 Data

A critical component of conducting a representative and informative benchmark study is the selection of appropriate datasets. These datasets must exhibit sufficient complexity to allow models to learn advanced features beyond simple linear mappings or periodic correlations. Therefore, this benchmark utilizes datasets derived from dynamical systems that display chaotic behavior. A dynamical system is classified as chaotic if it exhibits extreme sensitivity to even minor changes in its initial conditions [73]. In such systems, small perturbations can result in vastly different dynamical outcomes. Consequently, these systems are characterized by high nonperiodicity and nonlinearity, making them ideal for benchmarking time series machine learning models.

To quantify the complexity of chaotic time series, a measure of chaoticity is essential. One such measure is the Lyapunov exponent [74], which quantifies the rate at which trajectories with two neighboring initial conditions, x_0 and $x_0 + \epsilon$, diverge. Consider a one-dimensional map describing the dynamical system as $x_{n+1} = f(x_n)$. Applying the map N times gives $x_0 \to f^N(x_0)$ and $x_0 + \epsilon \to f^N(x_0 + \epsilon)$. The Lyapunov exponent λ is then defined as

$$\lim_{N\to\infty} e^{N\lambda} = \lim_{N\to\infty} \frac{|f^N(x_0 + \epsilon) - f^N(x_0)|}{\epsilon}\,. \tag{3.5}$$

The Lyapunov time τ, defined as the inverse of the Lyapunov exponent,

$$\tau = \frac{1}{\lambda} \tag{3.6}$$

provides a characteristic time scale over which a dynamical system exhibits chaotic behavior. Examples of chaotic systems include the Mackey-Glass equation [75], the Lorenz Attractor [76], the Rössler Attractor [77], and the Hénon Attractor [78]. In this benchmark, we will focus on the one-dimensional Mackey-Glass equation and the Lorenz Attractor. In the following, we will introduce these chaotic systems.

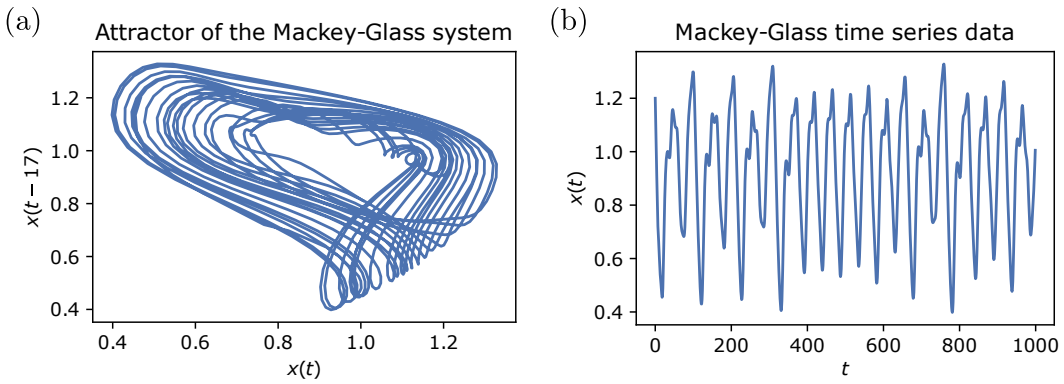

FIGURE 3.1: (a): Attractor trajectory $(x(t), x(t-17)$ of the Mackey-Glass system. The trajectory does dot follow stable paths and which implies chaotic behavior. (b): Mackey-Glass time series dataset as it us used throughout this thesis plotted over the time steps t.

3.3.1 Mackey-Glass Equation

The Mackey-Glass Equation [75] is a one-dimensional delayed differential equation given by

$$\frac{\mathrm{d}x(t)}{\mathrm{d}t} = \frac{\alpha x(t-\tilde{t})}{1+P(t-\tilde{t})^n} - \gamma x(t)\,. \tag{3.7}$$

In this equation, α, γ, n, and $\tilde{t}$ are system parameters. The solution $x(t)$ describes the Mackey-Glass time series. Chaotic behavior can emerge by selecting appropriate parameters. This is the case for $\alpha = 0.2$, $\gamma = 0.1$, $n = 10$, and $\tilde{t} = 17$. We set the initial condition to $x(t=0) = 1.2$. To solve Equation (3.7), we use the fourth-order Runge-Kutta method [79, 80] with a step size of 1.

Unless stated otherwise, we use the first 1000 data points from the time series for the experiments in this thesis. We limit the dataset to this size due to the computational expense of simulating the quantum machine learning models employed. To ensure that this number of data points is sufficient for meaningful feature learning, we compute the Lyapunov time τ of the dataset. Using the method of Eckmann et al. [81], we find a minimal Lyapunov time of approximately $\tau = 20$ time steps between distinct data points. This value is relatively small compared to the length of the entire dataset, indicating that chaotic behavior, as well as nonlinearity and nonperiodicity, is present throughout the dataset.

This chaotic behavior is illustrated by plotting the attractor trajectory $(x(t), x(t-17))$ in Figure 3.1 (a). The trajectory does not follow stable paths, indicating chaotic behavior. Additionally, Figure 3.1 (b) shows the data $x(t)$ plotted against the time steps t.

3.3.2 Lorenz Attractor

The second dataset in the benchmark study presented in this thesis is the Lorenz Attractor dataset [76]. This three-dimensional system is governed by a set of three

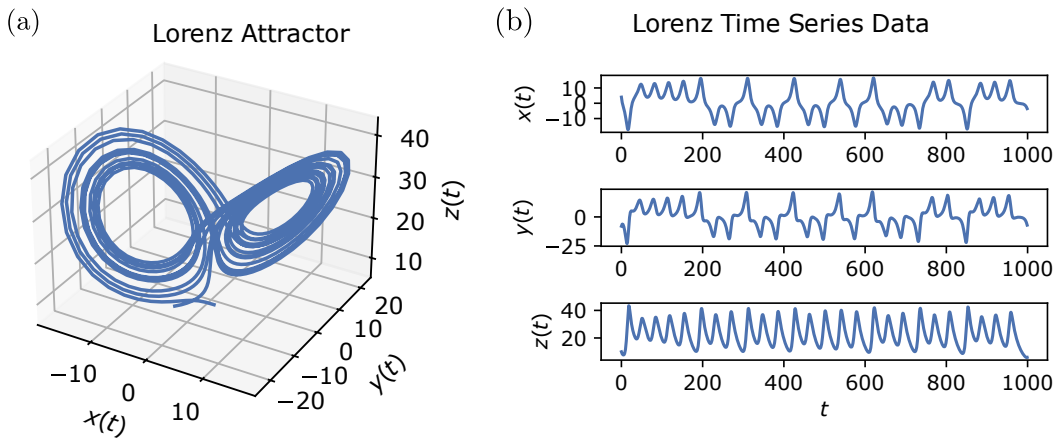

FIGURE 3.2: (a): Attractor trajectory $(x(t), y(t), z(t))$ of the Lorenz system. The unstable trajectory shows the chaoticity of the data. (b): Plots of the single coordinate functions $x(t)$, $y(t)$, and $z(t)$ of the Lorenz Attractor dataset used in this thesis.

ordinary coupled differential equations

$$\begin{aligned} \frac{\mathrm{d}x(t)}{\mathrm{d}t} &= \sigma(y - x)\,, \\ \frac{\mathrm{d}y(t)}{\mathrm{d}t} &= x(\rho - z) - y\,, \\ \frac{\mathrm{d}z(t)}{\mathrm{d}t} &= xy - \beta z\,. \end{aligned} \tag{3.8}$$

Here, σ, ρ, and β are parameters, and $(x(t), y(t), z(t))$ describe the system's trajectory. For $\sigma = 10$, $\rho = 28$, and $\beta = 8/3$, the system exhibits chaotic behavior. We use initial conditions $(x(t = 0), y(t = 0), z(t = 0)) = (4, -8, 10)$ and numerically solve the differential equations using the fourth-order Runge-Kutta method [79, 80] with a step size of 0.02.

As for the Mackey-Glass dataset, we use the first 1000 data points from the time series throughout this thesis unless specified otherwise. The minimal Lyapunov time for this dataset is calculated using the method of Eckmann et al. [81] and is approximately $\tau = 10$ in units of time steps between data points. This indicates that the dataset is sufficiently large to capture the timescale of chaotic behavior throughout the series.

The attractor trajectory $(x(t), y(t), z(t))$ for the dataset is shown in Figure 3.2 (a). The unstable trajectory demonstrates the chaotic nature of the data. The functions $x(t)$, $y(t)$, and $z(t)$ for the individual coordinates are plotted in Figure 3.2 (b).

3.4 Classical Machine Learning Models for Time Series Prediction

In this section, we introduce the classical machine learning models used in the benchmark study of this thesis. They are of crucial importance for the benchmark study as they set a baseline for the capabilities of the quantum machine learning models.

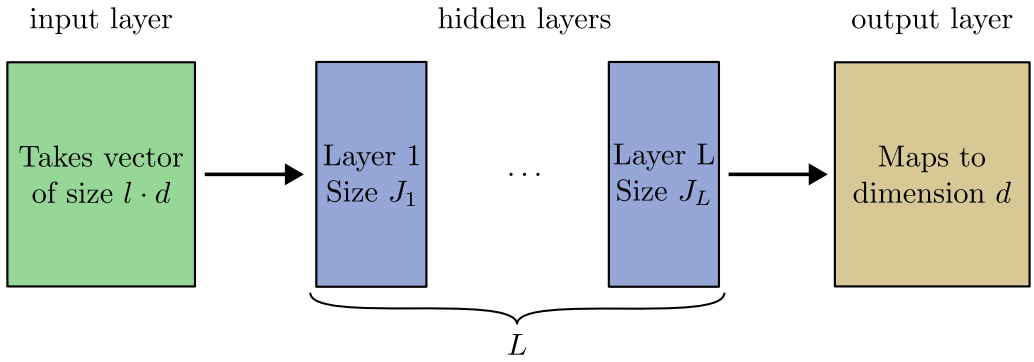

FIGURE 3.3: Sketch of the Multi-Layer Perceptron framework. An input vector of size $l \cdot d$ is mapped by the input layer into a hidden state. The state is passed through hidden layers of flexible sizes J_i and depth L before it is mapped to the dimension of the value in the time series that is being predicted.

3.4.1 Multi-Layer Perceptron

A simple classical machine learning model is the Multi-Layer Perceptron (MLP), introduced in Chapter 2 and illustrated in Figure 2.3 (b). To perform time series predictions, a time sequence $[\mathbf{x}_{t-l+1}, \ldots, \mathbf{x}_t]$ of length l is fed into the input layer. In the case of a multidimensional time series with $d \geq 2$, the input vector is flattened. Generally, the input layer maps a vector of size $l \cdot d$ to the size of the first hidden layer. As indicated in Figure 3.3, there is considerable flexibility in constructing the hidden layers. The number of hidden layers L and the size of each hidden layer J_i are hyperparameters of the model. In the benchmark, we typically evaluate a range of hidden layer configurations to identify regions in the hyperparameter space where the model performs well for a given problem. The sigmoid function and the hyperbolic tangent are used as activation functions between layers throughout this thesis. The output layer maps the last hidden state to the dimension d of the data point in the time series that is being predicted. Note that this architecture does not inherently account for the temporal structure of the data. However, it provides an initial baseline for the capabilities of classical neural networks, against which quantum models can be compared in the benchmark study.

3.4.2 Recurrent Neural Network

In contrast to the MLP, the Recurrent Neural Network (RNN) [29, 30] does account for sequential structure of data. Over the past decades, various RNN variants, such as Stacked RNNs and Bidirectional RNNs, have been developed [82]. However, since the goal of introducing RNNs in this thesis is to establish a baseline for comparison with quantum models, we will start with the simplest RNN structure as originally introduced. The concept of an RNN is illustrated in Figure 3.4. The time sequence $[\mathbf{x}_{t-l+1}, \ldots, \mathbf{x}_t]$ is fed sequential through a linear input layer into a hidden cell. This cell consists of a neural network with a variable number of layers and a variable hidden size. The hidden state h_i is passed forward to the next time step. The weights and bias of the input layer $\mathbf{W}_{ih}$ and $\mathbf{b}_{ih}$, as well as those of the hidden cell,$\mathbf{W}_{hh}$ and $\mathbf{b}_{hh}$, are shared across all hidden cells of all time steps. In this work, we use the hyperbolic tangent activation function. Based on the previous hidden state

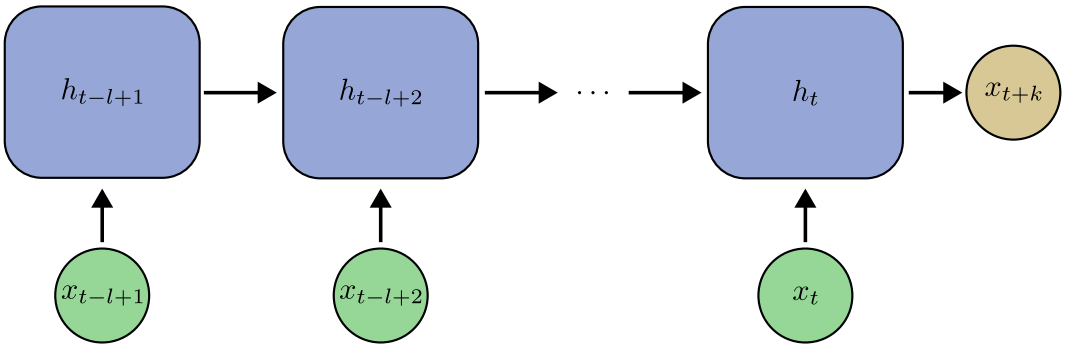

FIGURE 3.4: Illustration of the Recurrent Neural Network architecture. Time sequences are fed into hidden cells sequentially. The hidden state propagates throughout the sequence. To ensure compatibility with the dimensions required for time series prediction, the final hidden state undergoes a mapping process in an output layer.

vector $\mathbf{h}_{t-1}$ and the input $\mathbf{x}_t$ at time t, the output of the cell is given by

$$\mathbf{h}_t = \tanh\left(\mathbf{x}_t \mathbf{W}_{ih}^T + \mathbf{b}_{ih} + \mathbf{h}_{t-1}\mathbf{W}_{hh}^T + \mathbf{b}_{hh}\right). \tag{3.9}$$

After the final time step of the sequence has been processed by the last cell, a linear output layer maps the last hidden state $\mathbf{h}_t$ to the time series data point that needs to be predicted, $\mathbf{x}_{t+k}$.

3.4.3 Long Short-Term Memory

The Long Short-Term Memory (LSTM) network is an advanced version of a Recurrent Neural Network (RNN) [22], specifically designed to capture longer-range dependencies in sequential data. It represents a state-of-the-art model for tasks such as pattern classification, sequence generation, and time series prediction [83]. Similar to the RNN illustration in Figure 3.4, data is sequentially fed into the LSTM cells. The architecture of an LSTM cell is depicted in Figure 3.5. A key addition in the LSTM is the cell state $\mathbf{c}_t$, which is propagated along the sequence. While the hidden state $\mathbf{h}_t$ is designed to track short-term dependencies in the sequence, the cell state $\mathbf{c}_t$ captures long-term dependencies. This distinction is the basis for the name Long Short-Term Memory.

At each time step, the input sequence element $\mathbf{x}_t$ is concatenated with the hidden state from the previous cell, $\mathbf{h}_{t-1}$, forming a vector $\mathbf{v}_t$. The cell contains several gates, which are linear neural networks with specific activation functions, each fulfilling a distinct role within the LSTM architecture. The forget gate, parameterized by weights $\mathbf{W}_f$, processes the input $\mathbf{v}_t$ to determine how much of the cell state $\mathbf{c}_t$ should be retained. The input gate with weights $\mathbf{W}_i$ and the input node with weights $\mathbf{W}_g$ together modulate the updates to the cell state at the current time step. Finally, the output gate with weights $\mathbf{W}_o$ processes the input vector $\mathbf{v}_t$ to update the hidden state $\mathbf{h}_t$. The following equations describe these operations mathematically, where

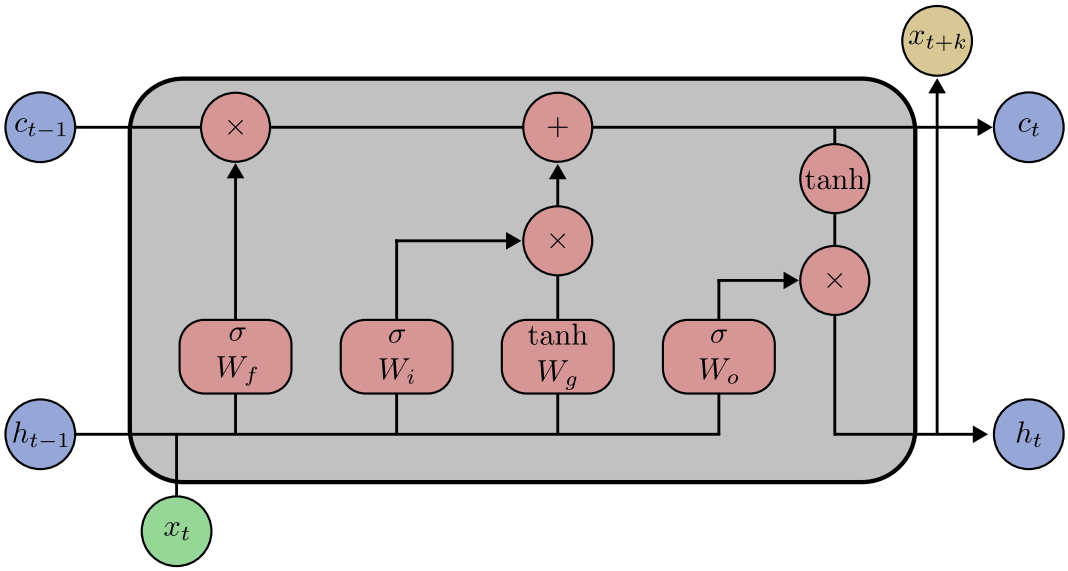

FIGURE 3.5: The structure of a Long Short-Term Memory cell. A hidden state $\mathbf{h}_t$ and a cell state $\mathbf{c}_t$ are propagated through a sequence. Within the cell, four gates update these two states based on the concatenated input of the current data point $\mathbf{x}t$ and the previous hidden state $\mathbf{h}_{t-1}$. A prediction for a future data point can be made by applying a linear layer to the hidden state of the final cell in the sequence.

$\mathbf{b}_f$, $\mathbf{b}_i$, $\mathbf{b}_g$, and $\mathbf{b}_o$ represent the biases of the respective linear layers:

$$\begin{aligned}
\mathbf{f}_t &= \sigma\left(\mathbf{v}_t\mathbf{W}_f^T + \mathbf{b}_f\right) \\
\mathbf{i}_t &= \sigma\left(\mathbf{v}_t\mathbf{W}_i^T + \mathbf{b}_i\right) \\
\mathbf{g}_t &= \tanh\left(\mathbf{v}_t\mathbf{W}_g^T + \mathbf{b}_g\right) \\
\mathbf{c}_t &= \mathbf{f}_t \times \mathbf{c}_{t-1} + \mathbf{i}_t \times \mathbf{g}_t \\
\mathbf{o}_t &= \sigma\left(\mathbf{v}_t\mathbf{W}_o^T + \mathbf{b}_o\right) \\
\mathbf{h}_t &= \mathbf{o}_t \times \tanh\left(\mathbf{c}_t\right)
\end{aligned} \tag{3.10}$$

Note that the symbol $\times$ denotes element-wise multiplication. As with the RNN introduced earlier, the weights of an LSTM cell are shared across all cells in the sequence. To make a prediction from a given sequence $[\mathbf{x}_{t-l+1}, \ldots, \mathbf{x}_t]$ for a future data point $\mathbf{x}_{t+k}$, the hidden state of the last cell can be mapped to the dimension of $\mathbf{x}_{t+k}$ using a linear layer. The initial choice of the hidden states $\mathbf{h}_t$ and $\mathbf{c}_t$ can be arbitrary, typically one sets these vectors initially to zero.

The architecture can be extended by introducing a layered structure within each cell, where multiple cells with different weights are stacked for a single data point in the sequence. This approach enhances the model's expressiveness. In the layered architecture, the number of layers is consistent across all data points, and the weights remain shared across the sequence. The number of layers is a hyperparameter of the model, as is the size of the hidden states $\mathbf{h}_t$ and $\mathbf{c}_t$. In the benchmark, we perform a parameter scan over these two hyperparameters to identify regions in the hyperparameter space where the model achieves optimal performance for a given problem.

Classical Model	Quantum Model
Multi-Layer Perceptron (MLP)	Variational Quantum Circuit (VQC)
Recurrent Neural Network (RNN)	Quantum Recurrent Neural Network (QRNN)
Long Short-Term Memory (LSTM)	Quantum Long Short-Term Memory (QLSTM)
Extreme Learning Machine (ELM)	Quantum Reservoir Computing (QRC)

TABLE 3.1: Quantum machine learning models employed in the benchmark of this thesis with their corresponding classical counterpart.

3.4.4 Extreme Learning Machine

We include an extreme learning machine (ELM) model [58] in our benchmark as a comparison to the quantum reservoir computing approach introduced at the end of the next section. The core idea of ELM is to project the input data into a higher dimensional space to obtain a transformed representation. An output layer is then trained to map this high-dimensional representation to the target dimension for a given learning task. For sequential data, we map the $l \cdot d$ values of a sequence $[\mathbf{x}t - l + 1, \ldots, \mathbf{x}t]$ of length l and dimension d into a higher dimensional space via a linear layer followed by a hyperbolic tangent activation function. This linear layer is fixed after initialisation and does not undergo any further training. The dimensionality of the higher dimensional space is a model hyperparameter.

In the next section, when discussing the QRC approach, we will clarify that the process involves measuring a set of random observables on the reservoir. To ensure comparability with the quantum model, the higher-dimensional state is projected to a dimension corresponding to the number of measurements made on the reservoir in the quantum case. This is again done by a randomly initialised but fixed linear layer. The number of measurements acts as another hyperparameter of the model. Finally, the measured values are used to train a separate linear layer that maps the output from the number of measurements to the dimension d, allowing the model to predict the values in the time series being learned.

3.5 Quantum Machine Learning Models for Time Series Prediction

Having introduced the classical machine learning models used in this thesis benchmark, we now shift our focus to their quantum counterparts. Each classical model is chosen to correspond to a specific quantum machine learning model, with these pairings shown in Table 3.1. We note here, that a VQC differentiates from a MLP as a VQC doesn't have any perceptron like structure. However, the two models are similar in how they are trained and what they are used for. The first three quantum models and their internal architectures are based on the works of [13–17]. Throughout the benchmark, both the original architectures proposed in these papers and well-justified modifications are employed. Here, we present the original architectures as described in the papers, along with the modifications applied in the benchmark. Additionally, we introduce a quantum reservoir computing approach developed specifically for time series forecasting and compare it to a classical extreme learning machine.

