Quantum Machine Intelligence (2023) 5:45
https://doi.org/10.1007/s42484-023-00133-0

RESEARCH ARTICLE

Time series quantum classifiers with amplitude embedding

M. P. Cuéllar[1] · C. Cano[1] · L. G. B. Ruiz[2] · L. Servadei[3]

Received: 30 June 2023 / Accepted: 24 October 2023 / Published online: 27 November 2023

Abstract
Quantum Machine Learning was born during the past decade as the intersection of Quantum Computing and Machine Learning. Today, advances in quantum computer hardware and the design of simulation frameworks able to run quantum algorithms in classic computers make it possible to extend classic artificial intelligence models to a quantum environment. Despite these achievements, several questions regarding the whole quantum machine learning pipeline remain unanswered, for instance the problem of classical data representation on quantum hardware, or the methodologies for designing and evaluating quantum models for common learning tasks such as classification, function approximation, clustering, etc. These problems become even more difficult to solve in the case of Time Series processing, where the context of past historical data may influence the behavior of the decision-making model. In this piece of research, we address the problem of Time Series classification using quantum models, and propose an efficient and compact representation of time series in quantum data using amplitude embedding. The proposal is capable of representing a time series of length n in $log_2(n)$ computational units, and experiments conducted on benchmark time series classification problems show that quantum models designed for classification can also outperform the accuracy of classic methods.

Keywords Quantum machine learning · Time series classification · Amplitude embedding · Quantum neural networks

1 Introduction

A time series $X(t) = \{x(1), x(2), ..., x(t)\}$ is a sequence of observations of a given phenomenon sampled periodically and indexed in time. If the time series is not infinite, we write a time series of length T as $X(T)$. Time Series analysis tools are very common in many scientific fields and solve different types of problems such as forecasting (Liu et al. 2021), clustering (Warren Liao 2005), classification (Fawaz et al. 2018), or anomaly detection (Schmidl et al. 2022), to mention just a few. In the case of a classification problem, a dataset containing pairs of time series and labels $\{X_i(T_i), y_i\}_{i=1}^N$ is provided, where the labels $y_i \in Y = \{y_k | 1 \leq k \leq K\}$ with K the number of classes. The objective is to find a model f usually parameterized by a set of parameters θ so that the model output is the label of a time series given as input, i.e., $y_i = f(X_i(T_i), \theta)$, using a supervised learning approach. Examples of well-known time series classifiers are k-Nearest Neighbors (k-NN) with either Euclidean or Dynamic Time Warping (DTW) distance measurements (Kate 2015), neural networks and deep learning (Fawaz et al. 2018), the BOSS algorithm (Schäfer 2015), shapelets (Bagnall et al. 2015), support vector machines (Cuturi 2011), time series forests (Deng et al. 2013), and so on. Depending on the method, we can distinguish two types of time series processing: one analyzes time series as a sequence of values (e.g., recurrent neural networks or DTW), and the other type

C. Cano, L. G. B. Ruiz, and L. Servadei contributed equally to this work

✉ M. P. Cuéllar
manupc@decsai.ugr.es

C. Cano
ccano@decsai.ugr.es

L. G. B. Ruiz
bacaruiz@ugr.es

L. Servadei
lorenzo.servadei@tum.de

[1] Department of Computer Science and Artificial Intelligence, University of Granada, ETSIIT. C/. Pdta. Daniel Saucedo Aranda s.n., Granada 18014, Spain

[2] Department of Software and Computer Engineering, University of Granada, ETSIIT. C/. Pdta. Daniel Saucedo Aranda s.n., Granada 18014, Spain

[3] School of Computation, Information and Technology, Technical University of Munich, Hans-Piloty-Straße 1, Garching bei Munchen, Munich 85748, Germany

removes the time component and analyzes the data as a timeless pattern (e.g., time series forests, Euclidean distance, or feedforward SVM/neural networks). In this paper we study time series classification problems using quantum computing algorithms. The proposal described in this manuscript fits in the second category, since loops are not yet allowed in a quantum algorithm, although recently intensive research efforts are being conducted regarding, for example, Recurrent Quantum Neural Networks (Bausch 2020).