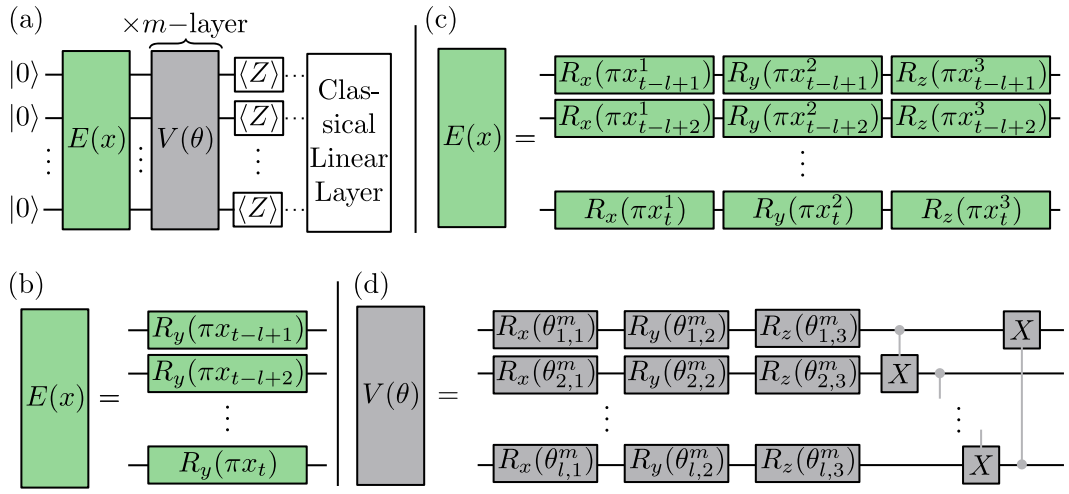

FIGURE 3.6: (a): Structure of the VQC employed in this thesis. A time sequence is encoded in an encoding layer $E(x)$, while m variational layers incorporate trainable weights into the model. The expectation values of Pauli-Z observables on all qubits are measured, and the results are passed through a classical linear layer to map the number of qubits to the dimension of the data point to be predicted. (b): Circuit of the encoding block $E(x)$. Data points in the sequence are encoded onto individual qubits using angle encoding. (c): For three-dimensional data, three rotation gates are used to encode the three dimensions of a data point onto a single qubit. (d): Circuit of a variational layer $V(\theta)$. Each layer has independent weights $\theta^m_{i,j}$.

3.5.1 Variational Quantum Circuit

We introduced the concept of VQCs in Chapter 2. In the following section, we illustrate how these circuits can be employed for time series forecasting tasks. Within the scope of our benchmark, we primarily reference the work of Rivera-Ruiz et al. [13], which compares the time series prediction accuracy of specific VQC architectures to that of classical models such as MLPs and LSTMs. To ensure comparability between our benchmark results and those presented in [13], we use identical architectures, which we will discuss, along with justified modifications to these models. Although the referenced paper overlaps with our work in evaluating the time series prediction capabilities of VQCs, this thesis extends the benchmark by incorporating additional quantum models and more advanced versions of the VQC.

The overall structure of the VQC is depicted in Figure 3.6 (a). As discussed in Chapter 2, an encoding block $E(x)$ is responsible for mapping data into a quantum state, while a variational circuit contains parameters that are iteratively optimized during the training process. To encode a sequence of time series data into a quantum state, the paper utilizes angle encoding, where each data point in the sequence is mapped onto an individual qubit. Specifically, for one-dimensional data, a sequence $[x_{t-l+1}, \ldots, x_t]$ of length l is encoded onto l qubits, with each data point x_i encoded onto qubit i through an R_y rotation

$$R_y(\pi x_i)\,. \tag{3.11}$$

As will be discussed at the end of this chapter, all data is scaled to the interval $[0, 1]$. Therefore, we multiply the encoded value by π to ensure the encoding corresponds

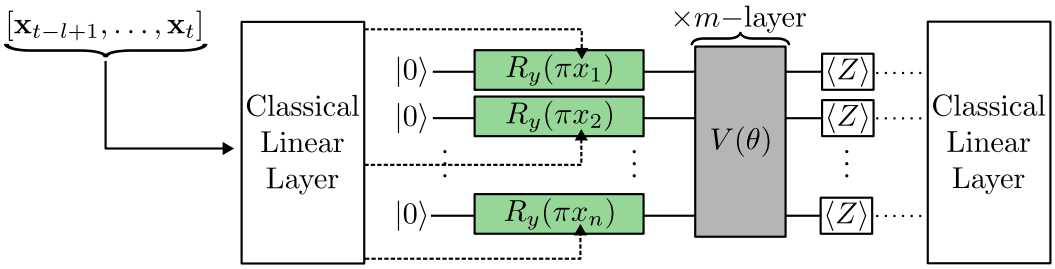

FIGURE 3.7: Architecture of the variational quantum circuit with a classical linear input layer. Each data point in the sequence is passed through the input layer and mapped to an n-dimensional representation, which is then encoded onto n qubits. This approach decouples the number of qubits from the sequence length, making them independent parameters.

to a $[0, \pi]$ rotation on the Bloch sphere. The encoding circuit diagram is illustrated in Figure 3.6 (b). Since we are dealing with not only one-dimensional data but also three-dimensional data throughout the benchmark, this encoding strategy must be adapted. The angle encoding is extended to encode all three dimensions of a data point $\mathbf{x_i} = (x_i^1, x_i^2, x_i^3)$ onto qubit i using three rotations around the orthogonal axes on the Bloch sphere

$$R_z(\pi x_i^3) R_y(\pi x_i^2) R_x(\pi x_i^1)\,. \tag{3.12}$$

The three-dimensional sequence encoding is shown in Figure 3.6 (c). It is worth noting that this type of encoding is similar to the strategies employed by other quantum models discussed in this thesis. Once the sequence is encoded into a quantum state, the variational layer $V(\theta)$ is applied. As suggested in [13], the structure consists of m such layers, where each layer includes rotation gates with weights $\theta_{i,j}^m$ followed by nearest-neighbor entangling $CNOT$ gates, as depicted in Figure 3.6 (d). On qubit i and in layer m, the rotation gates are given by

$$R_z(\theta_{i,3}^m) R_y(\theta_{i,2}^m) R_x(\theta_{i,1}^m)\,. \tag{3.13}$$

Before training, the weights $\theta_{i,j}^m$ are randomly sampled from a uniform distribution in $[0, 2\pi]$. After the application of the variational layers $V(\theta)$, the quantum state is measured, and the expectation values of the Pauli-Z observable are obtained for all qubits. Unlike the approach in [13], where only the Pauli-Z expectation value of the first qubit is measured to predict the next data point, we measure the expectation values of all qubits. These results are then fed into a classical linear layer, which maps the measurements to the dimensionality of the data point to be predicted. This ensures an equal and therefore more comparable treatment of datasets with different dimensions.

One limitation of the encoding architecture discussed above is that the required number of qubits scales with the sequence length. To increase flexibility in choosing the sequence length and the number of qubits, one approach is to pass the sequence $[\mathbf{x}_{t-l+1}, \ldots, \mathbf{x}_t]$ through a classical linear layer prior to encoding. This method, known as a dressed variational quantum circuit (DVQC) [84], is also utilized in [13]. The concept is illustrated in Figure 3.7. The sequence of length l and dimension d is mapped from the input dimension $l \cdot d$ to the target dimension n, which corresponds to the number of qubits. Each element in the target dimension is then encoded as a rotation around the y-axis in the Bloch sphere representation.

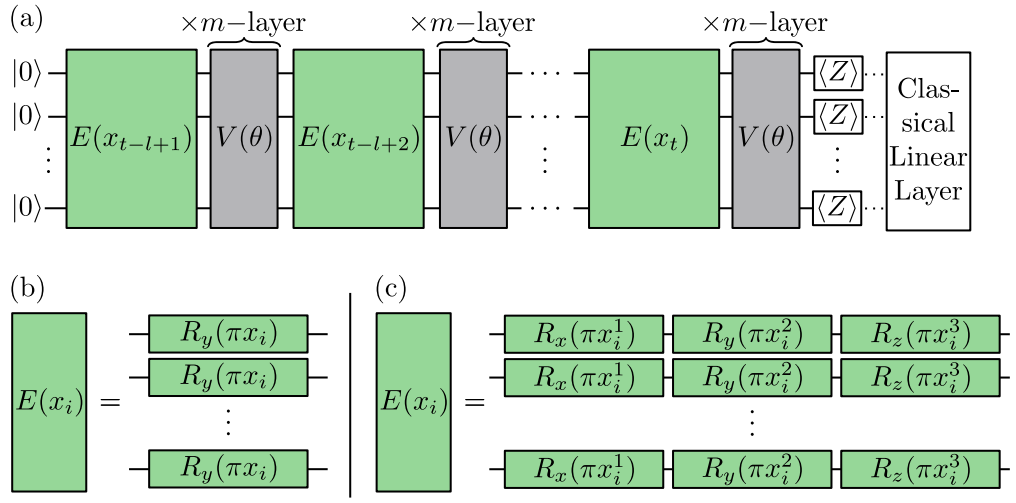

FIGURE 3.8: (a): Data re-uploading architecture of a VQC. The data points in a sequence are recurrently encoded, with each point being applied blockwise across all qubits. These encoding blocks are interspersed with variational layers $V(\theta)$. This approach decouples the number of qubits from the sequence length, allowing them to be treated as independent parameters. Furthermore, the data re-uploading structure enhances the expressivity of the circuit. (b): Encoding block for one-dimensional data. (c): Encoding block for three-dimensional data.

The subsequent variational layers and measurements are identical to those shown in Figure 3.6.

An alternative approach to addressing the issue of the increasing number of required qubits with sequence length is to encode the sequence in a recurrent manner. Instead of mapping each data point in the sequence to a separate qubit, we utilize a data re-uploading structure as introduced in 2.3 and depicted in Figure 2.7 (b). his method not only provides greater flexibility in selecting hyperparameters such as sequence length and number of qubits, but also enhances the expressivity of the quantum model [34]. The architecture is illustrated in Figure 3.8 (a). Data points $\mathbf{x}_i$ in the sequence $[\mathbf{x}_{t-l+1}, \ldots, \mathbf{x}_t]$ are individually encoded in an encoding layer $E(\mathbf{x}_i)$ using R_y rotations that correspond to the value of the data point across all qubits. These encoding layers alternate with variational layers $V(\theta)$ until all data points in the sequence are encoded. The encoding layers are constructed similarly to those shown in Figure 3.6 and are displayed in Figure 3.8 (b) and (c) for one-dimensional and three-dimensional data, respectively. The m variational layers $V(\theta)$ are identical to the one shown in Figure 3.6 (d) and each has independent weights. The subsequent measurements and classical post-processing are identical to those shown in Figure 3.6.

These three VQC architectures discussed above are part of the benchmark analysis presented in this thesis. In general, hyperparameters of the VQC models are the number of qubits n and the number of layers of the variational layer $V(\theta)$. All quantum machine learning models presented in this section, as well as the ones in the following sections are realized and simulated using the Python library PennyLane [85].

We conclude this section with a few final remarks. First, by introducing classical

linear layers, we transform the models from purely quantum machine learning models to quantum-classical hybrid models. Nonetheless, we continue to refer to them as quantum models because the number of trainable classical parameters is kept small relative to the number of trainable quantum parameters. Furthermore, as previously discussed in 2.3, the choice of architecture is highly non-trivial. As demonstrated in this section, there is a wide range of methods for encoding sequential data onto a VQC. The variety of possible ansätze for the variational layer is even greater. Unless stated otherwise, we consistently use the ansatz shown in Figure 3.6 (d) for VQC architectures throughout this thesis. This choice provides a baseline for comparing our results with those of [13] and adopts concepts popular in the literature [34]. The search for an optimal ansatz for a given learning problem is a research area in its own right [41–43] and is beyond the scope of this thesis.

3.5.2 Quantum Recurrent Neural Network

While the VQC can be considered a quantum counterpart to a classical neural network like a MLP, we now introduce the quantum equivalent of a classical Recurrent Neural Network. The Quantum Recurrent Neural Network (QRNN) was first proposed in [14]. The fundamental idea is to extend the concept of sequential learning in a VQC by allocating d qubits to a data register and h qubits to a hidden register. In this setup, a sequence of data points is encoded onto the data register in a recurrent manner. The hidden register is inspired by the classical RNN, where a hidden state is transferred from one data point to the next. In the QRNN, the quantum state of the qubits in the hidden register serves as the state that is passed along the sequence. Throughout this work, we adopt the specific architecture presented in [15], in which also numerical experiments on classical sequential data were conducted. The QRNN architecture is depicted in Figure 3.9 (a). Similar to a classical RNN, the QRNN employs a block-wise recurrent structure. In each block, a data point $\mathbf{x}_i$ from the sequence $[\mathbf{x}_{t-l+1}, \ldots, \mathbf{x}_t]$ is encoded onto the data register. We again distinguish between one- and three-dimensional data. For one-dimensional data, a data point x_i is encoded via angle encoding using a Pauli-Y rotation gate

$$R_y(\arccos(x_i)) \tag{3.14}$$

on all d qubits of the data register. The corresponding circuit diagram is shown in Figure 3.9 (b). To ensure comparability, we scale the input using the arc cosine function as proposed in [15]. Since the data is scaled to the interval $[0, 1]$, the rotation angle lies within $[0, \pi/2]$. In the case of three-dimensional data, the three dimensions of a data point are encoded sequentially with rotation gates on each qubit by

$$R_x(\arccos(x_i^3))R_y(\arccos(x_i^2)R_x(\arccos(x_i^1))\,. \tag{3.15}$$

The circuit diagram for this encoding is shown in Figure 3.9 (c). After encoding the data point into the data register, a variational layer is applied. This layer incorporates trainable weights $\theta_{i,j}$ and entangles qubits from the data register with those in the hidden register. This entanglement allows the model to learn how to transfer the quantum representation of the data into the hidden state, facilitating the extraction of relevant features from the sequence. The specific ansatz used in [15] and throughout the thesis is depicted in Figure 3.9 (d). After a layer of three parameterized rotation gates, a layer of nearest-neighbor entangling elements is applied. Each element consists of a parameterized Pauli-Z rotation gate placed between two $CNOT$ gates. It is important to note that the weights of the variational layers are shared

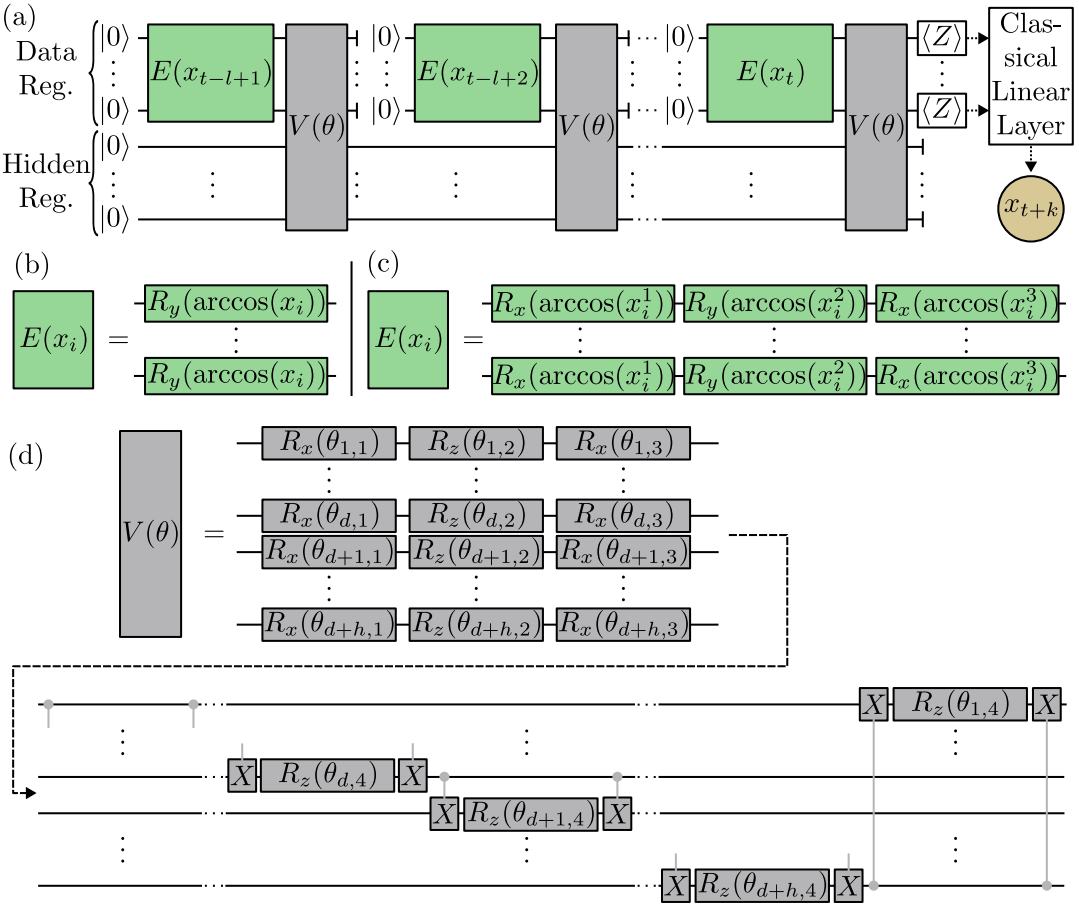

FIGURE 3.9: (a): Architecture of the Quantum Recurrent Neural Network. The circuit is divided into a data register and a hidden register. The sequence is encoded into the data register in a recurrent manner. After each encoding block, a variational layer is applied to facilitate the model's learning of the mapping from the quantum state in the data register to the hidden register. Once the final point in the sequence is encoded, the data register is measured, and the result is passed through a classical linear layer to predict the subsequent data point. (b): Encoding circuit for one-dimensional data. (c): Encoding circuit for three-dimensional data. (d): Ansatz of the variational layer used throughout the thesis.

across all cells of each time step, similar to the RNN ansatz. After each of the first $l-1$ blocks, the quantum state of the data register is reset to the ground state $|0\rangle$ to prepare for the initialization of the subsequent data point. Following the block that encodes the last data point of a sequence, the data register is measured. As with the VQC, the Pauli-Z expectation values of all d qubits are passed into a classical linear layer to map to the dimension of the data point $\mathbf{x}_{t+k}$ that is to be predicted.

The hyperparameters of the architecture include the number of qubits in the data register and the number of qubits in the hidden register. It is important noting that, for the Python library PennyLane [85] used in this thesis for simulating the training of the quantum circuits, resetting qubits is computationally expensive. Therefore, throughout the benchmark of the QRNN, both the sequence length and the number of data qubits have to be kept small to make the simulation possible. Despite these limitations, it is still possible to benchmark the QRNN against other classical and quantum models. However, for longer sequence lengths, the QRNN is excluded

from the benchmark study.

3.5.3 Quantum Long-Short Term Memory

We now introduce the quantum analogue of the LSTM, known as the Quantum Long Short-Term Memory (QLSTM), first proposed in [16]. The core idea is to replace the classical neural network layers, with weights W_f, W_i, W_g, and W_o from the classical LSTM as shown in Figure 3.5, with VQCs. The architecture of a QLSTM cell, as proposed in [16], is depicted in Figure 3.10 (a). In this architecture, the four classical neural networks are substituted by VQC_1 through VQC_4.

As in the classical LSTM, the previous hidden state h_{t-1} and the current data point $\mathbf{x}_t$ of the sequence with dimension d are concatenated into a vector $\mathbf{v}_t$. The dimension of this vector is equal to the number of qubits n in all VQCs. Thus, the hidden state dimension is defined as $h = n - d$. The vector $\mathbf{v}_t$ is processed through VQC_1 to VQC_4, where the expectation value of the Pauli-Z observable is measured on each qubit of each VQC. The outputs of these VQCs are then fed into the internal QLSTM structure, which mirrors the classical LSTM architecture. Since the outputs of VQC_1 to VQC_4 are multiplied element-wise with the cell state c_t, the dimension of c_t is also equal to the number of qubits n.

To pass a hidden state h_t to the next cell, the dimension must be reduced from n to h. This is accomplished by introducing an additional VQC_5. Although its ansatz is identical to the other VQCs, only the expectation values of the first h qubits are measured and passed on to the next QLSTM cell. As in the classical LSTM, these cells are stacked, allowing the hidden state h_t and cell state c_t to propagate through the sequence. When the last data point in the sequence is reached, an additional VQC_6 is applied. The internal n-dimensional state is passed through VQC_6, where the expectation values of all qubits are measured. Similar to the other QML models, linear layer then maps the measurement results to the dimension of the predicted data point $\mathbf{x}_{t+k}$.

The circuit architecture of all VQCs is identical and shown in Figure 3.10 (b) as it was introduced in [16] and implemented throughout the benchmark of this thesis. Initially, a Hadamard gate H is applied to all qubits, followed by encoding the i-th element of the input vector $\mathbf{v}_t$ onto the i-th qubit via

$$R_z(\arctan(v_i^2))R_y(\arctan(v_i))\,. \tag{3.16}$$

Since the data is scaled to the interval $[0, 1]$, the rotation angles for encoding lie within $[0, \pi/4]$ for VQC_1. After the encoding layer, a series of m variational layers is applied. Each layer consists of nearest-neighbor and second-nearest-neighbor $CNOT$ entanglements, followed by parameterized rotation gates defined as

$$R(\theta_{i,1}^m, \theta_{i,2}^m, \theta_{i,3}^m) = R_z(\theta_{i,3}^m)R_x(\theta_{i,2}^m)R_z(\theta_{i,1}^m) \tag{3.17}$$

for each qubit i. The weights $\theta_{i,j}^m$ are initialized by uniformly sampling from $[0, 2\pi]$ before training. These parameters are independent across different VQC_i, but remain identical for the same VQC_i across different QLSTM cells. Therefore, the total number of trainable quantum parameters is $6 \cdot 3 \cdot n \cdot m$.

A limitation of this architecture is that the size of the hidden state $h = n - d$ and the cell state $c = n$ both scale with the number of qubits n. Since the hidden and cell states are meant to store information about the sequence, it would be advantageous to have more flexibility in choosing these sizes. However, a large hidden

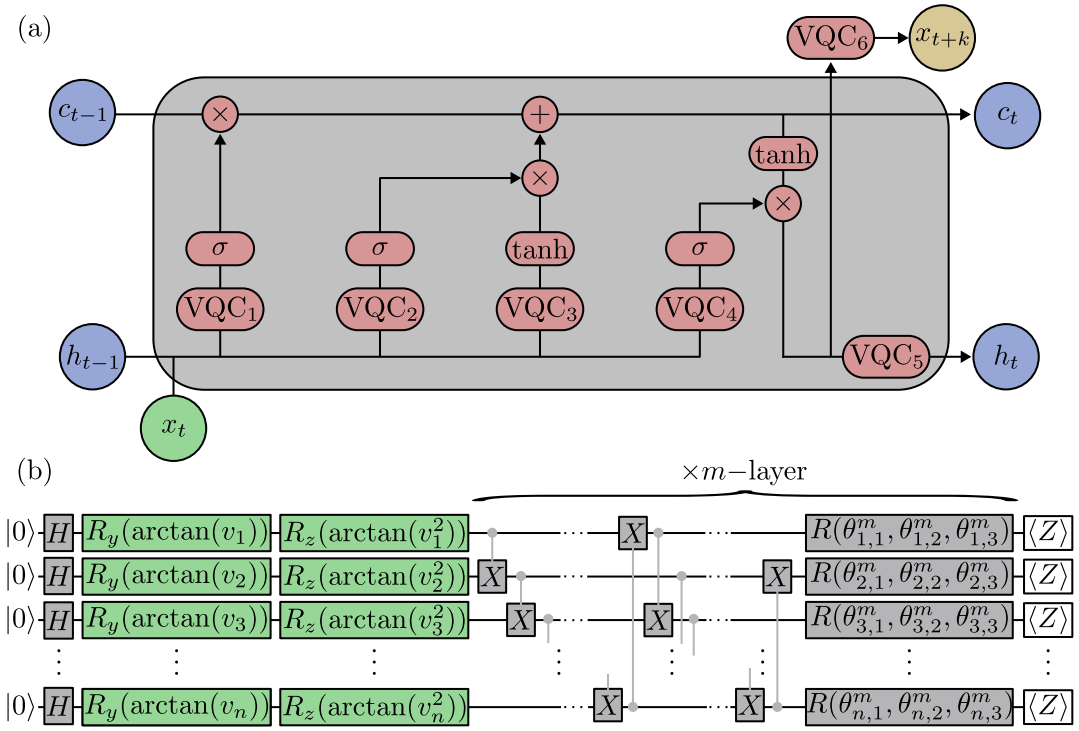

FIGURE 3.10: (a): Architecture of a QLSTM cell. In this design, the classical neural networks used in a traditional LSTM cell are replaced by VQCs. Two additional VQCs are incorporated: one to match the dimensionality of the hidden state and another to process the cell output, ensuring it is mapped appropriately to the data point that needs to be predicted. (b): Circuit diagram of the VQCs. The input vector $\mathbf{v}_t$ is encoded using angle encoding, scaled by an arc tangent function. This is followed by the application of a layered variational block. Each layer consists of entangling operations between nearest and second-nearest neighbors using *CNOT* gates, followed by a set of parameterized single-qubit rotation gates.

state requires a correspondingly large number of qubits, which can exceed the hardware capabilities of current quantum computers. Additionally, classical simulation of quantum circuits demands resources that scale exponentially with the number of qubits. Consequently, we are forced to limit the number of qubits to the single-digit range in many parts of the benchmark.

To decouple the hidden state size from the number of qubits, an advanced version of the QLSTM was proposed in [17]. This variant, called the Linear Enhanced Quantum Long Short-Term Memory (LE-QLSTM), incorporates classical linear layers into the architecture, as shown in Figure 3.11. A linear input layer maps the input vector $\mathbf{v}_t$ to the number of qubits n in VQC_1-VQC_4. After processing through these VQCs, additional linear layers map the qubit count n to the hidden and cell state size $h = c$. This eliminates the need of a VQC_5 as in the other QLSTM architecture. Similarly, VQC_6 is replaced by a linear layer that maps the hidden state size h to the dimension d of the predicted data point. In [17], as well as in the benchmarks conducted, the circuit design remains identical to that shown in Figure 3.10 (b). While this approach effectively decouples the number of qubits n from the dimensions of the hidden and cell states h and c, it also increases the number parameters beside the quantum circuit relative to the quantum parameters. This makes it challenging

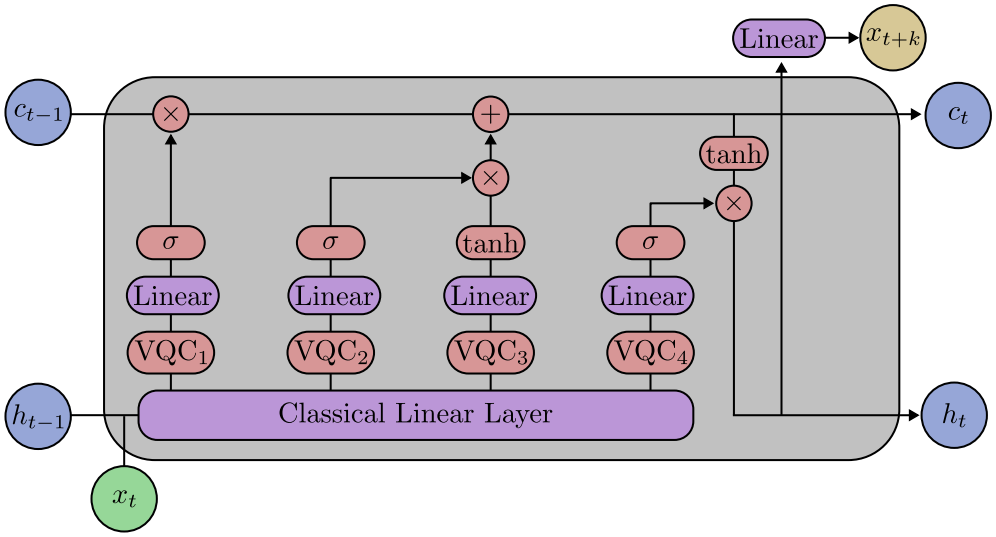

FIGURE 3.11: Structure of the Linear Enhanced Quantum Long Short-Term Memory Cell. To decouple the dimensions of the hidden and cell states from the number of qubits in the VQCs, a series of classical linear layers is inserted. A linear input layer first maps the input vector $\mathbf{v}_i$ to a dimension corresponding to the number of qubits n. Following the VQCs, another set of linear layers then maps the n measurement results to the dimension of the hidden states.

to determine the extent of the quantum neural network's influence during training compared to the contribution of the classical linear layers. This question will be explored throughout the benchmark.

In the following, we discuss limitations faced by both the QLSTM and LE-QLSTM architectures. Unlike the QRNN, the hidden state is not propagated through the sequence as a quantum state. Instead, the quantum state is reduced by measuring n observables, collapsing from a 2^n-dimensional state to a n-dimensional state after each VQC. This reduction may result in missing out on learning features in the exponentially large Hilbert space of the sequence. Additionally, executing and measuring up to six VQCs in parallel demands significant resources on actual quantum hardware. Simulating models with multiple VQCs using classical resources is also computationally expensive, which is why we limit the number of qubits n throughout the benchmark. These limitations have to be considered when interpreting the benchmark results of both architectures, QLSTM and LE-QLSTM, in comparison to the classical LSTM in Chapter 4.

3.5.4 Quantum Reservoir Computing

In Chapter 2, we introduced QRC and discussed its practical advantages over quantum machine learning models, such as those presented in previous sections. QRC for time series prediction has been studied in [70, 86, 87]. In this thesis, we propose an architecture with a more generalized measurement approach compared to prior methods. This approach involves measuring a set of randomly sampled observables. The key idea is that by measuring a sufficient number of observables, one can obtain a comprehensive representation of the data mapped into the Hilbert space. We previously discussed this concept in [88], where we demonstrated time series prediction and data interpolation for a Heisenberg spin chain system and a quantum circuit. In this thesis, we further explore this architecture and compare it to

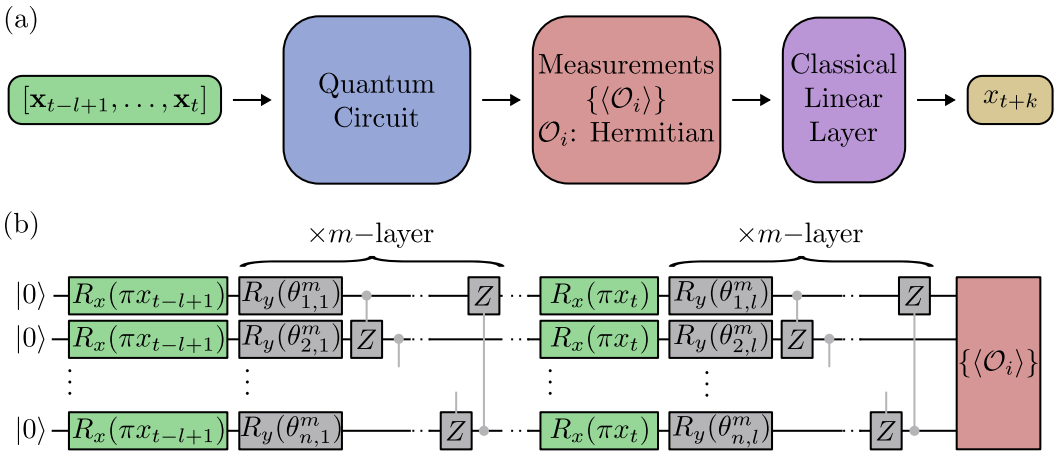

FIGURE 3.12: (a): Schematic of the QRC strategy employed in this thesis. A data sequence is encoded and processed through a quantum circuit. Expectation values of randomly sampled hermitian matrices are measured and passed into a classical linear layer, which is trained to approximate the target time series. (b): Quantum circuit architecture used in this thesis. The sequences are encoded in a blockwise recurrent manner, with fixed rotation gates and nearest-neighbor entangling applied within each block to create a versatile state representation of the time series.

other machine learning models, both classical and quantum, in the comprehensive benchmark study.

The QRC strategy used in this thesis is illustrated in Figure 3.12 (a). A time sequence $[\mathbf{x}_{t-l+1}, \ldots, \mathbf{x}_t]$ is sequentially encoded onto a quantum circuit. After executing the circuit, the resulting quantum state is measured by evaluating the expectation values of M random hermitian observables $\langle \mathcal{O}_i \rangle$. These matrices must satisfy the hermitian condition $\mathcal{O}_i = \mathcal{O}_i^\dagger$ to ensure that the eigenvalues, and thus the measurement results, are real. If this condition is not met, nonphysical results could arise. Given this constraint, we sample the real and imaginary parts of the hermitian matrix entries from a normal distribution with a mean of 0 and a variance of 1. The measurement results are then fed into a classical linear output layer, which is trained to approximate the target value $\mathbf{x}_{t+k}$.

In Figure 3.12 (b), the quantum circuit architecture employed in this thesis is depicted. Each sequence $[\mathbf{x}_{t-l+1}, \ldots, \mathbf{x}_t]$ is encoded sequentially in a block-wise structure. Within each block, the data point from the sequence is first encoded. For one-dimensional data, a data point x_i is encoded using an R_x-rotation on all qubits. As with other quantum machine learning models, all data is scaled to the interval $[0, 1]$. Therefore we scale the values to be encoded with a factor π resulting in

$$R_x(\pi x_i)\,. \tag{3.18}$$

For three-dimensional data, a data point $\mathbf{x}_i = (x_i^1, x_i^2, x_i^3)$ is encoded by

$$R_z(\pi x_i^3) R_y(\pi x_i^2) R_x(\pi x_i^1)\,. \tag{3.19}$$

Next, within each block, m layers of single-qubit R_y rotations with parameters $\theta^m_{i,j}$ and nearest-neighbor entangling using controlled-Z gates are applied. It is important to note that the parameters $\theta^m_{i,j}$ are randomly chosen, fixed and not optimized

at any point. Consistent with the principles of reservoir computing, only the classical linear output layer is trained based on the measurements from the quantum reservoir. The parameterized R_y rotation gates are used solely to achieve a versatile quantum state representation of the data.

The model's hyperparameters include the number of layers m within the quantum circuit architecture, the number of qubits n, and the number of measurements M conducted and passed into the classical linear layer. Exploring the effect of these hyperparameters on the prediction accuracy of the model will be subject to this thesis.

3.6 Technicalities of Benchmarking

Benchmarking is a crucial tool for systematically comparing systems to ensure fair evaluations across different approaches. In QML, it is particularly relevant to assess the performance of quantum models relative to each other and to classical methods. By conducting benchmarks, we can identify potential advantages of quantum techniques over classical ones, or the absence thereof. In this thesis, we examine the time series prediction accuracy of the models introduced in previous sections. While factors such as time complexity or energy consumption are also significant, this thesis focuses on benchmarking the learning capabilities of the models.

A good benchmark should be fair, comparable, and reproducible [89]. Fairness means that all models must have the ability to complete the task without restrictions, such as limited hardware resources. To ensure this, we select time series prediction tasks that allow quantum models, simulated on classical hardware, to converge. Comparability is achieved by using identical prediction tasks, training procedures, and evaluation metrics across models, which will be detailed in subsequent sections. Reproducibility is guaranteed by enabling consistent random initializations using predefined random seeds. In the following sections, we first discuss the models' hyperparameters and time series data. Then, we explain the data preprocessing and training procedures. Lastly, we present the metrics used to assess prediction accuracy.

3.6.1 Hyperparameters and Setup

Throughout the benchmark, we explore a range of time series prediction tasks with varying degrees of complexity. To control this complexity, we adjust both the sequence length l fed into the models and the number of prediction steps k, which represent how many steps the target value is ahead of the input sequence. Longer sequences provide more information for the model to learn from, so in principle increasing l reduces the complexity of the problem. However, processing longer sequences requires models that can handle the additional information they contain. Conversely, increasing k makes the prediction task more nonlinear and therefore more difficult to learn.

As shown in Table 3.2, we use sequence lengths of $l \in \{4, 8, 16\}$ and prediction steps of $k \in \{1, 10, 100\}$ throughout the benchmark. We set the lower bound of the sequence length to $l = 4$, as this provides enough information for the models to learn effectively. The upper bound, $l = 16$, ensures the feasibility of simulating quantum models within the available computational resources. For the prediction steps, $k = 1$ represents a highly linear scenario where the task is to predict the next value based on the last l values. At $k = 10$, the models face a task where the predicted value is

Hyperparameter	Values
Sequence Length l	$\{4, 8, 16\}$
Prediction Step k	$\{1, 10, 100\}$

TABLE 3.2: Hyperparameters of the time series prediction problems employed in this thesis.

approximately one Lyapunov time τ ahead, given the datasets used in the benchmark. As noted earlier in this chapter, this is $\tau = 10$ for the Mackey-Glass dataset and $\tau = 20$ for the Lorenz dataset. Finally, at $k = 100$, the models must predict a value several Lyapunov times into the future, which is a much more challenging problem.

We now discuss the model hyperparameters used in the benchmark. All hyperparameters are listed in Table 3.3. For the MLP, we vary both the activation functions and the hidden layer structures. The activation functions considered are $\{\tanh, \sigma\}$, while the hidden layer structures include three configurations with sizes $\{4\_16\_4, 6\_64\_6, 8\_256\_8\}$. These configurations correspond to the dimensionality of the data representation in a VQC using 4, 6 or 8 qubits.

For the RNN and LSTM models, we explore different hidden layer sizes {2, 4, 8, 16, 32} and different numbers of layers in the internal neural networks $\{1, 2, 3\}$. These values allow a comprehensive exploration of model configurations. As shown in the results, these parameter ranges contain the optimal settings for the time series prediction tasks.

For the ELM model, we choose hidden state dimensions corresponding to the dimension of the Hilbert space of the QRC architecture. Therefore, we choose this hyperparameter from the set {16, 64, 256, 1024}, which corresponds to quantum systems with {4, 6, 8, 10} qubits. Although we only train the quantum models with up to 8 qubits, we include a hidden state dimension of 1024 to get a wider range of this hyperparameter. The hyperparameter of the number of measurements is optimised for values from the set $\{1, 10, 100, 1000\}$.

We now discuss the hyperparameters of the quantum machine learning models. For the VQC, different architectures are included in the benchmark, as described in the previous section. In all cases, the number of qubits n, serves as the model hyperparameter. Note that for the VQC model without input layer, the sequence length must be identical to n. For these architectures we choose $n \in \{4, 8\}$. For all other models we set $n \in \{4, 6, 8\}$. This ensures that the models remain small enough to allow the training to be simulated with the classical computing resources available. All VQC architectures follow a layered structure in the ansatz. For the architecture without input layer, we investigate the dependence of the number of ansatz layers on the time series prediction capability. Therefore, we chose the number of layers for this architecture from the set $\{1, 2, 3, 4\}$. For all other architectures the number of layers is taken from $\{1, 2\}$.

In the QRNN model, the number of qubits in the hidden register is a hyperparameter, $h \in \{2, 4\}$. For the data register, we use $d = 2$ qubits. The limited number of qubits is due to the exponential growth of the computational resources required to simulate and train the QRNN using the Python library PennyLane [85], especially when resetting qubits to zero. Therefore, we only train QRNN models with a sequence length of $l = 4$.

For the different QLSTM architectures, we use $n \in \{4, 6\}$ qubits and $\{1, 2\}$ layers in the VQC ansatz. For the LE-QLSTM, hidden sizes of $\{4, 8, 16\}$ are explored to ensure compatibility with the classical LSTM architecture.

Model	Hyperparameter	Values
MLP	Activation function	$\{\tanh, \sigma\}$
	Hidden Layer Structure	$\{4\_16\_4, 6\_64\_6, 8\_256\_8\}$
RNN	Hidden Sizes	$\{2, 4, 8, 16, 32\}$
	Number of Layers	$\{1, 2, 3\}$
LSTM	Hidden Sizes	$\{2, 4, 8, 16, 32\}$
	Number of Layers	$\{1, 2, 3\}$
ELM	Hidden State Dimensions	$\{16, 64, 256, 1024\}$
	Number of Measurements	$\{1, 10, 100, 1000\}$
VQC	Number of qubits (no input layer)	$\{4, 8\}$
	Number of qubits (other architectures)	$\{4, 6, 8\}$
	Ansatz Layers (no input layer)	$\{1, 2, 3, 4\}$
	Ansatz Layers (other architectures)	$\{1, 2\}$
QRNN	Number of hidden qubits	$\{2, 4\}$
QLSTM	Number of qubits	$\{4, 6\}$
	Ansatz Layers	$\{1, 2\}$
	Hidden Sizes (LE-QLSTM)	$\{4, 8, 16\}$
QRC	Number of qubits	$\{4, 6, 8\}$
	Ansatz Layers	$\{1, 2\}$
	Number of Measurements	$\{1, 2, 4, 6, 8, 12, 18, 26,$ $37, 54, 78, 112, 162,$ $233, 335, 483, 695, 1000\}$

TABLE 3.3: The hyperparameters for the different classical and quantum models that are part of the benchmark study conducted in this thesis.

Finally, in the QRC architecture, the hyperparameters are the number of qubits $n \in \{4, 6, 8\}$ and the number of layers in the ansatz $\{1, 2\}$. Additionally, the number of measurements M varies from 1 to 1000, with 20 integer values logarithmically distributed over this range. As will be discussed in the results in Chaper 4, this range is sufficient to observe convergence to an optimal value for M for the problems studied.

To reduce the influence of statistical variance, each set of hyperparameters is trained for ten different and randomly sampled initializations. From these ten models, the median is determined with respect to a prediction accuracy metric discussed at the end of this section. We use the median rather than the mean as it is less sensitive to outliers within the results of different model initializations. For a given set of hyperparameters, these values are examined throughout the benchmark. Unless we specifically want to examine the dependence of a specific model hyperparameter, we perform a hyperparameter optimization over the different parameters shown in Table 3.3. In this work we mean by hyperparameter optimization, that we determine the best median prediction accuracy obtained by a distinct selection of model hyperparameters with respect to a given metric for each architecture and each combination of dataset, number of prediction steps and sequence length. We use this value for further evaluation. As an measure for the error within the different initializations we use the median absolute deviation (MAD)

$$\text{MAD} = \text{median}\left(|x_i - \text{median}(\{x_i\})|\right) . \tag{3.20}$$

In the following, we discuss the data preprocessing and the model training.

3.6.2 Data Preprocessing and Training

To ensure comparability across models, we scale both datasets X, the Lorenz Attractor dataset and the Mackey-Glass time series dataset, to the interval $[0, 1]$ using min-max scaling

$$X = \frac{X - x_{min}}{x_{max} - x_{min}}\,. \tag{3.21}$$

Here, x_{min} and x_{max} represent the minimum and maximum values of X, respectively. For the Lorenz Attractor dataset, each dimension is scaled individually to $[0, 1]$.

After rescaling, we construct a dataset $\hat{X}$ consisting of tuples of sequences of consecutive data points and their corresponding labels, as described in Equation (3.4). This dataset is then divided into a training set, a validation set, and a test set. The training set contains the first 60% of the data points in $\hat{X}$ and is used exclusively for model training. We use the Mean-Square Error (MSE) loss function and the Adam optimizer [28] for training. The learning rate is an hyperparameter of the training. We train for learning rates in $\{0.001, 0.0001, 0.00001\}$, where the results of the learning rate with the best prediction accuracy with respect to a metric discussed in the next section is taken into account for further analysis.

The validation set, which contains the 20% after the training dataset of the data points in $\hat{X}$, are used to determine when the model has converged. Convergence is judged using the following criteria, which is also used in [71]. At each epoch, we look at the last 400 loss values over the validation set. We calculate the mean of the first 200 values μ_1, the mean of the second 200 values μ_2, and the standard deviation of the second 200 values σ_2. The training stops when the condition

$$|\mu_1 - \mu_2| < \frac{\sigma_2}{2\sqrt{200}} \tag{3.22}$$

is satisfied. During training, the loss decreases, so the difference between μ_1 and μ_2 should be greater than half the standard deviation of μ_2. Once the model converges, the difference between μ_1 and μ_2 should be less than the standard deviation of μ_2 over 200 epochs. Observing the loss over the epochs for all models, we found this criteria as suitable to determine the convergence of the models.Finally, the test dataset, which contains the remaining 20% of the data points in $\hat{X}$, is used to evaluate the trained model. The test dataset is not used during training. Prediction accuracy is evaluated on this dataset, and the metrics for evaluation are discussed in the following section.

3.6.3 Measures for Prediction Accuracy

In this benchmark, we evaluate the predictive capabilities of quantum machine learning models. To compare the prediction accuracy of different models, an appropriate metric is required. Several metrics are available, each with specific advantages and disadvantages. In the following, we discuss the metrics used in this work.

One class of metrics is based on the error between predicted values $\tilde{\mathbf{x}}_i$ and true labels $\mathbf{x}_i$ from the N values in the test dataset. A common metric is the Mean Squared Error (MSE)

$$E(\text{MSE}) = \frac{1}{N}\sum_{i=1}^{N}(\tilde{\mathbf{x}}_i - \mathbf{x}_i)^2\,. \tag{3.23}$$

MSE is also used as a loss function during training. Another error-based metric is the Mean Absolute Error (MAE)

$$E(\text{MAE}) = \frac{1}{N}\sum_{i=1}^{N} |\tilde{\mathbf{x}}_i - \mathbf{x}_i| \,. \tag{3.24}$$

As discussed in Chapter 2, MSE is more sensitive to outliers due to the quadratic contribution of large errors. In contrast, MAE treats all prediction errors uniformly. In this benchmark, we consider both metrics.

An alternative metric for prediction accuracy is the correlation between predicted values $\tilde{\mathbf{x}}_i$ and true labels $\mathbf{x}_i$. We employ the Pearson Correlation Coefficient (PCC) [90], which quantifies the linear correlation between the two datasets $\{\tilde{\mathbf{x}}_i\}$ and $\{\mathbf{x}_i\}$

$$E(\text{PCC}) = \frac{\sum_i^N (\mathbf{x}_i - \bar{\mathbf{x}})(\tilde{\mathbf{x}}_i - \bar{\tilde{\mathbf{x}}})}{\sqrt{\sum_i^N (\mathbf{x}_i - \bar{\mathbf{x}})^2}\sqrt{\sum_i^N (\tilde{\mathbf{x}}_i - \bar{\tilde{\mathbf{x}}})^2}} \,. \tag{3.25}$$

Here, $\bar{\mathbf{x}}$ and $\bar{\tilde{\mathbf{x}}}$ are the sample means

$$\bar{\mathbf{x}} = \frac{1}{M}\sum_{i}^{M} \mathbf{x}_i \,, \quad \bar{\tilde{\mathbf{x}}} = \frac{1}{M}\sum_{i}^{M} \tilde{\mathbf{x}}_i \tag{3.26}$$

of the datasets $\{\tilde{\mathbf{x}}_i\}$ and $\{\mathbf{x}_i\}$. In the case of perfect prediction accuracy, $\{\tilde{\mathbf{x}}_i\} = \{\mathbf{x}_i\}$, the correlation coefficient is equal to one. If no linear correlation exists, the coefficient is zero. The benefit of using the Pearson Correlation Coefficient is that it is independent of the metrics used during training. We consider this metric throughout the benchmark alongside error-based measures. In the following chapter, we present the results of this work. Throughout this chapter, we primarily consider the median MSE on the test dataset of all ten different initializations of each model trained for a given set of hyperparameters. We do so because the MSE more heavily weights outliers in the prediction of data points in the test dataset. While we primarily use this metric, we show that the results obtained in this benchmark are robust to the use of the other metrics discussed above.