Quantum Computing (QC) was introduced by Richard Feynman in 1982 (Feynman 1982) after observing the complexity of simulating a quantum system with a classic computer. The foundations of QC come from the area of physics (quantum mechanics) and its mathematical modeling using linear algebra over Hilbert spaces. QC natively includes computational tools that are not present in classical computing, such as superposition, entanglement, quantum parallelism, or tunneling. These tools have been the cornerstone for the design of algorithms capable of solving certain problems more efficiently in a quantum computer than in a classic one, for example, the search over unordered sets with the Grover's algorithm (Grover 1997), the factorization of integers with Shor's method (Shor 1999), or the identification of constant or balanced functions with the Deutsch-Jozsa proposal (Deutsch and Jozsa 1992). On the other hand, Quantum Machine Learning (QML) (Wittek 2014) has been studied theoretically during the last two decades, and it attempts to design and implement supervised, unsupervised, and reinforcement learning methods on quantum computers. The advances in the development of quantum computers and QC simulators and the emergence of a plethora of software libraries for writing, optimizing, and executing quantum circuits and algorithms such as Google's Cirq, IBM's Qiskit, or Xanadu's Pennylane, among others, have allowed researchers to develop and adapt the ideas of classical Machine Learning to the quantum level. Essential tools in QML are the Variational Quantum Eigensolver (Peruzzo et al. 2014) to find the eigenvalues of a matrix efficiently, the methods of quantum unconstrained optimization with algorithm QAOA (Farhi et al. 2014), or methods to compute gradients on quantum computer hardware such as the parameter-shift rule (Crooks 2019). Examples of QML methods range from regression problems (Wang 2017) to classification (Schuld et al. 2014), clustering (Aïmeur et al. 2007), and reinforcement learning (Dong et al. 2008). A detailed review of QML methods is available in the survey (Umer and Sharif 2022).

With respect to the classification problem, the early work (Schuld et al. 2014) proposed a quantum algorithm analogous to the k-NN method, applied to binarized data on the MNIST digits dataset. Currently, most quantum classifiers rely on hybrid classic-quantum approaches using variational quantum circuits (VQC) trained with classical optimization algorithms. Examples, in this case, are the work by Bohhan et al. (Bokhan et al. 2022), which performs multi-class classification in the MNIST dataset using Quantum Neural Networks to predict 4 classes, and the article by Maheshwari et al. (2022) which addresses binary classification problems using amplitude encoding. If we focus on time series processing algorithms with quantum computers, we find that the literature is scarce and the problem has not been extensively addressed. Time Series forecasting has been studied in Rivera-Ruiz et al. (2022) using VQCs as predictive models, and also in Emmanoulopoulos and Dimoska (2022) for real financial datasets. In the case of classification problems, we have found no pure time series quantum classification approaches, although a time series clustering algorithm that it is later adapted for classification is proposed in Yarkoni et al. (2021).

In this work, we explore how Time Series classification can be performed in a quantum computing environment using tools from the QML paradigm. The challenges of the approach are varied and range from the problem of time series representation to the design and optimization of the classifier. Regarding the representation problem, contemporary quantum computers do not allow loops in a quantum algorithm, so encoding time series as sequences of values can be difficult (Sutor 2019). However, there is a set of tools for encoding classical information in quantum data, also known as quantum embedding (Ganguly 2021), as basis encoding to represent binary data, amplitude embedding to encode data as amplitudes of a quantum state, tensor product encoding to map a classic scalar data to a qubit, or hamiltonian encoding to embed data into the evolution of a quantum system. Tensor product encoding is the most widely used embedding technique, as it is an efficient and natural way of feeding information to variational quantum circuits. However, representing a time series with tensor product encoding would require a large number of available qubits (as many as the length of the time series). For this reason, our design studies amplitude encoding as an alternative to effectively embed a time series of length T into $log_2(T)$ qubits, so that we achieve an exponential gain in space/memory and necessary computational units to represent a time series with respect to a classical computing paradigm. For the classifier, we use VQCs since they have been widely studied for both classification and regression problems in recent years with promising results (Wang 2017; Rivera-Ruiz et al. 2022; Emmanoulopoulos and Dimoska 2022). Typically, the output of a VQC for regression or classification is measured using quantum observables such as Pauli-Z operators on a qubit or set of qubits (Andrés et al. 2022). However, in this work, we study a different and almost unexplored strategy which consists of measuring probability amplitudes of the quantum state. This strategy has the advantage that, for a system containing n qubits, it can be modeled with a maxi-

mum number of 2^n possible outputs. The model is validated on benchmark time series classification datasets containing binary and multi-class problems. The results obtained suggest that the approach is capable of achieving higher accuracy than methods of the existing literature so that quantum algorithms for Time Series classification can offer advantages in both compact time series representation and performance.