Chapter 4

Results and Discussion

After establishing the theoretical foundation in Chapter 2 and discussing the methods used in this thesis in Chapter 3, we now present the results of our work.

As mentioned in the previous chapter, the analysis performed in this thesis includes the investigation of various hyperparameters in the data as well as the optimization of hyperparameters of the models. Presenting the results over all hyperparameters and all different datasets, number of prediction steps and sequence lengths is not feasible for a clear presentation of the results. Therefore, we follow the following principle throughout this chapter: We show the most relevant or representative results in this chapter, while all other results can be found in the appendix A.

We begin by evaluating various quantum machine learning models alongside their classical counterparts. In this analysis, we examine different aspects of quantum machine learning architectures, such as the impact of tuning the number of qubits in quantum circuits and changing the number of layers in a VQC. Furthermore, we investigate how the presence of an entangled quantum state affects the prediction accuracy of the quantum models.

Next, we explore the proposed quantum reservoir computing architecture and analyze the influence of different hyperparameters, including the number of reservoir measurements, on the prediction accuracy. We also evaluate how the construction of observables affects model performances and compare this approach to a classical extreme learning machine. Finally, we present the results of the comprehensive benchmark study that compares all models analyzed.

4.1 Evaluation of Quantum Machine Learning Models

4.1.1 Variational Quantum Circuit

We begin our analysis of QML models for time series prediction by exploring the VQC architectures discussed in Chapter 3. The three architectures shown in Figures 3.6, 3.7, and 3.8 will be referred to as VQC without a classical input layer, VQC with a classical input layer, and VQC with a data re-uploading structure.

We first consider the architecture without a classical input layer. In this design, all time steps of a sequence of length l are encoded at the beginning of the quantum circuit using angle encoding. The number of qubits n is required to match the sequence length l. After encoding, a layered approach is applied. In this section, we investigate how the number of layers affects the prediction performance. In addition, we evaluate the role of controlled two-qubit gates in the ansatz shown in Figure 3.6 (d), which allow the model to access entangled quantum states. To explore this, we train models with the same architecture and hyperparameters but without two-qubit gates, thereby restricting the quantum state to the tensor product of individual qubit states and limiting access to the full Hilbert space.

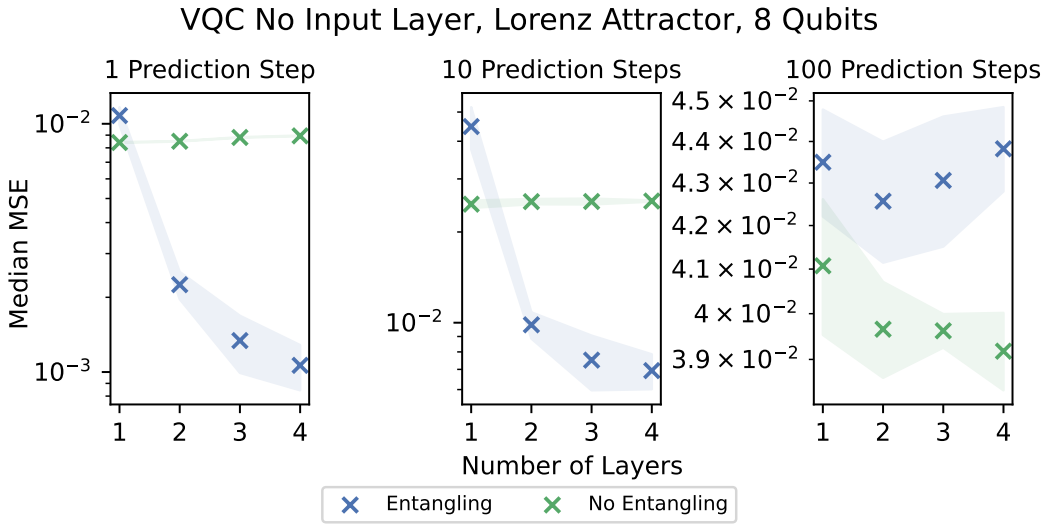

FIGURE 4.1: Median MSE of the VQC architecture without input layer with eight qubits trained on the Lorenz Attractor dataset. The subplots show the prediction accuracy for one, ten, and 100 steps ahead for different numbers of input layers. The blue markers represent results from models with controlled two-qubit gates, while the green markers show results from models without two-qubit gates. The shaded area shows the Median Absolute Deviation (MAD) over ten runs with different random initializations. Increasing the number of layers generally improves short-term predictions, especially in the presence of entangled gates, while performance saturates for long-term predictions. Models without entangled gates show less sensitivity to the number of layers.

Figure 4.1 shows results from training on the Lorenz Attractor dataset with quantum circuits using eight qubits. These results are representative of those obtained with different numbers of qubits or with data from the Mackey-Glass dataset. Additional results can be found in the appendix in Figures A.1 and A.2. The subplots show the performance of the model for predicting values one, ten, and 100 steps ahead of the encoded sequence. We report the median Mean Squared Error (MSE) from training ten models with identical model hyperparameters but different random initializations of the trainable parameters. The markers represent the median MSE, while the shaded area indicates the median absolute deviation (MAD) of the ten MSE values. This visualization is consistent across all plots in this section.

We first analyze models using the entangled ansatz, where controlled two-qubit gates are present, allowing entangled quantum states. These results are shown with blue markers. For predictions one and ten steps ahead, the accuracy improves with increasing layers within the trained domain. As discussed in the previous chapter, predicting only a few steps ahead is a relatively linear task. Our results suggest that for such problems, increasing the number of trainable quantum parameters by adding layers improves prediction performance. More trainable parameters are likely to provide greater flexibility, while nearest-neighbor entanglement gates may expand the accessible Hilbert space, as suggested by [53]. Their work, using a layered VQC ansatz similar to ours, shows that the parameter capacity increases with the number of layers until it stagnates at a characteristic value. This implies that beyond a certain number of layers, prediction accuracy may not improve. In [91], the

number of layers and parameters in a VQC is related to the phenomenon of overparameterization, where trainable parameters exceed the number needed to explore the independent directions of the Hilbert space. Their results are consistent with our findings, suggesting that prediction accuracy improves with more layers until a saturation point is reached. For predictions 100 steps ahead, there is no significant difference in error. This task, which is highly nonlinear, may already exceed the capacity of the model, potentially being in an overparameterized regime. Alternatively, additional layers may improve accuracy. However, further simulations are limited by hardware constraints.

We then compare these results with those from non-entangled models that omit the controlled two-qubit gates, shown in Figure 4.1 with green markers. These models provide insight into whether learning in higher-dimensional spaces benefits VQC architectures. We find that the number of layers has a minimal effect on accuracy. For one- and ten-step predictions with a single layer, this approach outperforms the entangled architecture, while for more layers the original approach performs better. This trend holds across datasets and number of qubits. For 100-step predictions, the differences between the two approaches are small, with each performing slightly better for different tasks. The lack of layer dependency in the no-entanglement approach can be understood by considering the circuit ansatz. Once a data point is encoded on a qubit, subsequent layers merely adjust single-qubit rotation gates, which become redundant. Thus, entangling gates help to achieve more independent trainable parameters. However, this does not necessarily mean that entangling gates allow the model to fully explore the high-dimensional Hilbert space. To our knowledge, this is the first investigation of the impact of entangling gates in the VQC approach to time series prediction. Previous studies on classification tasks [71, 92] found minimal correlation between entanglement and classification accuracy. However, these studies typically used architectures with few layers, suggesting that increasing the number of layers may benefit models with entangled gates. Alternatively, the improved performance observed here may be due to architectural artifacts, such as the entangled ansatz being better suited for the specific data considered here, rather than effective use of the Hilbert space.

We continue by investigating the VQC architecture with a classical input layer. This input layer allows the number of qubits to be chosen independently of the sequence length, since the classical layer maps the sequential data to a dimension corresponding to the number of qubits. Taking advantage of this flexibility, we investigate how increasing the number of qubits affects the prediction accuracy. As the number of qubits increases, the VQC has access to a Hilbert space whose dimension scales exponentially with the number of qubits. So in theory, this provides the model with a larger space in which it can learn features in the data.

Figure 4.2 shows the median MSE for training data with a sequence length of eight and ten prediction steps as a function of the number of qubits. We compare a one-layer ansatz with a two-layer ansatz. The left panel presents results for the Mackey-Glass dataset. We observe that increasing the number of qubits does not significantly affect the prediction accuracy within the error margin, regardless of the number of layers. The reduction is small for most combinations of dataset, sequence lengths and prediction steps, as shown in the appendix in Figures A.3 and A.4. In contrast, the right panel, showing results for the Lorenz-Attractor dataset, reveals a slight decrease in median MSE for both ansätze as the number of qubits increases. Here, a system of four qubits may be too small to capture relevant features in the data.

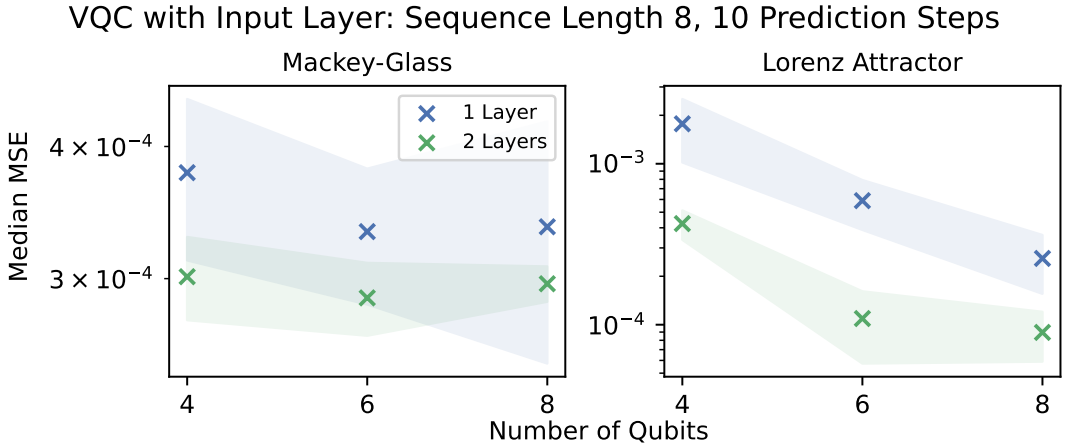

FIGURE 4.2: We show the median MSE on the test dataset over the number of qubits in a VQC with one input layer. A single-layer VQC architecture is compared to a two-layer architecture. This figure shows the data for a sequence length of eight and ten prediction steps. On the left, we show the results for the Mackey-Glass dataset. It can be seen that increasing the number of qubits has no effect on the prediction accuracy for either number of layers within the error range of the results. For the Lorenz-Attractor dataset, shown on the right, we observe a small decrease in the median MSE when increasing the number of qubits, but the amount of decrease is still less than an order of magnitude.

We propose two hypotheses to explain the overall insignificant impact of the number of qubits on the results. First, the number of qubits used may already be sufficient to achieve optimal prediction accuracy, with the system size being large enough to capture the relevant data features. This assumption is supported by the fact that doubling the number of qubits from four to eight does not significantly improve prediction accuracy in any of the tasks studied. Also, when taking classical models into account later, it becomes apparent that those yield prediction accuracy comparable to the VQC architecture discussed here. Alternatively but less likely, the number of qubits in the VQC may be insufficient to reveal significant learning in the exponentially larger Hilbert space. However, testing this hypothesis with numerical simulations is not feasible because simulating circuits with a large number of qubits requires exponentially more classical computational resources.

So far, we analyzed the effect of the number of ansatz layers structure as well as the effect of two-qubit gates on the prediction accuracy in a VQC without a classical input layer. Additionally, we studied how increasing the number of qubits in VQC with a classical input layer affects the results. Now, we finally compare these architectures with the data re-uploading architecture.

Figure 4.3 shows the results of these three quantum architectures alongside those of a classical MLP. For each architecture and learning problem, we plot the median MSE from ten random model initializations over varying sequence lengths. For the VQC models, these results reflect the best median MSE obtained after optimizing the number of qubits and layers. For the MLP model, results are obtained by optimizing hyperparameters such as the activation function and hidden layer configuration. The corresponding hyperparameter sets are listed in Table 3.3. The left subplot in the figure shows results from training on the Lorenz Attractor dataset with ten-level

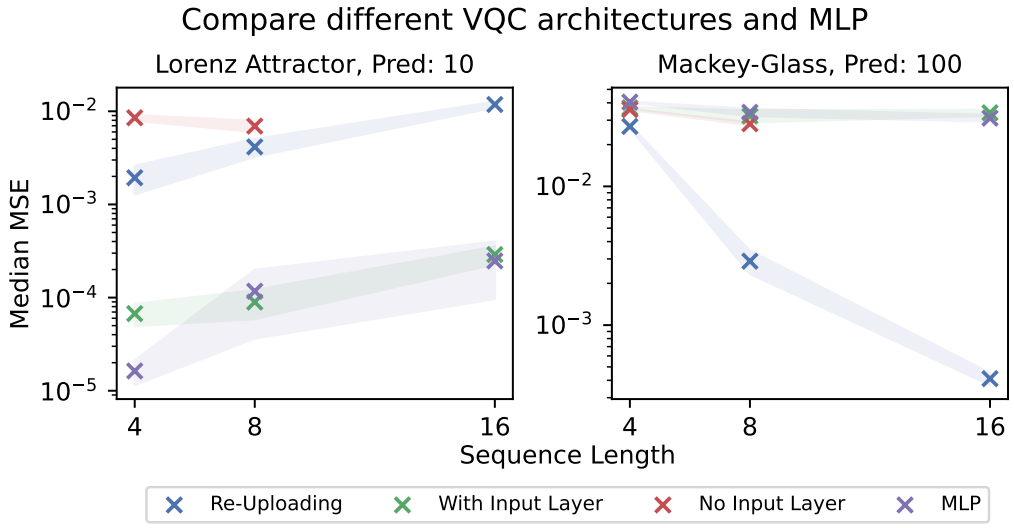

FIGURE 4.3: Here we compare the time series prediction performance of the different VQC architectures with a classical MLP. The left plot shows the median MSE over the sequence length for training the models on the Lorenz Attractor dataset with ten prediction steps. The results shown here are representative for most of the learning problems considered. While the VQC with an input layer and the MLP archive have the highest prediction accuracy, the VQC architectures without an input layer and the data re-uploading architecture are at best equal to the prediction accuracy. The right plot shows the results for training the Mackay-Glass dataset for 100 prediction steps. In contrast to all other learning problems, the VQC with data re-uploading structure outperforms the other models that are trained.

predictions, which is representative of most dataset and prediction-level combinations explored in this study. The right subplot shows results for training on the Mackey-Glass dataset with 100-step predictions. The special behavior observed for this particular task is discussed at the end of this section. Comprehensive results for all dataset-prediction step combinations are provided in the appendix in Figure A.5.

The first major observation across all learning problems is that the performance of the VQC with a classical input layer is quite similar to that of the MLP, as indicated in Figure 4.3, by the green and purple markers. This similarity holds across all datasets, number of prediction steps and sequence lengths. These results suggest that a VQC sandwiched between a classical linear input layer and a classical linear output layers may not offer a significant advantage over the MLP, raising questions about the added value of using a VQC in this configuration. In such cases, the quantum circuit may serve as little as a nonessential quantum layer.

For the VQC without an input layer, we report results for sequence lengths of four and eight, as the sequence length is tied to the number of qubits in this model. Simulating a circuit with 16 qubits for a sequence length of 16 is beyond our available classical computational resources. We find that this model generally underperforms compared to other VQC architectures across all tasks. For short prediction steps on the Lorenz Attractor dataset, the VQC without an input layer shows a prediction error that is about two orders of magnitude worse than the MLP.

A similar trend is observed in the VQC architecture employing a data re-uploading structure for most of the problems studied. In this architecture, each sequence step

is encoded sequentially across consecutive layers. Surprisingly, the prediction accuracy remains low, contrary to [34], which suggests that the data re-uploading structure should enhance the VQC's expressivity. One exception is the prediction of 100 steps ahead on the Mackey-Glass dataset. As indicated by the blue markers in the right subplot of Figure 4.3, the prediction accuracy in this case is significantly higher than that of the other VQC architectures and the MLP. For sequence lengths of 16, this model outperforms the others by more than an order of magnitude. This suggests that the model may be exploiting a frequency spectrum well-suited to this specific learning task, highlighting the potential for the data re-uploading architecture to improve prediction accuracy in certain time series prediction problems.

However, this does not mean that the success of VQC on this dataset and for this prediction length is necessarily due to quantum mechanical advantages, or that it outperforms all classical MLPs. We trained only a limited set of MLP structures, and it is possible that other MLP configurations might perform better, especially given the highly nonlinear nature of the learning task. The same is true for the VQC models, different ansatz designs could further improve prediction accuracy. However, we are constrained to analyze only a reasonable subset of possible architectures. Another limitation we must consider is that each model architecture underwent different degrees of hyperparameter optimization. The VQC without an input layer is only optimized for the number of layers, while the other two VQC architectures are also optimized for the number of qubits. This disparity in the number of optimizations across architectures is a fundamental challenge in model benchmarking and must be taken into account when interpreting the results, as well as in the final evaluation of all models at the end of this chapter.

4.1.2 Quantum Recurrent Neural Network

We now analyze the time series prediction capabilities of the Quantum Recurrent Neural Network [15], as introduced in Chapter 3. The core concept is to use a quantum circuit where one register of qubits sequentially encodes data into a quantum state, while another register of qubits provides a hidden quantum state passed along the sequence, similar to a classical RNN. In the original architecture, the data register qubits are reset to the state $|0\rangle$ after each step, as shown in Figure 3.9. This section compares the reset approach with an alternative method where the qubits in the data register are not reset. In the latter case, the data of the next time step is encoded over of the quantum state of the previous time step using angle encoding. Notably, the ansatz used in both approaches remains identical. In addition, we compare the prediction accuracy of the QRNN architectures with that of a classical RNN.

First, we want to remind that resetting qubits and subsequently reusing them in further computations requires classical resources that scale exponentially with each reused qubit when simulated using the Python library PennyLane [85], which is used in this benchmark study. Therefore, we limit the sequence length to four in the learning problems studied for the QRNN architecture. In addition, the number of qubits in the data register must remain small.

In Figure 4.4, we present the median MSE plotted over different prediction steps for the QRNN architecture, both with and without resetting the data qubits, as well as for the classical RNN. The results for the QRNN architectures are obtained by training models with two qubits in both the data and hidden registers. The left plot shows the results for learning the Mackey-Glass dataset, while the right plot shows the results for learning the Lorenz Attractor dataset. Comparing the QRNN with and without qubit resets in the data register shows that the non-reset approach

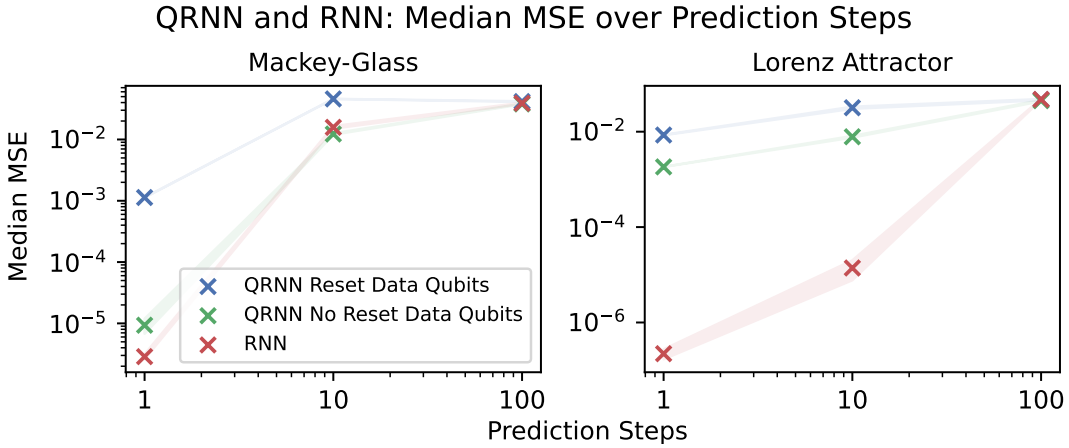

FIGURE 4.4: Here, the median MSE over the prediction steps is plotted for the QRNN architectures, both with and without resetting the qubits in the data register, as well as for the classical RNN model. For small prediction steps, not resetting the qubits in the data register leads to better prediction accuracy compared to resetting them. At best, the QRNN models only match the performance of their classical counterpart. In particular, for small prediction steps on the Lorenz Attractor dataset, the classical RNN achieves a prediction accuracy about three orders of magnitude better than that of the QRNN architectures.

achieves better prediction accuracy for small prediction steps in both datasets. In particular, for the Mackey-Glass dataset with a prediction step of one, the non-reset approach outperforms the reset approach by about two orders of magnitude. Only for larger prediction steps does the difference in prediction accuracy between the two approaches decrease.

These results question the necessity of resetting the qubits in the data register to the state $|0\rangle$. The loss of information during the reset process may explain this result. The QRNN architecture aims to transfer the information encoded in the data register to the hidden register. However, without reset, the model not only transfers information from the data register to the hidden register, but also retains information in the data register itself. This raises the question of whether there is any advantage to assigning qubits to different registers, which limits the number of qubits available for encoding data. A more expressive model might emerge without this restriction [34]. However, such an architecture would be similar to the VQC data re-uploading architecture discussed in the previous section.

Figure 4.4 also shows the prediction errors of the classical RNN. These results are obtained by hyperparameter optimization with respect to the size of the hidden part and the number of neural network layers. For the Mackey-Glass dataset, the QRNN without data qubit resets shows similar prediction accuracy across all prediction steps. However, for the Lorenz Attractor dataset, the RNN outperforms the quantum models by more than three orders of magnitude for small prediction steps. For larger prediction steps, corresponding to more highly nonlinear learning problems, the QRNN architectures achieve comparable prediction accuracy.

The results for a larger hidden register with four qubits are similar to those in Figure 4.4 and are shown in Figure A.6 in the appendix.

In summary, resetting the data qubits in the QRNN architecture can degrade time series prediction performance. For the problems studied, QRNN models at

best match the performance of their classical counterpart. These results call into question the usefulness of the originally proposed QRNN architecture for time series prediction.

4.1.3 Quantum Long Short-Term Memory

In this section, we analyze the time series prediction capabilities of various quantum and classical LSTM architectures introduced in Chapter 3. The core idea of the LSTM architecture is to pass time sequences through stacked LSTM cells, as shown in Figure 3.5. These cells are designed to propagate two hidden states, representing long-term and short-term memory, along the sequence. While the classical LSTM uses neural networks, its quantum counterpart replaces these networks with VQCs, as shown in Figure 3.10. In the model presented in [15], the hidden state dimension is tied to the number of qubits in the VQCs. To increase flexibility and to allow larger hidden state dimensions even when the VQCs have few qubits, the linear-enhanced QLSTM (LE-QLSTM) architecture has been proposed [17]. This approach introduces additional classical linear layers that map the inputs and outputs of the VQCs to the desired dimensions, as shown in Figure 3.11. Throughout this section, we abbreviate the former model as QLSTM and the latter as LE-QLSTM.

In Figure 4.5, we present the median MSE of different model initializations plotted against sequence length for both one-step and 100-step predictions on the Mackey-Glass dataset. For the QLSTM architecture, the results are obtained by hyperparameter optimization of the number of qubits and layers in the VQC. We report the best median MSE for each learning task based on this optimization. For the LE-QLSTM architecture, the hyperparameter optimization includes the number of qubits, layers, and hidden state size. For the LSTM model, the hyperparameters of the number of neural network layers and the hidden state size are tuned.

On the left side of the Figure 4.5 we show the results for training the models to predict one step ahead. The classical LSTM achieves the lowest prediction error over all sequence lengths, followed by the LE-QLSTM. In contrast, the QLSTM shows the highest prediction error for these tasks. While the LSTM and LE-QLSTM improve with longer sequence lengths, the prediction accuracy of the QLSTM decreases. For the 100-step prediction task shown on the right, we observe no significant difference in prediction accuracy across models or sequence lengths. Furthermore, there is no significant effect of sequence length on the median MSE. Similar trends are observed for the Lorenz Attractor dataset, where the differences in prediction accuracy between architectures decrease as the prediction horizon increases. The results for training on ten-step predictions follow this pattern. All results are presented in the appendix, Figure A.7.

These results suggest that for problems with linear data, as in the case of short prediction horizons, adding linear layers to the QLSTM architecture does indeed improve prediction accuracy, which is consistent with the results in [17]. However, the classical LSTM still achieves higher accuracy for these tasks. The LE-QLSTM architecture differs from the classical LSTM only by incorporating VQCs between the neural network layers. In this study, we chose comparable system sizes for both architectures. The fact that the LSTM outperforms the LE-QLSTM for short prediction horizons indicates that the addition of quantum neural networks to the LSTM cells does not improve the time series prediction capabilities. Furthermore, no significant differences between the models is found for highly nonlinear learning tasks. This may be due to limited expressiveness, which could be improved by increasing hidden state sizes or adding more layers in the VQC or neural networks. Larger

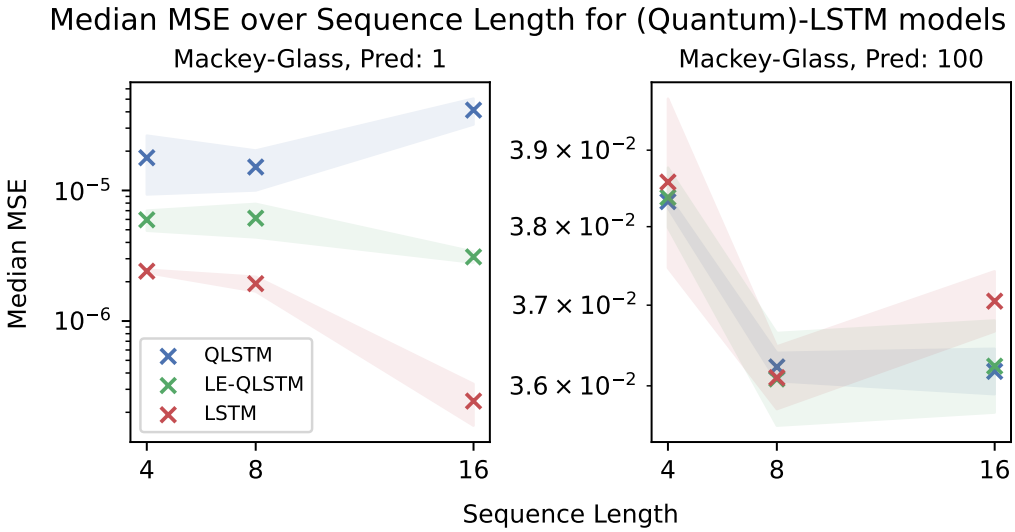

FIGURE 4.5: Median MSE of one-step and 100-step predictions for the classical LSTM, QLSTM, and LE-QLSTM models over different sequence lengths on the Mackey-Glass dataset. The left panel shows the results for one-step predictions, where the classical LSTM achieves the lowest MSE, followed by the LE-QLSTM, while the QLSTM has the highest error. The prediction accuracy of the LSTM and LE-QLSTM improves with increasing sequence length, while the performance of the QLSTM degrades. The right panel shows the results for 100-step predictions, where no significant differences in MSE are observed between the models or across sequence lengths. Overall, the findings suggest, that the classical LSTM outperforms the quantum models for short prediction horizons for time series prediction tasks.

architectures may allow the models to reach their full potential and reveal performance differences. However, this study has already explored a wide range of hyperparameters. It is likely that the limitations arise from the LSTM structure itself, resulting in similar performance across quantum and classical architectures. Finally, predicting time series far beyond Lyapunov time remains a complex challenge. In the comprehensive benchmark, we will discuss these results in the light of all other models examined.

4.2 Evaluation of the Quantum Reservoir Computing Architecture

After discussing the time series prediction capabilities of various variational QML models, we now focus on evaluating the QRC architecture, proposed in Chapter 3. We begin by analyzing the effect of the number of measurements made on the quantum reservoir on prediction accuracy. We then investigate how alternative reservoir measurements and the exclusion of two-qubit entangling gates in the circuit ansatz affect the model's ability to predict time series. Finally, we compare the architecture to a classical ELM.

Figure 4.6 presents a hyperparameter scan of the QRC architecture, exploring the effect of the number of measurements on the quantum reservoir for different sequence lengths and prediction steps over both datasets. The results are for the

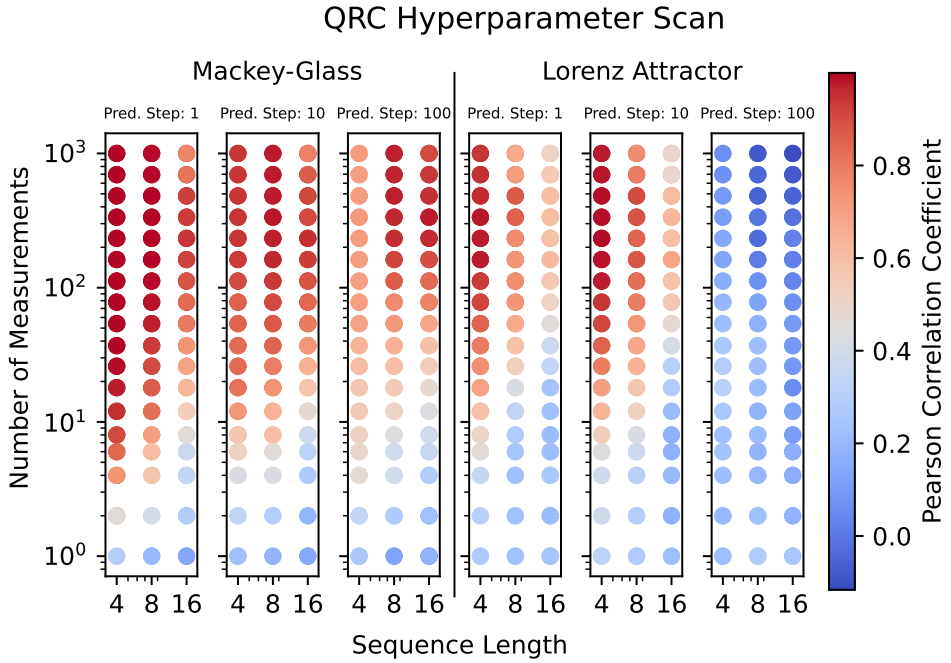

FIGURE 4.6: Color plot of a hyperparameter scan for the QRC architecture with a single-layer approach and for four qubits. The plot shows the effect of the number of measurements on prediction accuracy over different sequence lengths and prediction steps for both datasets. The colors represent the Pearson correlation coefficient between the predicted and true values on the test dataset, with red indicating high accuracy and blue indicating low accuracy. The results show that accuracy generally improves with more measurements. However, it plateaus or even drops for the Lorenz Attractor dataset with 100-step predictions.

single-layer QRC ansatz with four qubits. The colors in the plot represent the Pearson correlation coefficient (PCC) between the predicted values and the true values of the test dataset. A high PCC, indicating high prediction accuracy, is represented by red, while a low PCC is represented by blue. We choose PCC as a metric because it provides a clearer contrast in the color plot than MSE or MAE, even when using a logarithmic color scale.

It is evident that for most learning tasks, each defined by a unique combination of sequence length, number of prediction steps, and dataset, prediction accuracy improves as more randomly sampled hermitian observables are measured, up to a point of optimal accuracy. Two factors probably contribute to this effect. First, the size of the classical output layer increases, giving the model more flexibility as it gains access to additional parameters for optimization. Second, the model may acquire a more complete representation of the quantum state, allowing it to extract more relevant features from the data. However, for the Lorenz Attractor dataset with 100 prediction steps, this trend is not observed. In fact, as the number of measurements increases, the prediction error remains largely unchanged, and for longer sequence lengths, the error actually increases. This may be due to the highly nonlinear nature of the data, where the model struggles to capture relevant features, rendering the number of measurements ineffective. Predicting values ten times Lyapunov time ($\tau = 10$) ahead in such a chaotic time series is a particularly challenging task.

Next, we consider the prediction accuracy over different sequence lengths. For the Mackey-Glass dataset, it is evident that as the number of prediction steps increases, the best performance occurs at longer sequence lengths, provided a sufficient number of measurements are taken. For small prediction steps, capturing the linear relationship between the last encoded sequence step and the target value may be sufficient to achieve high accuracy. Encoding longer sequences in such cases may introduce unnecessary complexity. For longer prediction steps, however, the model may learn more complex correlations beyond successive time steps, such as the explicit memory within the Mackey-Glass data. In the Lorenz Attractor dataset, we observe improved accuracy for small sequences with short prediction steps, but no improvement for longer sequences with increased prediction steps, likely due to the model's difficulty in capturing the relevant data features, as discussed earlier.

We aim to further explore the quantum reservoir computing architecture by modifying certain model features. Figure 4.7 shows the median MSE as a function of the number of measurements made on the reservoir. These results are based on the Lorenz Attractor dataset, trained with a sequence length of eight and predicting ten steps ahead, as a representative case for different learning tasks. Additional results are shown in the appendix in Figures A.8 and A.9. All variations of the QRC architecture are subjected to hyperparameter optimisation for the number of qubits, with one ansatz layer used for all models. We chose this setup because for the standard architecture, no significant improvement in prediction accuracy is observed when increasing the number of layers, as confirmed in the appendix.

First, we modify the way measurements are constructed in the quantum reservoir. Previously, we randomly sampled m hermitian measurements by drawing the real and imaginary components from a normal distribution. We compare this approach to randomly sampling Pauli strings:

$$\sigma_1\sigma_2\cdots\sigma_n \equiv \sigma_1 \otimes \sigma_2 \otimes \cdots \otimes \sigma_n \tag{4.1}$$

with $\sigma_i \in I, X, Y, Z$. This observable is then measured on a quantum system of n qubits. The advantage of Pauli strings is that they are easier to implement on quantum hardware, which typically measures in the computational basis, which corresponds to measuring the Pauli-Z operator on all qubits. Performing a Pauli string measurement requires only single-qubit rotations, while more complex operations are required for general hermitian observables, increasing potential quantum computational errors. In Figure 4.7, blue markers represent the standard hermitian approach, while green markers correspond to Pauli string measurements. The shaded area again shows the MAD of the MSE over ten model initializations. The two measurement approaches show no significant difference in prediction accuracy as the number of measurements increases. Across all datasets and sequence lengths, Pauli string measurements perform either similarly or slightly better, suggesting that random Pauli strings, a subset of all possible hermitian observables, are sufficient to capture relevant quantum state features. This also makes quantum hardware implementation more feasible.

Next, we investigate the effect of two-qubit gates in the ansatz, which introduce entanglement. Our goal is to determine whether QRC benefits from mapping data into high-dimensional, entangled quantum states. We modified the standard hermitian architecture by removing all two-qubit gates, as shown in Figure 3.12. This modification results in no interaction between qubits. The red markers in the figure

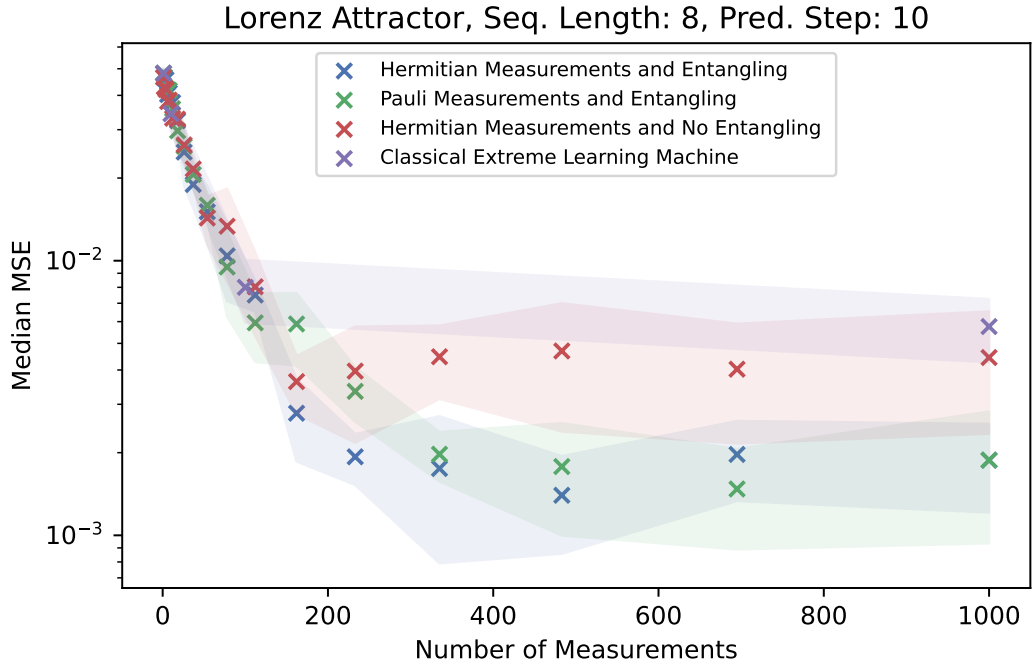

FIGURE 4.7: Here we show the median MSE over the number of measurements for different modifications of the QRC architecture as well as for a classical extreme learning machine. The results shown are obtained by training the models on the Lorenz Attractor data with a sequence length of eight and ten prediction steps. The shaded area indicates the MAD of the median MSE. Blue markers indicate the results of the standard QRC architecture, while green markers show the results of changing the observables to Pauli strings. Red markers illustrate the QRC architecture without entanglement gates in the ansatz. Finally, the purple markers show the results for the classical extreme learning machine.

represent this architecture compared to the standard one represented by blue markers. We found that removing entanglement gates only slightly increases the prediction error, with no significant difference across different datasets, sequence lengths, or prediction steps. This raises questions about the necessity of entanglement for QRC, since encoding data in a product state of single qubits, seems to provide comparable prediction accuracy. In particular, this suggests that the quantum circuit could be efficiently simulated classically, for example by matrix product states.

When we previously investigated the role of entangling gates in the VQC architecture without an input layer, we found that entangling gates improve prediction accuracy in multi-layer approaches by making quantum parameters more independent. However, this does not apply to QRC, which lacks trainable quantum parameters. Thus, the success of QRC may lie in its architecture rather than in exploiting high-dimensional quantum states.

Finally, we compare QRC with a classical counterpart, the ELM. Both methods share the concept of mapping sequential data into a higher dimensional space and measuring M features to train a linear output layer. The ELM results, shown in purple, are optimized for the hidden state dimension. We observe that the ELM performs only slightly worse than QRC, especially for single-step predictions, where it actually outperforms QRC. This may be due to the linear nature of the tasks, where

the simpler preprocessing of the ELM by a linear layer with an activation function benefits the linear output layer more than the sequential encoding in QRC.

However, for more nonlinear tasks, especially on the Mackey-Glass dataset with longer sequence lengths, QRC shows improved prediction accuracy over ELM. This suggests that a quantum structure may offer advantages for such problems. These types of learning tasks are of particular interest and will be discussed in the benchmark study in the next section.

4.3 Comprehensive Benchmark Study

Finally, we put together the results from the previous sections and compare all quantum and classical machine learning models analyzed for time series prediction. This work is fundamental to determining whether quantum models are suitable for accurate time series prediction. In addition, it helps to identify categories of learning problems where certain models excel.

Figure 4.8 is the central figure of our benchmark study. It shows the best median MSE for each model on the test dataset for each combination of dataset, number of prediction steps, and sequence length. The left plots show the results for the Mackey-Glass dataset, while the right plots show the results for the Lorenz Attractor dataset. From top to bottom, the subplots represent an increasing number of prediction steps, with the top plots corresponding to one step and the bottom plots corresponding to 100 steps. Within each subplot, the median MSE on the test dataset for different sequence lengths is shown for each model, with different marker types representing different sequence lengths. The quantum model results are shown on the left side of each subplot, and the classical model results are shown on the right. Error bars show the MAD over ten different initializations. Each result shown is the best result obtained by evaluating all model hyperparameters and architectures. For example, for the VQC, the best median MSE is reported for each specific problem after considering all architectures, number of layers, and number of qubits. This ensures that the results reflect the maximum prediction accuracy achievable for each model and learning task within the investigated range of architectures and hyperparameters. However, we must keep in mind that models differ in the number of hyperparameters and architectures to optimize. For example, the QRNN is only optimized for the number of qubits, so its parameter space is smaller than that of the VQC. This imbalance must be taken into account when interpreting the benchmark results.

Regarding the performance metric, we use the median MSE as in the previous sections. Note that all results and their interpretations are robust to using the median MAE instead, which is less sensitive to outliers. The trends discussed in this section based on the median MSE also hold for the alternative metric. This can be confirmed by comparing Figure 4.8 with Figure A.10 in the appendix, which shows the results for the median MAE.

We first focus on the quantum models. Across all time series prediction problems, the VQC consistently achieves the highest prediction accuracy among all quantum models. This suggests that more advanced architectures, such as those proposed for the QRNN and QLSTM, may not be necessary to achieve strong time series prediction results. Although we could only classically simulate QRNN training for a sequence length of four, we found that resetting the qubits in the data register yielded at best comparable results to omitting the reset. In this scenario, the architecture becomes similar to a VQC, potentially rendering the use of a QRNN

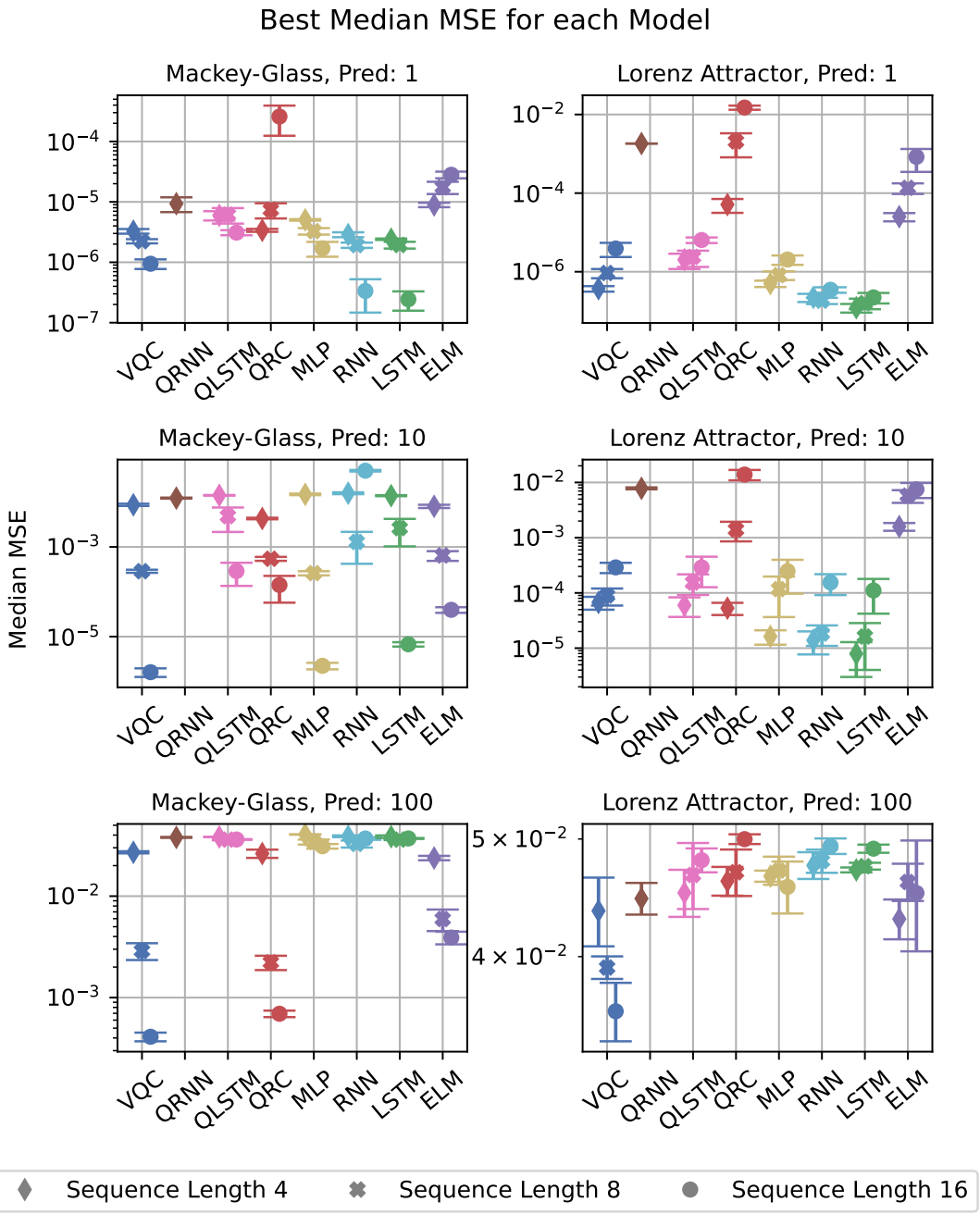

FIGURE 4.8: Best median MSE for each model on the test dataset across combinations of datasets, prediction steps, and sequence lengths. The left panels show the results for the Mackey-Glass dataset, while the right panels show results for the Lorenz Attractor dataset. From top to bottom, the subplots represent increasing prediction steps. Within each subplot, median MSEs for different sequence lengths are shown for each model, with different marker types indicating various sequence lengths. Quantum models appear on the left side of each subplot, classical models on the right. Error bars represent the MAD over ten random initializations. The results reflect the best performance from all tested model architectures and hyperparameters.

for time series prediction unnecessary. The QLSTM, which includes multiple VQCs, requires more resources when running on quantum hardware compared to a single VQC. In addition, within a QLSTM cell, the quantum state is measured after each VQC, discarding a significant amount of information stored in the quantum state. This could prevent the model from taking full advantage of learning in a quantum Hilbert space. Therefore, for time series prediction tasks using variational quantum models, a simpler VQC may be the most efficient choice. The QRC architecture shows large fluctuations in prediction accuracy over different sequence lengths. Furthermore, the accuracy is only comparable to that of a VQC for some of the problems studied. This is the case, for example, for training Lorenz Attractor data for ten prediction steps and Mackey-Glass data for 100 prediction steps. However, the QRC architecture is much easier to run on quantum hardware and may not be limited in its training capability by barren plateaus contrary to a VQC. Therefore, it is of interest to further investigate QRC for these problems.