The remaining of the manuscript is structured as follows: Sect. 2 introduces the fundamental concepts of quantum computing and quantum machine learning to make the article self-complete. After that, Sect. 3 describes the approach. Section 4 then performs the experimentation using benchmark data, and Sect. 5 concludes.

2 Fundamentals of quantum computing and quantum machine learning

This section explains the fundamental concepts of quantum computing and quantum machine learning necessary for the proposal, to achieve the goals of article self-completeness and describe the notation used. The basic references supporting of the text in this section are Sutor (2019) for QC and Ganguly (2021) for QML.

2.1 Quantum computing and quantum algorithms

Classical computing is substantiated over a binary computing unit or *bit*, whose values belong to the group with two elements $\mathbb{Z}_2$. Operations on systems containing n bits are performed over the space built by the cartesian product of the group, i.e., $\times^n \mathbb{Z}_2 = \mathbb{Z}_2^n$, and each time a new bit is included into the system, the dimension of the resulting space increases by one. On the other hand, the fundamental computing unit in quantum computing is the *qubit*, whose values are in the vector space $\mathbb{C}^2$ with an orthonormal basis denoted by the column vectors $|0\rangle = (1, 0)^t$ and $|1\rangle = (0, 1)^t$, also known as the *computational basis*. As an element of the vector space, the value of an arbitrary qubit $|\psi\rangle$ can be expressed as a linear combination of the basis vectors, i.e., $|\psi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle$, where $\alpha_0, \alpha_1 \in \mathbb{C}$ are called *amplitudes* and hold the additional constraint that $|\alpha_0|^2 + |\alpha_1|^2 = 1$. Unlike classical computing, where a bit can be set to a value in $\{0, 1\}$, a qubit can potentially hold an infinite number of values. Furthermore, the mathematical model for a system containing n qubits is the tensor product of the complex plane, i.e., $\bigotimes^n \mathbb{C}^2 = \mathbb{C}^{2^n}$ which means that every time a new qubit is included in the system, the resulting space doubles its size. For example, the mathematical model that supports a system with $n = 2$ qubits is $\mathbb{C}^{2^2}$, and its computational basis is obtained as $\{|0\rangle \otimes |0\rangle = |00\rangle, |0\rangle \otimes |1\rangle = |01\rangle, |1\rangle \otimes |0\rangle = |10\rangle, |1\rangle \otimes |1\rangle = |11\rangle\}$. A quantum state $|\psi\rangle$ in this system can be expressed as $|\psi\rangle = \sum_{i=0}^{2^n-1} \alpha_i |i\rangle$, where $|i\rangle$ stands for the i-th basis vector in binary representation, with the constraint $\sum_{i=0}^{2^n-1} |\alpha_i|^2 = 1$.

One of the limitations of quantum computing is that it is not possible to know the exact value of an arbitrary quantum state $|\psi\rangle$ at a given time. The *measurement* operation on the quantum state causes the state to collapse to any basis state $|i\rangle$ with probability $|\alpha_i|^2$, which is the system output. Except for measurement, all quantum operations must be reversible and can be modeled as a unitary matrix U that multiplies the state $|\psi\rangle$, causing a state transition to $|\psi'\rangle$, i.e., $|\psi'\rangle = U|\psi\rangle$. Examples of quantum operations are the rotation X/Y/Z gates parameterized by θ ($R_x(\theta)$, $R_y(\theta)$, $R_z(\theta)$) on a single qubit, or the Controlled-NOT (CNOT gate) applied to two qubits (Eq. 1).

$$R_x(\theta) = \begin{pmatrix} cos(\frac{\theta}{2}) & -isin(\frac{\theta}{2}) \\ -isin(\frac{\theta}{2}) & cos(\frac{\theta}{2}) \end{pmatrix} \quad R_y(\theta) = \begin{pmatrix} cos(\frac{\theta}{2}) & -sin(\frac{\theta}{2}) \\ sin(\frac{\theta}{2}) & cos(\frac{\theta}{2}) \end{pmatrix}$$
$$R_z(\theta) = \begin{pmatrix} e^{-i\theta/2} & 0 \\ 0 & e^{i\theta/2} \end{pmatrix} \qquad CNOT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{1}$$