We now also consider the classical machine learning models. We find that almost all machine learning models perform well for one-step predictions on both datasets as well as for ten prediction steps on the Lorenz Attractor dataset. However, the LSTM and the RNN have the best prediction accuracy. For these linear problems, the classical machine learning models designed to process sequential data seem to learn the data best. The MLP also performs well for these problems. We want to emphasise again, that we are only able to train a small subset of all possible MLP structures with different number of layers and different number of hidden sizes. It is likely that there exist other MLP architectures that perform better on the data tested here. The quantum machine learning models can at best keep up with the classical models. This suggests that quantum models may not be advantageous for sequential learning tasks with highly linear data. Also for this data, the reservoir based architectures have a higher prediction error compared to the other models included in this study.

For the prediction of ten steps ahead on the Mackey-Glass dataset, we observe a slightly different behavior. The VQC and the MLP archive a comparable accuracy, followed by the LSTM. However, the RNN is no longer able to predict with an accuracy competitive with other models. The QRC and ELM models now attain comparable results, at least for smaller sequence lengths.

Before discussing the results for 100-step prediction, we want to note a common observation for each dataset. For the Mackey-Glass dataset, prediction accuracy consistently improves as sequence length increases. On the other hand, the opposite is true for the Lorenz Attractor dataset. There, the accuracy decreases for longer sequence lengths. We interpret this as follows. The Mackey-Glass dataset has explicit memory in its data. It may be that encoding longer sequences allows the models to learn these explicit correlations in the data. On the other hand, shorter sequences leading to better results may indicate that there are no learnable features beyond linear relationships in the data. In this case, longer sequences confound the models, since short sequences are sufficient to achieve high prediction accuracy for linear learning tasks.

This hypothesis is further supported by considering the results for the Lorenz Attractor dataset and 100 prediction steps. Almost all models attain identical prediction accuracy within the error range. For the VQC model, the figure might suggest that this architecture outperforms all other models. However, considering the scale of the y-axis, the difference is not significant. This suggests that the models may not be able to learn any further relevant features in the data. Potentially all models

achieve a prediction accuracy where the maximal informational content of the data is exploited.

Finally, we move our attention to the subplot containing the results for the Mackey-Glass dataset with 100 prediction steps. Here, the classical machine learning models, as well as QRNN and LSTM, do not seem to be able to achieve any meaningful prediction accuracy. However, the VQC model is able to significantly outperform these models by about two orders of magnitude for long sequence lengths. This result is achieved by a VQC model with a data re-uploading architecture. The reservoir-based architectures are also able to significantly outperform the other models included in the study. Since both QRC and ELM perform well, the advantage over other models may lie in the structure of the mapping of the data into the higher dimensional space itself. Since the VQC with a re-uploading structure and the QRC have a similar circuit structure, there may also be an advantage in using such a circuit for this task. Whether there is an inherent quantum mechanical advantage in using these quantum models remains an open question for further research, but is beyond the scope of this work.

We want to further investigate this particular learning problem to gain deeper insights and to ensure that the models are large enough to learn the nonlinear data. Figure 4.9 shows the median MSE as a function of the number of trainable parameters in each model. By trainable parameters, we mean all parameters that are optimized during the training process. For models with both quantum and classical parameters, we sum these parameters. The results are obtained by searching for the optimal architecture and hyperparameters within 15 logarithmically spaced intervals of parameters ranging from 1 to 20,000. We present results for the most relevant models in this dataset: the VQC, the QRC and the ELM. The LSTM is included as a representative of models that perform poorly on this task. We choose the LSTM because it has the best overall performance on other learning problems considered in this study. Again, the error bars represent the MAD over ten random model initializations. Figures A.11 and A.12 in the appendix show all plots of the median MSE versus the number of parameters for all models and all other learning problems.

We focus first on the reservoir-based models. For both the QRC and the ELM, the number of trainable parameters is primarily equal to the number of measurements from the reservoir, plus a parameter for the bias in the linear layer. This is because for the one-dimensional data, we are mapping from a space with dimensionality equal to the number of measurements to a one-dimensional output through the linear layer. As the number of measurements increases, the prediction accuracy for both models improves, potentially plateauing at a larger number of measurements. For comparable sizes of trainable output layers, within the range of parameters where prediction accuracy improves, the QRC outperforms the ELM in this task. Regarding the VQC, it is important to note that different architectures are considered in this study. In particular, the two VQC results shown in Figure 4.9, with about 100 trainable parameters, do not achieve high prediction accuracy compared to the best reservoir-based architectures. These correspond to VQC architectures with a classical input layer. The results with more parameters correspond to the architecture with data re-uploading. This suggests that there may be an inherent advantage in using re-uploading structures for this task, since the QRC circuit design also follows a similar arrangement. For the LSTM model, we observe that for different combinations of neural network layers and hidden layer sizes, there is no significant variation in prediction accuracy as system size increases. This suggests that the LSTM architecture may not be well suited for learning this particular dataset. While it is possible that larger systems may show improvement, we argue that this is unlikely

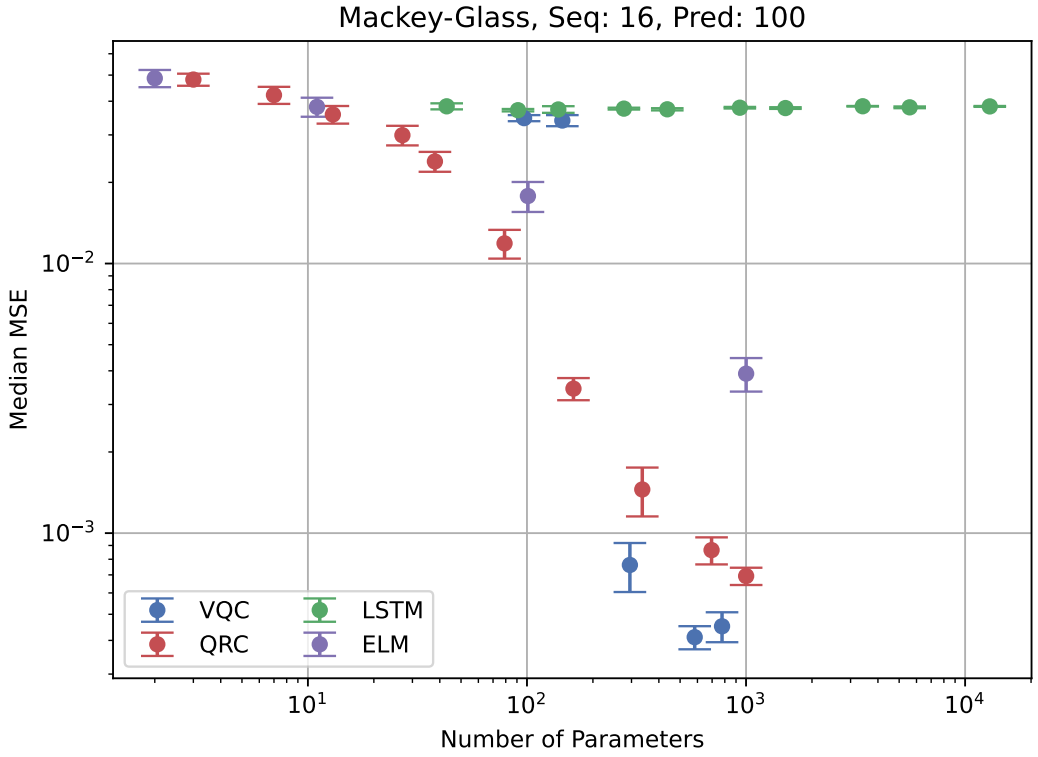

FIGURE 4.9: For the Mackey-Glass dataset, 100 prediction steps, and a sequence length of 16, we plot the median MSE over the number of trainable parameters for the VQC, QRC, LSTM, and ELM. The results shown are obtained by searching for the optimal architecture and hyperparameters of each model within 15 logarithmically spaced intervals of parameters ranging from 1 to 20,000. Error bars correspond to the MAD of ten different model initializations.

because we trained LSTM models with parameter counts that are more than an order of magnitude higher than other models tested here. For this dataset, the QRC and the re-uploading VQC achieve the highest prediction accuracy of all the models tested, within a range of relatively small parameter counts that undergo training. It is important to reiterate that these results represent only a subset of all the learning problems studied. As discussed earlier, model performance is highly dependent on the dataset and other problem-specific hyperparameters.

To get a final overview of the prediction capabilities of the different models across all datasets, prediction steps, and sequence lengths tested, we rank the models for each learning problem. For each model, we search for the architecture and hyperparameter set that yields the best median MSE. We then rank the models by their individual best median MSE for each learning problem. Since we have two datasets and have analyzed all models for three different sequence lengths and three different numbers of prediction steps, we have $2 \cdot 3 \cdot 3 = 18$ learning problems. We show the results of this comparison in Figure 4.10. Different rankings are indicated by different colors, ranging from deep blue for a model that ranks first in a learning problem to deep red for a model that ranks eighth. While we investigate eight different models in this study, we are only able to consider the QRNN for sequence lengths of four. Thus, only for six of the learning problems all eight models are part of the ranking. For problems with sequence lengths of eight or 16, the other models

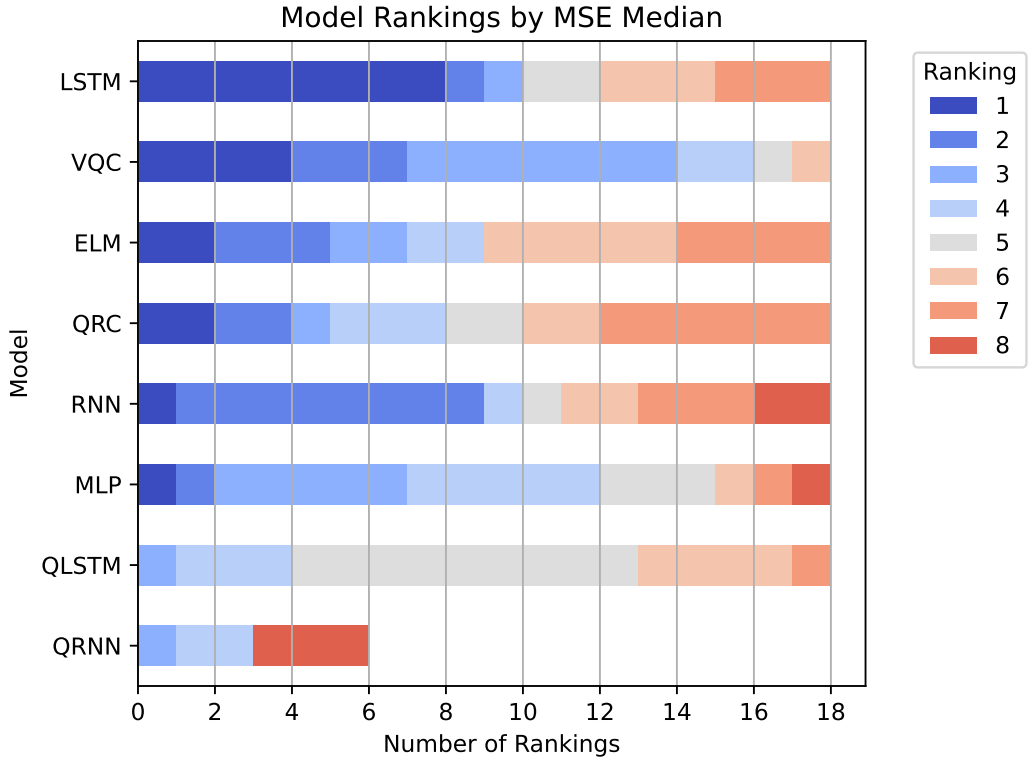

FIGURE 4.10: Ranking of the eight different models based on their best median MSE across all 18 learning problems obtained by combining two datasets, three sequence lengths, and three prediction steps. For each learning problem, the models are ranked according to their best median MSE, which is determined after selecting the optimal architecture and hyperparameter set for each model. Colors represent ranks, from deep blue (1st) to deep red (8th). For sequence lengths of eight and 16, only seven models are ranked because the QRNN is only trained for a sequence length of four. Models are ranked by the number of first place rankings, and ties are broken by second place rankings. The figure provides an overall comparison of the time series prediction performance of the models across all learning problems.

are ranked from first to seventh. In the figure, the models are ordered from top to bottom according to their number of first rankings. If different models have the same number of first rankings, the number of second rankings is taken into account. We acknowledge that this does not give a detailed insight into the performance of the models on specific data, and that it is highly dependent on how the learning tasks are constructed. However, this representation gives a final overview of the benchmark of the time series prediction capability of the studied quantum and classical models for the problems studied. We treat this result as a competitive study of the models for the different learning problems considered.

We see that the LSTM archives the highest number of first rankings. This is due to the high accuracy achieved by the LSTM for more linear tasks, as seen in Figure 4.8. Next, the VQC architectures archive competitive time series prediction results across all learning problems. For other classical models, as well as for the QRC, prediction performance is highly dependent on the learning task, resulting in an intermediate ranking among all models tested. Finally, the two quantum machine learning

models QRNN and QLSTM, which are specifically designed for learning sequential data, are not competitive with the other models. For the QRNN, we found that not resetting qubits in the data register leads to higher prediction accuracy, making the QRNN similar to a VQC. Furthermore, the QLSTM, due to its internal architecture of repeatedly measuring quantum circuits, is unlikely to benefit from learning in a quantum representation of the data. This suggests that these models may not be the best choice for time series prediction tasks.

Nevertheless, for certain, presumably nonlinear learning tasks, there may be a benefit in employing quantum structures as a VQC or a QRC. Further research should focus on further categorizing these classes of time series prediction problems and providing a path to understanding the underlying reason for the observed learning capability.

Chapter 5

Conclusion and Outlook

In this work we investigated the potential in employing quantum machine learning models for time series predictions. This work was motivated by the significance of predicting temporal data in a broad range of fields such as methodology, medicine and finance. As quantum computing has been proven advantageous for some tasks compared to classical methods it is reasonable to explore the time series prediction capability with quantum computing techniques. We here conclude our work on the first large-scale benchmark study comparing different quantum and classical models for time series prediction.

We started by individually analyzing different quantum machine learning models. Distinct VQC architectures were investigated where we found that entangling gates can play a role in enabling parameters of the quantum circuit to contribute to learning data and therefore increasing prediction accuracy. Moreover, we observed that the efficacy of different architectures varies depending on the dataset in question. The QRNN proposed in the literature was compared to an approach where data qubits are not reset. The results showed that not resetting the qubits overall leads to better prediction accuracy. This approach is similar to employing a VQC which implies that the initially proposed model might not be essential. The QLSTM model showed inferior performance compared to its classical counterpart in the context of this study. Although the inclusion of linear layers in the architecture design improved prediction accuracy, these models exhibited inferior performance compared to their classical counterpart. This observation suggests that there may be no discernible advantage to replacing classical neural layers within the LSTM architecture with quantum neural networks. In addition, in all QLSTM architectures, the information encoded in a quantum state is repeatedly lost by measuring a quantum circuit within each LSTM cell. This process can hinder the model's ability to take advantage of the potential benefits associated with learning in Hilbert space.

A comparative analysis of all models studied in the central benchmark revealed that, for tasks involving only forecasting a small number of time steps ahead, quantum models exhibit prediction accuracy that, at best, match with that of classical methods. However, for more complex prediction tasks, especially predicting up to 100 time steps ahead on the Mackey-Glass dataset, the VQC data re-uploading architecture, as well as the QRC architecture proposed by us, demonstrate significantly higher prediction accuracy than all other models investigated, including classical ones.

Further research is required to investigate the specific classes of time series prediction problems where quantum models demonstrate superior performance. This approach may help identify the key factors behind these models' predictive performance. In order to ascertain whether the learning success roots in inherit quantum mechanical properties, it is necessary to examine various quantum mechanical properties, like the entanglement entropy [93] and the Quantum Fisher Information [94].

In light of the recent challenges encountered in variational QML, with the emergence of barren plateaus, training VQCs comprising a substantial number of qubits appears to be impractical due to vanishing gradients [51]. Consequently, quantum reservoir computing is a promising avenue of research, as they are likely to not being affected by barren plateaus.

While we have been dealing with classical data throughout this thesis, an interesting direction of research is to look at learning with quantum mechanical data. For example, one can use quantum machine learning methods to classify quantum phases [95] or to reconstruct quantum states [96]. While research has mostly focused on classification tasks for quantum data, work on learning sequential quantum data such as temporal quantum tomography [97] is rare. Quantum models may benefit from their intrinsic quantum mechanical structure when learning quantum mechanical data.

Further investigation into these research avenues may facilitate a more comprehensive understanding of quantum machine learning models, potentially leading to the identification of practical applications for quantum computers.

Appendix A

Additional Results

A.1 Variational Quantum Circuits

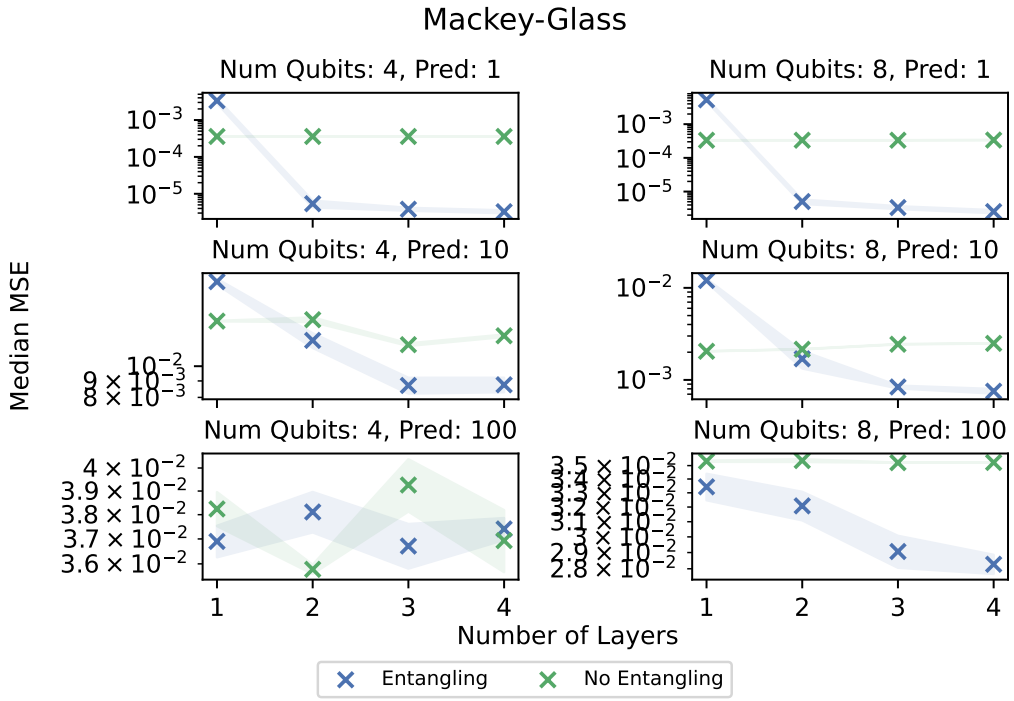

FIGURE A.1: Median MSE over the number of layers in the VQC architecture with no classical input layer. Results are for different number of qubits and different prediction steps on the Mackey-Glass dataset.

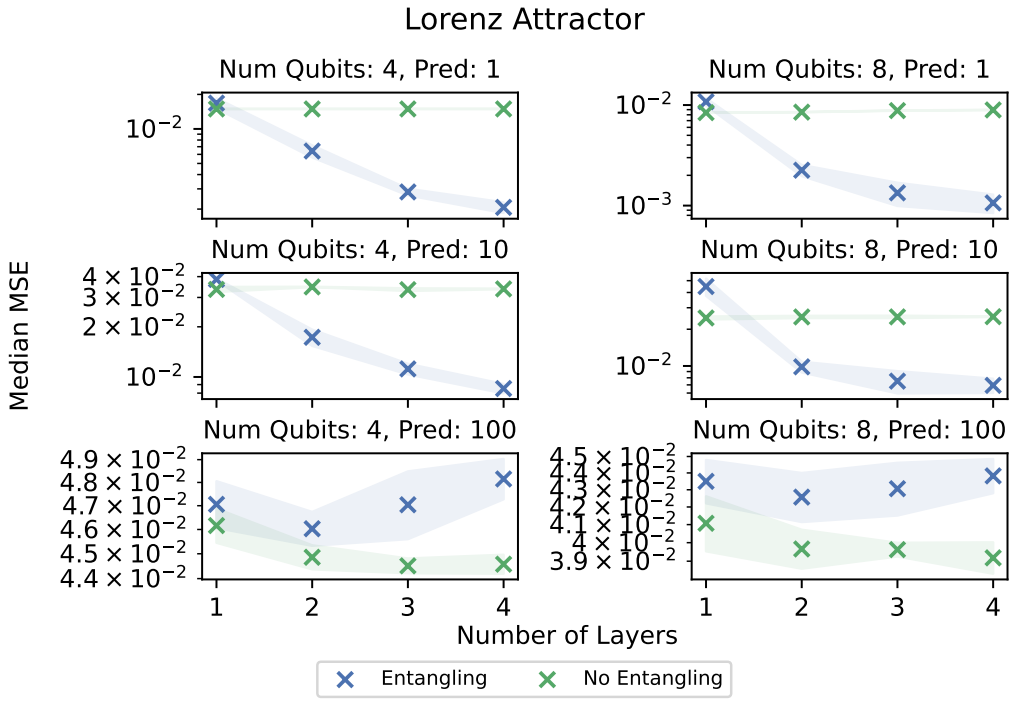

FIGURE A.2: Median MSE over the number of layers in the VQC architecture with no classical input layer. Results are for different number of qubits and different prediction steps on the Lorenz Attractor dataset.

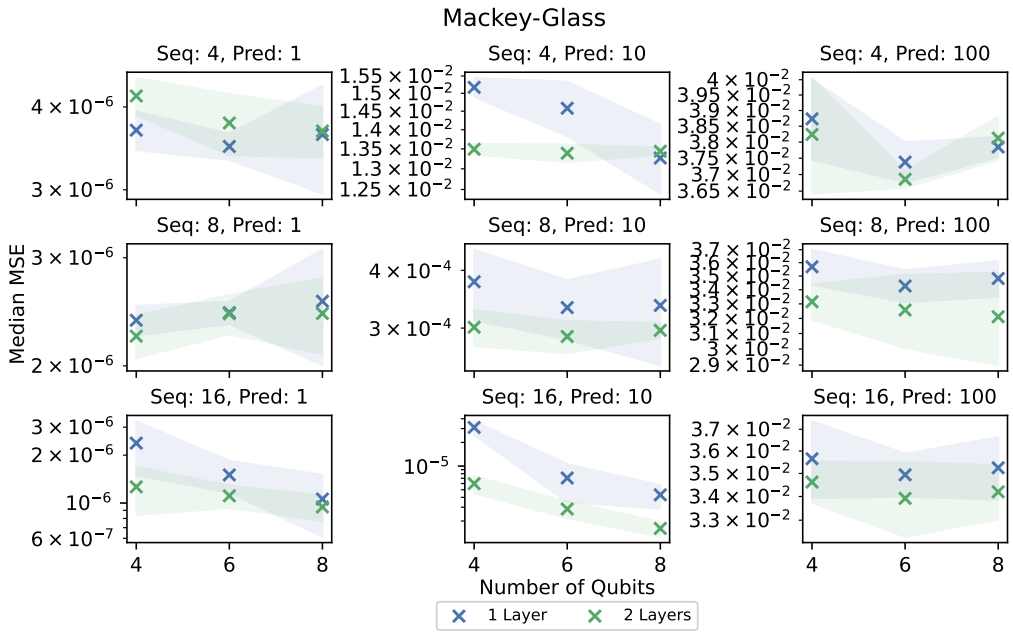

FIGURE A.3: Median MSE over the number of qubits in the VQC architecture with a classical input layer. Results are for different sequence lengths and different prediction steps on the Mackey-Glass dataset.

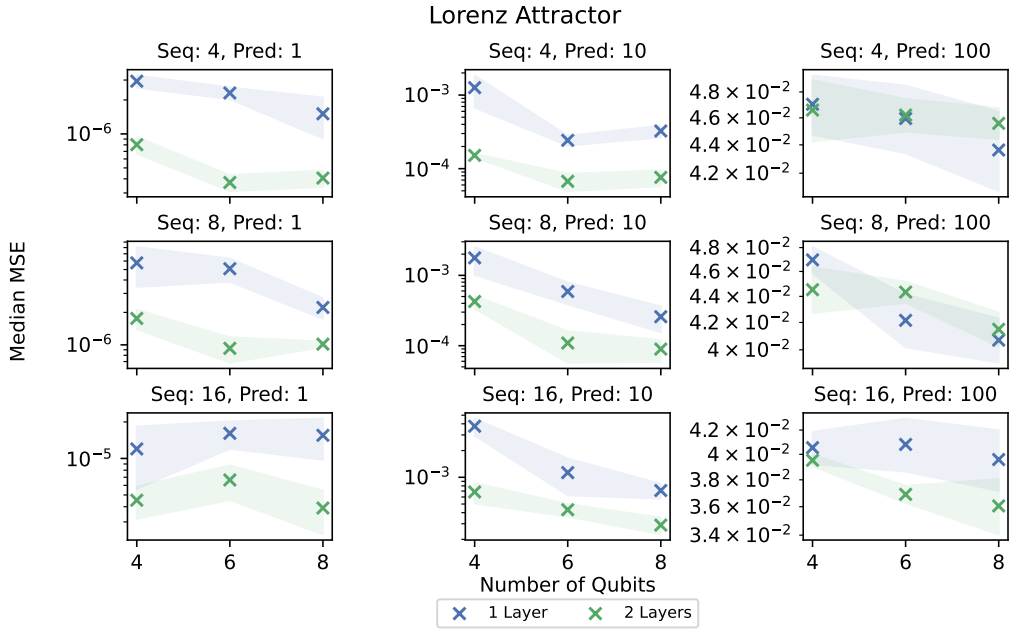

FIGURE A.4: Median MSE over the number of qubits in the VQC architecture with a classical input layer. Results are for different sequence lengths and different prediction steps on the Lorenz-Attractor dataset.

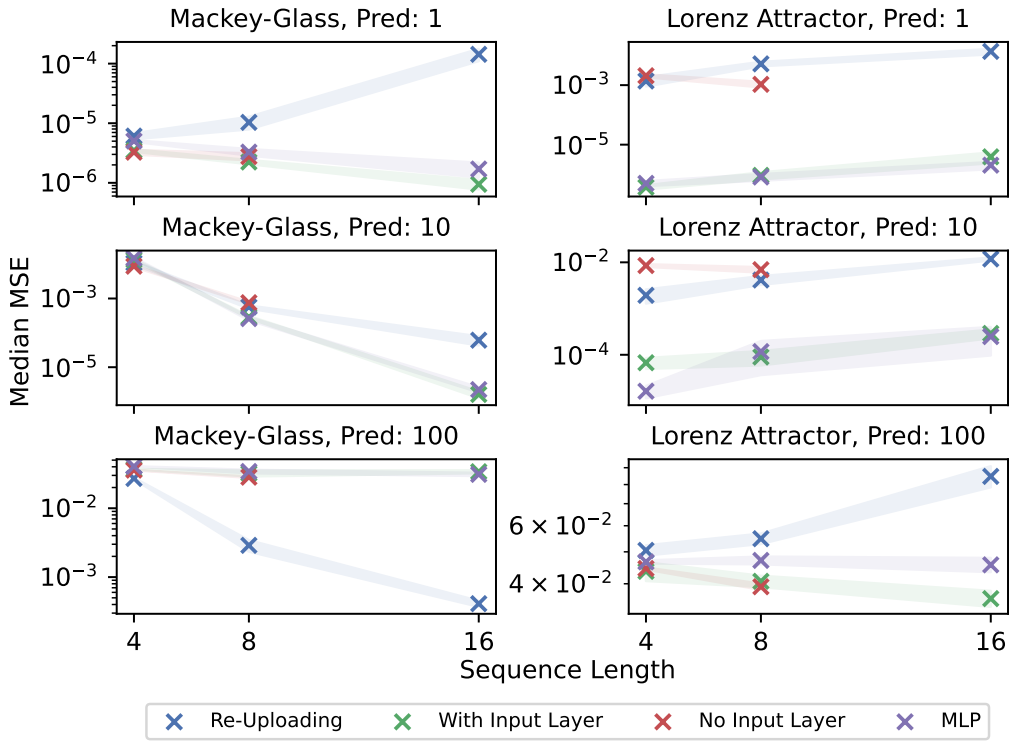

FIGURE A.5: Median MSE over the sequence length for the different VQC architectures and the MLP model. Shown is the best result after a hyperparameter optimization according to Table 3.3.

A.2 Quantum Recurrent Neural Network

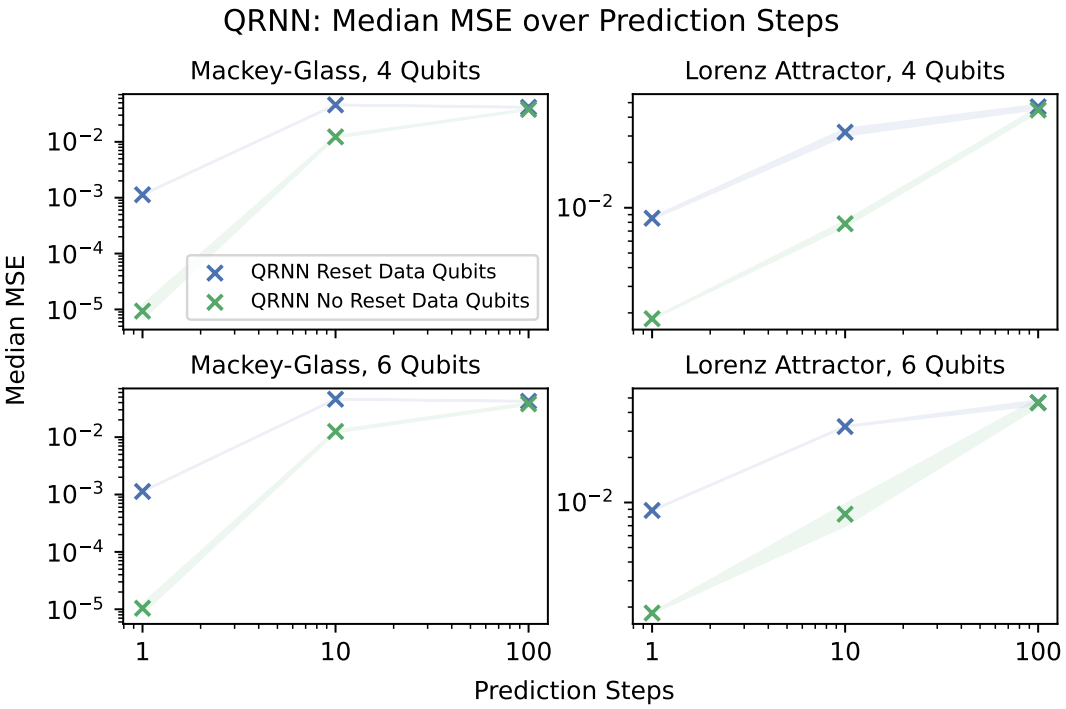

FIGURE A.6: Median MSE over the number of prediction steps for different data sets and different number of qubits. The results are shown for the originally proposed QRNN architecture [15] as well as a architecture where qubits in the data register are not reset after each sequence step.

A.3 Quantum Long Short-Term Memory

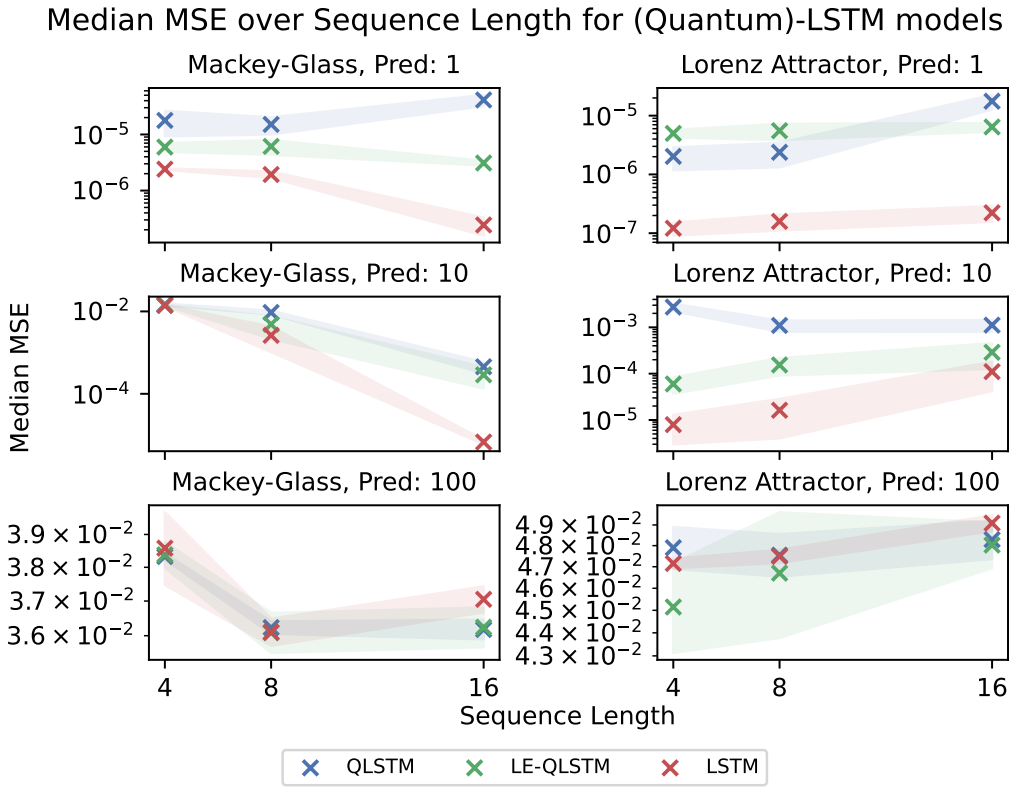

FIGURE A.7: For the different (Quantum) LSTMs, the median MSE is plotted of the sequence length. The subplots show the results for different data and different prediction steps.

A.4 Quantum Reservoir Computing

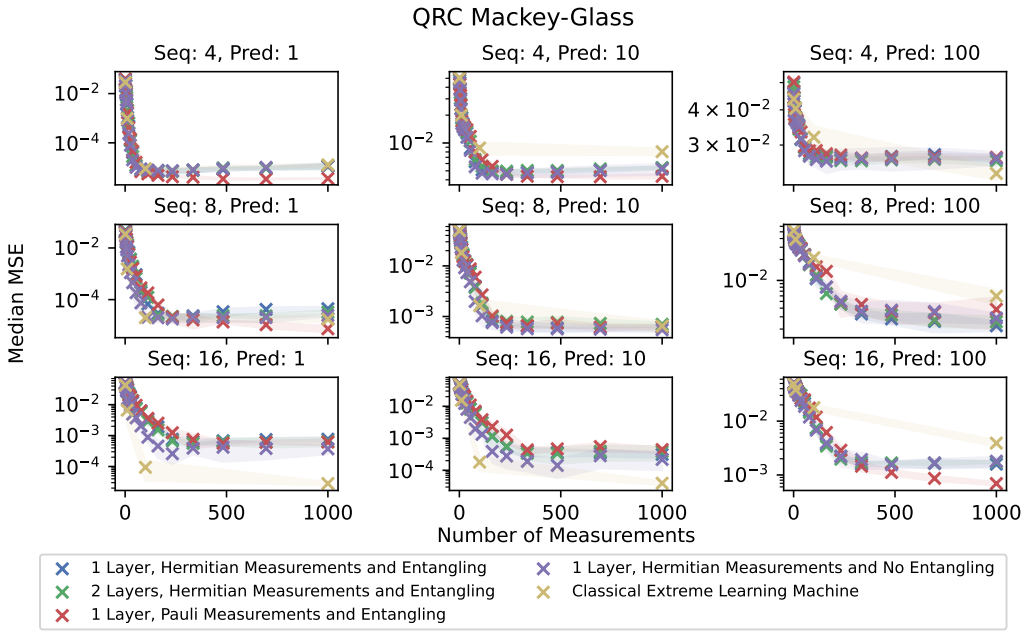

FIGURE A.8: Results for different QRC architecture modifications and the classical ELM. Shown is the median MSE over the number of measurements for different sequence lengths and prediction steps on the Mackey-Glass data set.

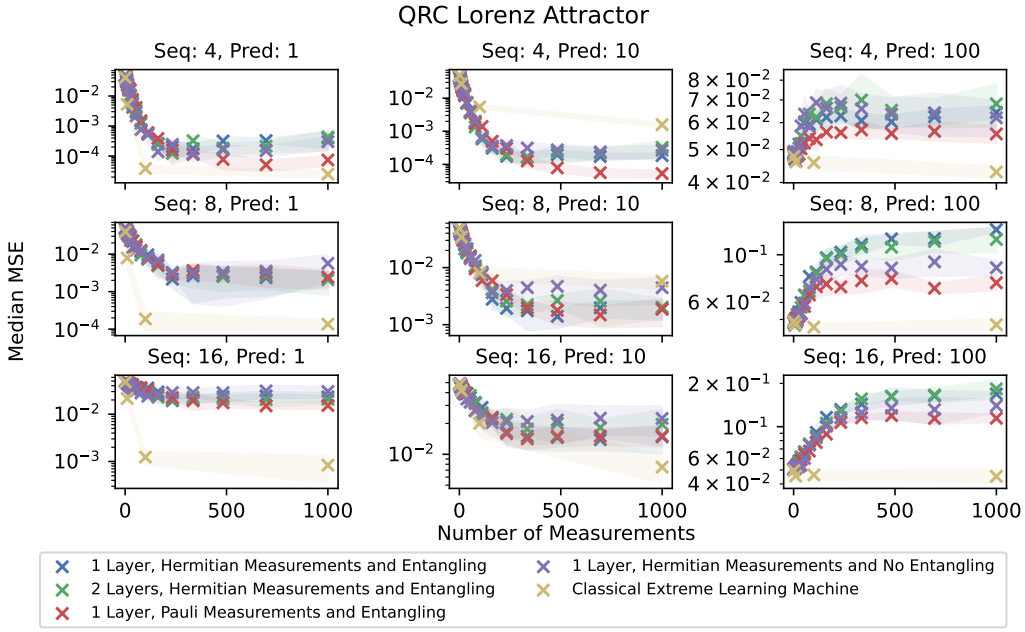

FIGURE A.9: Results for different QRC architecture modifications and the classical ELM. Shown is the median MSE over the number of measurements for different sequence lengths and prediction steps on the Lorenz Attractor data set.

A.5 Benchmark

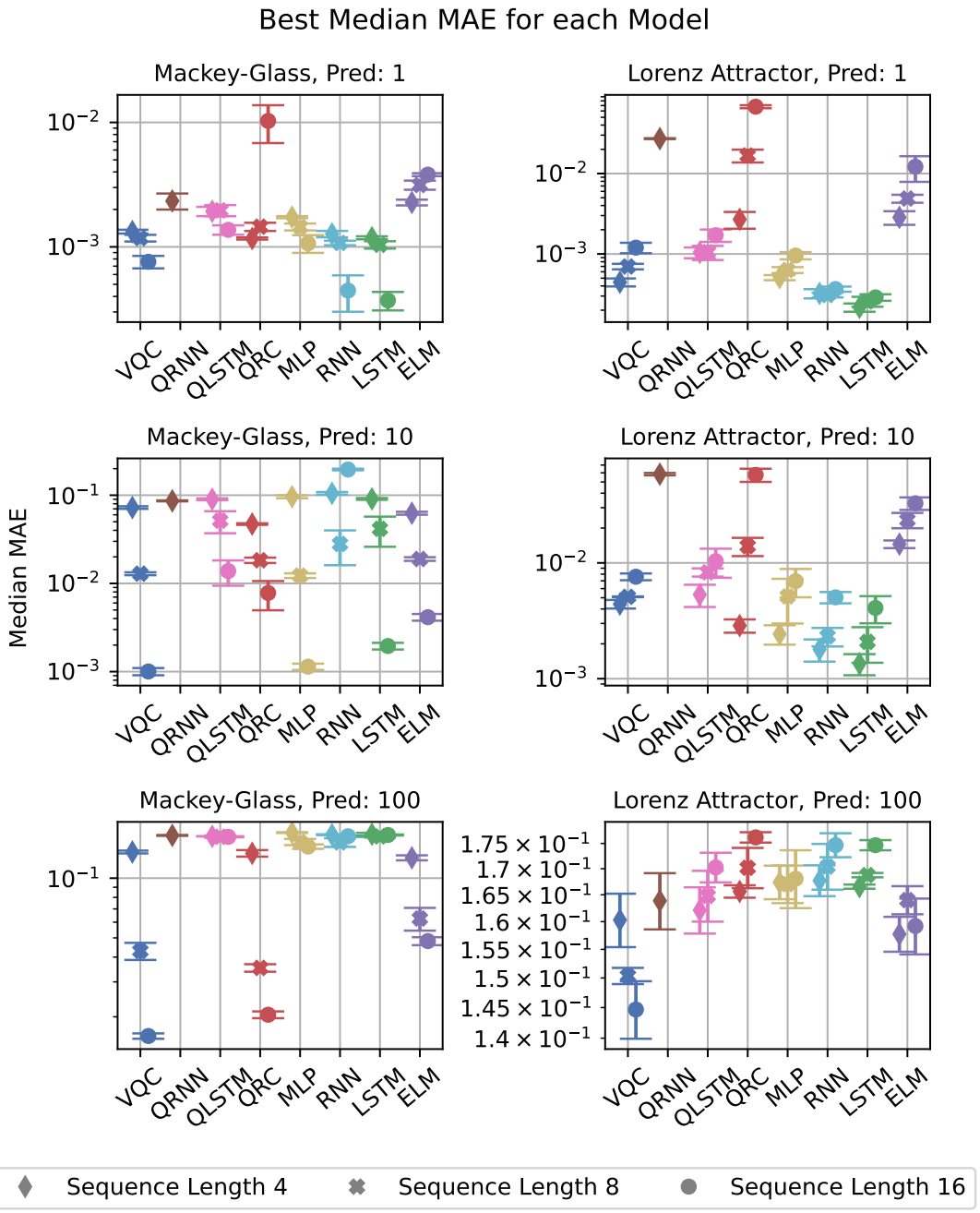

FIGURE A.10: This is the same figure as Figure 4.8, but instead of the median MSE, here the median MAE is employed as a metric for measuring the prediction accuracy.

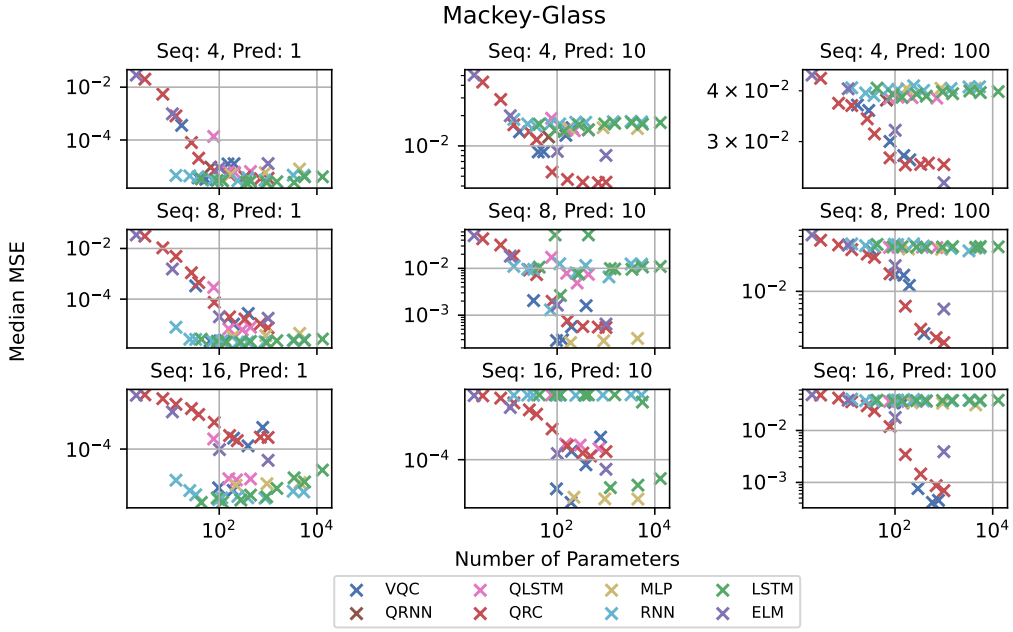

FIGURE A.11: We plot the median MSE over the number of trainable parameters for different models. The results are obtained by searching for the optimal architecture and hyperparameters of each model within 15 logarithmically spaced intervals of parameters ranging from 1 to 20,000. This plot shows the results on the Mackey-Glass dataset.

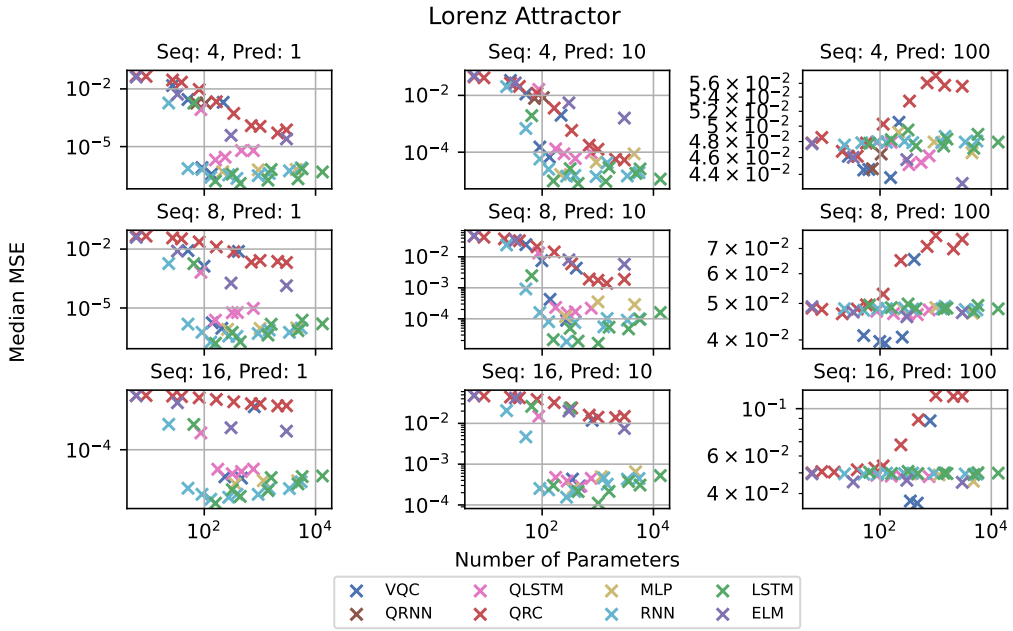

FIGURE A.12: We plot the median MSE over the number of trainable parameters for different models. The results are obtained by searching for the optimal architecture and hyperparameters of each model within 15 logarithmically spaced intervals of parameters ranging from 1 to 20,000. This plot shows the results on the Lorenz Attractor dataset.

Bibliography

[1] Changqing Cheng et al. "Time series forecasting for nonlinear and non-stationary processes: a review and comparative study". In: *IIE Transactions* 47.10 (Oct. 3, 2015), pp. 1053–1071. DOI: 10.1080/0740817X.2014.999180.

[2] Chirag Deb et al. "A review on time series forecasting techniques for building energy consumption". In: *Renewable and Sustainable Energy Reviews* 74 (July 1, 2017), pp. 902–924. DOI: 10.1016/j.rser.2017.02.085.

[3] Ruey S. Tsay. "Time Series and Forecasting: Brief History and Future Research". In: *Journal of the American Statistical Association* 95.450 (June 2000), pp. 638–643. DOI: 10.1080/01621459.2000.10474241.

[4] Zhongyang Han et al. "A Review of Deep Learning Models for Time Series Prediction". In: *IEEE Sensors Journal* 21.6 (Mar. 2021), pp. 7833–7848. DOI: 10.1109/JSEN.2019.2923982.

[5] Antonio Rafael Sabino Parmezan, Vinicius M. A. Souza, and Gustavo E. A. P. A. Batista. "Evaluation of statistical and machine learning models for time series prediction: Identifying the state-of-the-art and the best conditions for the use of each model". In: *Information Sciences* 484 (May 1, 2019), pp. 302–337. DOI: 10.1016/j.ins.2019.01.076.

[6] Ahmed Tealab. "Time series forecasting using artificial neural networks methodologies: A systematic review". In: *Future Computing and Informatics Journal* 3.2 (Dec. 1, 2018), pp. 334–340. DOI: 10.1016/j.fcij.2018.10.003.

[7] Peter W. Shor. "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer". In: *SIAM Journal on Computing* 26.5 (Oct. 1997), pp. 1484–1509. DOI: 10.1137/S0097539795293172.

[8] Lov K. Grover. "A fast quantum mechanical algorithm for database search". In: *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing - STOC '96*. Philadelphia, Pennsylvania, United States: ACM Press, 1996, pp. 212–219. DOI: 10.1145/237814.237866.

[9] Sukhpal Singh Gill et al. "Quantum computing: A taxonomy, systematic review and future directions". In: *Software: Practice and Experience* 52.1 (2022), pp. 66–114. DOI: 10.1002/spe.3039.

[10] Sandeep Kumar Sood and Pooja. "Quantum Computing Review: A Decade of Research". In: *IEEE Transactions on Engineering Management* 71 (2024), pp. 6662–6676. DOI: 10.1109/TEM.2023.3284689.

[11] Jacob Biamonte et al. "Quantum machine learning". In: *Nature* 549.7671 (Sept. 2017), pp. 195–202. DOI: 10.1038/nature23474.

[12] Maria Schuld, Ilya Sinayskiy, and Francesco Petruccione. "An introduction to quantum machine learning". In: *Contemporary Physics* 56.2 (Apr. 3, 2015), pp. 172–185. DOI: 10.1080/00107514.2014.964942.

[13] Mayra Alejandra Rivera-Ruiz, Andres Mendez-Vazquez, and José Mauricio López-Romero. "Time Series Forecasting with Quantum Machine Learning Architectures". In: *Advances in Computational Intelligence*. Cham: Springer Nature Switzerland, 2022, pp. 66–82. DOI: 10.1007/978-3-031-19493-1_6.

[14] Yuto Takaki et al. "Learning temporal data with a variational quantum recurrent neural network". In: *Physical Review A* 103.5 (May 13, 2021), p. 052414. DOI: 10.1103/PhysRevA.103.052414.

[15] Yanan Li et al. "Quantum recurrent neural networks for sequential learning". In: *Neural Networks* 166 (Sept. 1, 2023), pp. 148–161. DOI: 10.1016/j.neunet.2023.07.003.

[16] Samuel Yen-Chi Chen, Shinjae Yoo, and Yao-Lung L. Fang. "Quantum Long Short-Term Memory". In: *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. May 2022, pp. 8622–8626. DOI: 10.1109/ICASSP43922.2022.9747369.

[17] Yuji Cao et al. "Linear-layer-enhanced quantum long short-term memory for carbon price forecasting". In: *Quantum Machine Intelligence* 5.2 (July 5, 2023), p. 26. DOI: 10.1007/s42484-023-00115-2.

[18] Maria Schuld and Nathan Killoran. "Is Quantum Advantage the Right Goal for Quantum Machine Learning?" In: *PRX Quantum* 3.3 (July 14, 2022), p. 030101. DOI: 10.1103/PRXQuantum.3.030101.

[19] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: 2016, pp. 770–778.

[20] Karen Simonyan and Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. Apr. 10, 2015. DOI: 10.48550/arXiv.1409.1556.

[21] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems*. Vol. 25. Curran Associates, Inc., 2012.

[22] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory". In: *Neural Computation* 9.8 (Nov. 1997), pp. 1735–1780. DOI: 10.1162/neco.1997.9.8.1735.

[23] Ashish Vaswani et al. "Attention is All you Need". In: *Advances in Neural Information Processing Systems*. Vol. 30. Curran Associates, Inc., 2017.

[24] Amanpreet Singh, Narina Thakur, and Aakanksha Sharma. "A review of supervised machine learning algorithms". In: *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*. Mar. 2016, pp. 1310–1315.

[25] Muhammad Usama et al. "Unsupervised Machine Learning for Networking: Techniques, Applications and Research Challenges". In: *IEEE Access* 7 (2019), pp. 65579–65615. DOI: 10.1109/ACCESS.2019.2916648.

[26] Maria Schuld and Francesco Petruccione. *Machine Learning with Quantum Computers*. Cham: Springer International Publishing, 2021. DOI: 10.1007/978-3-030-83098-4.

[27] Léon Bottou. "Large-Scale Machine Learning with Stochastic Gradient Descent". In: *Proceedings of COMPSTAT'2010*. Ed. by Yves Lechevallier and Gilbert Saporta. Heidelberg: Physica-Verlag HD, 2010, pp. 177–186. DOI: 10.1007/978-3-7908-2604-3_16.

[28] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. Jan. 29, 2017. DOI: 10.48550/arXiv.1412.6980.

[29] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. "Learning representations by back-propagating errors". In: *Nature* 323.6088 (Oct. 1986), pp. 533–536. DOI: 10.1038/323533a0.

[30] Michael I. Jordan. "Chapter 25 - Serial Order: A Parallel Distributed Processing Approach". In: *Advances in Psychology*. Ed. by John W. Donahoe and Vivian Packard Dorsel. Vol. 121. North-Holland, Jan. 1, 1997, pp. 471–495. DOI: 10.1016/S0166-4115(97)80111-2.

[31] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, Dec. 9, 2010.

[32] M. Cerezo et al. "Variational Quantum Algorithms". In: *Nature Reviews Physics* 3.9 (Aug. 12, 2021), pp. 625–644. DOI: 10.1038/s42254-021-00348-9.

[33] Maria Schuld and Nathan Killoran. "Quantum Machine Learning in Feature Hilbert Spaces". In: *Physical Review Letters* 122.4 (Feb. 1, 2019), p. 040504. DOI: 10.1103/PhysRevLett.122.040504.

[34] Maria Schuld, Ryan Sweke, and Johannes Jakob Meyer. "The effect of data encoding on the expressive power of variational quantum machine learning models". In: *Physical Review A* 103.3 (Mar. 24, 2021), p. 032430. DOI: 10.1103/PhysRevA.103.032430.

[35] Lennart Bittel, Sevag Gharibian, and Martin Kliesch. "Optimizing the depth of variational quantum algorithms is strongly QCMA-hard to approximate". In: *LIPIcs, Volume 264, CCC 2023* 264 (2023), 34:1–34:24. DOI: 10.4230/LIPIcs.CCC.2023.34.

[36] Anbang Wu et al. *Towards Efficient Ansatz Architecture for Variational Quantum Algorithms*. Nov. 26, 2021. DOI: 10.48550/arXiv.2111.13730.

[37] Hsin-Yuan Huang et al. "Power of data in quantum machine learning". In: *Nature Communications* 12.1 (May 11, 2021), p. 2631. DOI: 10.1038/s41467-021-22539-9.

[38] Seth Lloyd et al. *Quantum embeddings for machine learning*. Feb. 10, 2020. DOI: 10.48550/arXiv.2001.03622.

[39] M. P. Cuéllar et al. "Time series quantum classifiers with amplitude embedding". In: *Quantum Machine Intelligence* 5.2 (Nov. 27, 2023), p. 45. DOI: 10.1007/s42484-023-00133-0.

[40] M. Cerezo et al. "Challenges and opportunities in quantum machine learning". In: *Nature Computational Science* 2.9 (Sept. 2022), pp. 567–576. DOI: 10.1038/s43588-022-00311-3.

[41] Mateusz Ostaszewski et al. "Reinforcement learning for optimization of variational quantum circuit architectures". In: *Advances in Neural Information Processing Systems*. Vol. 34. Curran Associates, Inc., 2021, pp. 18182–18194.

[42] Frederic Rapp et al. *Reinforcement learning-based architecture search for quantum machine learning*. Aug. 5, 2024. DOI: 10.48550/arXiv.2406.02717.

[43] Yuxuan Du et al. "Quantum circuit architecture search for variational quantum algorithms". In: *npj Quantum Information* 8.1 (May 23, 2022), pp. 1–8. DOI: 10.1038/s41534-022-00570-y.

[44] Nathalie P. de Leon et al. "Materials challenges and opportunities for quantum computing hardware". In: *Science* 372.6539 (Apr. 16, 2021), eabb2823. DOI: 10.1126/science.abb2823.

[45] Emanuel Knill and Raymond Laflamme. "Theory of quantum error-correcting codes". In: *Physical Review A* 55.2 (Feb. 1, 1997), pp. 900–911. DOI: 10.1103/PhysRevA.55.900.

[46] Barbara M. Terhal. "Quantum error correction for quantum memories". In: *Reviews of Modern Physics* 87.2 (Apr. 7, 2015), pp. 307–346. DOI: 10.1103/RevModPhys.87.307.

[47] Samson Wang et al. "Noise-induced barren plateaus in variational quantum algorithms". In: *Nature Communications* 12.1 (Nov. 29, 2021), p. 6961. DOI: 10.1038/s41467-021-27045-6.

[48] Martin Larocca et al. *A Review of Barren Plateaus in Variational Quantum Computing*. May 1, 2024.

[49] K. Mitarai et al. "Quantum circuit learning". In: *Physical Review A* 98.3 (Sept. 10, 2018), p. 032309. DOI: 10.1103/PhysRevA.98.032309.

[50] Maria Schuld et al. "Evaluating analytic gradients on quantum hardware". In: *Physical Review A* 99.3 (Mar. 21, 2019), p. 032331. DOI: 10.1103/PhysRevA.99.032331.

[51] Jarrod R. McClean et al. "Barren plateaus in quantum neural network training landscapes". In: *Nature Communications* 9.1 (Nov. 16, 2018), p. 4812. DOI: 10.1038/s41467-018-07090-4.

[52] Amira Abbas et al. "The power of quantum neural networks". In: *Nature Computational Science* 1.6 (June 24, 2021), pp. 403–409. DOI: 10.1038/s43588-021-00084-1.

[53] Tobias Haug, Kishor Bharti, and M.S. Kim. "Capacity and Quantum Geometry of Parametrized Quantum Circuits". In: *PRX Quantum* 2.4 (Oct. 14, 2021), p. 040309. DOI: 10.1103/PRXQuantum.2.040309.

[54] M. Cerezo et al. "Cost function dependent barren plateaus in shallow parametrized quantum circuits". In: *Nature Communications* 12.1 (Mar. 19, 2021), p. 1791. DOI: 10.1038/s41467-021-21728-w.

[55] Zoë Holmes et al. "Connecting Ansatz Expressibility to Gradient Magnitudes and Barren Plateaus". In: *PRX Quantum* 3.1 (Jan. 24, 2022), p. 010313. DOI: 10.1103/PRXQuantum.3.010313.

[56] Michael Ragone et al. *A Unified Theory of Barren Plateaus for Deep Parametrized Quantum Circuits*. Sept. 20, 2023. DOI: 10.48550/arXiv.2309.09342.

[57] M. Cerezo et al. *Does provable absence of barren plateaus imply classical simulability? Or, why we need to rethink variational quantum computing*. Dec. 14, 2023. DOI: 10.48550/arXiv.2312.09121.

[58] Guang-Bin Huang, Qin-Yu Zhu, and Chee-Kheong Siew. "Extreme learning machine: Theory and applications". In: *Neurocomputing* 70.1 (Dec. 1, 2006), pp. 489–501. DOI: 10.1016/j.neucom.2005.12.126.

[59] Herbert Jaeger and Harald Haas. "Harnessing Nonlinearity: Predicting Chaotic Systems and Saving Energy in Wireless Communication". In: *Science* 304.5667 (Apr. 2, 2004), pp. 78–80. DOI: 10.1126/science.1091277.

[60] David Verstraeten, Benjamin Schrauwen, and Dirk Stroobandt. "Reservoir Computing with Stochastic Bitstream Neurons". In: *Proceedings of the 16th annual Prorisc workshop* (2005), pp. 454–459.

[61] Kohei Nakajima and Ingo Fischer, eds. *Reservoir Computing: Theory, Physical Implementations, and Applications*. Singapore: Springer Singapore, 2021. DOI: 10.1007/978-981-13-1687-6.

[62] Gouhei Tanaka et al. "Recent advances in physical reservoir computing: A review". In: *Neural Networks* 115 (July 1, 2019), pp. 100–123. DOI: 10.1016/j.neunet.2019.03.005.

[63] Miguel C. Soriano et al. "Minimal approach to neuro-inspired information processing". In: *Frontiers in Computational Neuroscience* 9 (June 2, 2015). DOI: 10.3389/fncom.2015.00068.

[64] Jialing Li et al. "A deep learning based approach for analog hardware implementation of delayed feedback reservoir computing system". In: *2018 19th International Symposium on Quality Electronic Design (ISQED)*. Mar. 2018, pp. 308–313. DOI: 10.1109/ISQED.2018.8357305.

[65] Thomas Lymburn et al. "Reservoir computing with swarms". In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 31.3 (Mar. 8, 2021), p. 033121. DOI: 10.1063/5.0039745.

[66] Xiangzun Wang and Frank Cichos. "Harnessing synthetic active particles for physical reservoir computing". In: *Nature Communications* 15.1 (Jan. 29, 2024), p. 774. DOI: 10.1038/s41467-024-44856-5.

[67] François Duport et al. "All-optical reservoir computing". In: *Optics Express* 20.20 (Sept. 24, 2012), pp. 22783–22795. DOI: 10.1364/OE.20.022783.

[68] Mitsumasa Nakajima, Kenji Tanaka, and Toshikazu Hashimoto. "Scalable reservoir computing on coherent linear photonic processor". In: *Communications Physics* 4.1 (Feb. 10, 2021), pp. 1–12. DOI: 10.1038/s42005-021-00519-1.

[69] Keisuke Fujii and Kohei Nakajima. "Harnessing Disordered-Ensemble Quantum Dynamics for Machine Learning". In: *Physical Review Applied* 8.2 (Aug. 30, 2017), p. 024030. DOI: 10.1103/PhysRevApplied.8.024030.

[70] Yudai Suzuki et al. "Natural quantum reservoir computing for temporal information processing". In: *Scientific Reports* 12.1 (Jan. 25, 2022), p. 1353. DOI: 10.1038/s41598-022-05061-w.

[71] Joseph Bowles, Shahnawaz Ahmed, and Maria Schuld. *Better than classical? The subtle art of benchmarking quantum machine learning models*. Mar. 14, 2024. DOI: 10.48550/arXiv.2403.07059.

[72] C. K. Koç. "Analysis of sliding window techniques for exponentiation". In: *Computers & Mathematics with Applications* 30.10 (Nov. 1, 1995), pp. 17–24. DOI: 10.1016/0898-1221(95)00153-P.

[73] Robert Devaney. *An Introduction To Chaotic Dynamical Systems*. 2nd ed. Boca Raton: CRC Press, June 17, 2019. 360 pp. DOI: 10.4324/9780429502309.

[74] G. C. Layek. *An Introduction to Dynamical Systems and Chaos*. Singapore: Springer Nature, 2024. DOI: 10.1007/978-981-99-7695-9.

[75] Michael C. Mackey and Leon Glass. "Oscillation and Chaos in Physiological Control Systems". In: *Science* 197.4300 (July 15, 1977), pp. 287–289. DOI: 10.1126/science.267326.

[76] Edward N. Lorenz. "Deterministic Nonperiodic Flow". In: *Journal of the Atmospheric Sciences* 20.2 (Mar. 1, 1963), pp. 130–141. DOI: 10.1175/1520-0469(1963)020<0130:DNF>2.0.CO;2.

[77] O. E. Rössler. "An equation for continuous chaos". In: *Physics Letters A* 57.5 (July 12, 1976), pp. 397–398. DOI: 10.1016/0375-9601(76)90101-8.

[78] M. Hénon. "A Two-dimensional Mapping with a Strange Attractor". In: *The Theory of Chaotic Attractors*. Ed. by Brian R. Hunt et al. New York, NY: Springer, 2004, pp. 94–102. DOI: 10.1007/978-0-387-21830-4_8.

[79] C. Runge. "Ueber die numerische Auflösung von Differentialgleichungen". In: *Mathematische Annalen* 46.2 (June 1, 1895), pp. 167–178. DOI: 10.1007/BF01446807.

[80] W. Kutta. "Beitrag zur näherungsweisen Integration totaler Differentialgleichungen". In: *Zeit. Math. Phys.* 46 (1901), pp. 435–53.

[81] J. -P. Eckmann et al. "Liapunov exponents from time series". In: *Physical Review A* 34.6 (Dec. 1, 1986), pp. 4971–4979. DOI: 10.1103/PhysRevA.34.4971.

[82] Zachary C. Lipton, John Berkowitz, and Charles Elkan. *A Critical Review of Recurrent Neural Networks for Sequence Learning*. Oct. 17, 2015. DOI: 10.48550/arXiv.1506.00019.

[83] Kamilya Smagulova and Alex Pappachen James. "A survey on LSTM memristive neural network architectures and applications". In: *The European Physical Journal Special Topics* 228.10 (Oct. 1, 2019), pp. 2313–2324. DOI: 10.1140/epjst/e2019-900046-x.

[84] Andrea Mari et al. "Transfer learning in hybrid classical-quantum neural networks". In: *Quantum* 4 (Oct. 9, 2020), p. 340. DOI: 10.22331/q-2020-10-09-340.

[85] Ville Bergholm et al. *PennyLane: Automatic differentiation of hybrid quantum-classical computations*. July 29, 2022. DOI: 10.48550/arXiv.1811.04968.

[86] Aki Kutvonen, Keisuke Fujii, and Takahiro Sagawa. "Optimizing a quantum reservoir computer for time series prediction". In: *Scientific Reports* 10.1 (Sept. 7, 2020), p. 14687. DOI: 10.1038/s41598-020-71673-9.

[87] Pere Mujal et al. "Time-series quantum reservoir computing with weak and projective measurements". In: *npj Quantum Information* 9.1 (Feb. 23, 2023), pp. 1–10. DOI: 10.1038/s41534-023-00682-z.

[88] Samuel Tovey et al. *Generating Reservoir State Descriptions with Random Matrices*. arXiv.org. Apr. 10, 2024. URL: https://arxiv.org/abs/2404.07278v2 (visited on 08/14/2024).

[89] Jóakim v. Kistowski et al. "How to Build a Benchmark". In: *Proceedings of the 6th ACM/SPEC International Conference on Performance Engineering*. New York, NY, USA: Association for Computing Machinery, Jan. 31, 2015, pp. 333–336. DOI: 10.1145/2668930.2688819.

[90] Philip Sedgwick. "Pearson's correlation coefficient". In: *BMJ* 345 (July 4, 2012), e4483. DOI: 10.1136/bmj.e4483.

[91] Martin Larocca et al. "Theory of overparametrization in quantum neural networks". In: *Nature Computational Science* 3.6 (June 26, 2023), pp. 542–551. DOI: 10.1038/s43588-023-00467-6.

[92] Thomas Hubregtsen et al. "Evaluation of parameterized quantum circuits: on the relation between classification accuracy, expressibility, and entangling capability". In: *Quantum Machine Intelligence* 3.1 (Mar. 11, 2021), p. 9. DOI: 10.1007/s42484-021-00038-w.

[93] Pasquale Calabrese and John Cardy. "Entanglement entropy and quantum field theory". In: *Journal of Statistical Mechanics: Theory and Experiment* 2004.6 (June 2004), P06002. DOI: 10.1088/1742-5468/2004/06/P06002.

[94] Carl W. Helstrom. "Quantum detection and estimation theory". In: *Journal of Statistical Physics* 1.2 (June 1, 1969), pp. 231–252. DOI: 10.1007/BF01007479.

[95] Johannes Herrmann et al. "Realizing quantum convolutional neural networks on a superconducting quantum processor to recognize quantum phases". In: *Nature Communications* 13.1 (July 16, 2022), p. 4144. DOI: 10.1038/s41467-022-31679-5.

[96] Gerasimos Angelatos, Saeed A. Khan, and Hakan E. Türeci. "Reservoir Computing Approach to Quantum State Measurement". In: *Physical Review X* 11.4 (Dec. 29, 2021), p. 041062. DOI: 10.1103/PhysRevX.11.041062.

[97] Quoc Hoan Tran and Kohei Nakajima. "Learning Temporal Quantum Tomography". In: *Physical Review Letters* 127.26 (Dec. 22, 2021), p. 260401. DOI: 10.1103/PhysRevLett.127.260401.