A quantum algorithm is a sequence of quantum operations applied on an initially known quantum state (usually $|0\rangle$). Quantum algorithms are implemented in quantum circuits, whose graphical representation assigns a horizontal line for each qubit in the system, and the gates are graphical symbols located on a line. If a gate involves an operation between more than one qubit, a vertical line is used to connect the source/target qubits of the gate. Figure 1 shows an example of a cicuit with two qubits q_0, q_1 that implements the algorithm $CNOT((H|q_0\rangle) \otimes |q_1\rangle)$. This circuit is a basic quantum entanglement over two qubits q_0, q_1, which first moves qubit q_0 from $|0\rangle$ to standard superposition using the Hadamard (H) gate, and then applies a CNOT gate on q_1 controlled by q_0. Finally, both qubits are measured in classical bits c_0, c_1, respectively. The resulting quantum state of the circuit before measurement is $|\psi\rangle = \frac{\sqrt{2}}{2}|00\rangle + \frac{\sqrt{2}}{2}|11\rangle$, so the measurement will provide the basis states $|00\rangle$ or $|11\rangle$ with probability of 0.5.

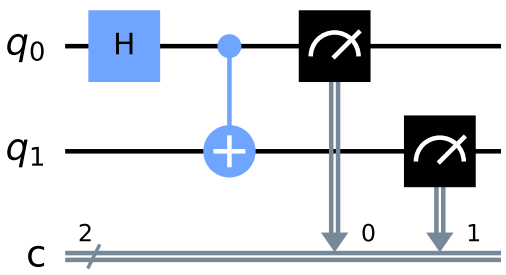

Fig. 1 Quantum circuit example containing 2 qubits q_0, q_1, and 2 bits for measurement in line c

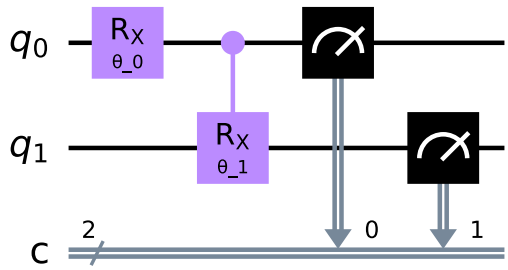

Fig. 2 Example of variational quantum circuit with parameterized gates $R_x(\theta_0)$ over qubit q_0 and $cR_x(\theta_1)$ over q_1 controlled by q_0

A variational quantum circuit (VQC) is a quantum circuit containing parameterized gates. The parameters for these gates are typically provided to the computer at runtime, and can help tune a template circuit to fit different behaviors. Figure 2 shows an example of a parameterized quantum circuit with a X-rotation gate R_x whose parameter is θ_0, and a controlled X-rotation gate cR_x with parameter θ_1. The implemented algorithm is $cRx_{\theta_1}((Rx_{\theta_0}|q_0\rangle) \otimes |q_1\rangle)$ and, from Eq. 1, it is easy to verify that the circuit of Fig. 2 is equivalent to the circuit of Fig. 1 when $\theta_0 = \pi/2$ and $\theta_1 = \pi$, or also that the circuit returns the quantum state $|11\rangle$ when $\theta_0 = \theta_1 = \pi$. VQCs play a very important role in Quantum Machine Learning since they can be used to embed classical information in a quantum state, and also to control the behavior of classification/regression models (Wittek 2014; Wang 2017; Bokhan et al. 2022; Andrés et al. 2022) such as Quantum Support Vector Machines or Quantum Neural Networks.

2.2 Quantum machine learning

Quantum Machine Learning sits at the intersection of Quantum Computing and Machine Learning, and attempts to apply ML techniques on quantum computers. There are three main ways to approach this general goal (Pushpak and Jain 2021):

- Quantum Algorithms for Quantum Data (Q-Q): It assumes quantum datasets that are processed by quantum algorithms on quantum hardware.
- Classic Algorithms for Quantum Data (C-Q): It uses classical machine learning models running on classical computers to process quantum information. An example in this category is the analysis of quantum states, or quantum tomography, with traditional machine learning.
- Quantum Algorithms for Classic Data (Q-C): A traditional classic dataset is processed by a quantum algorithm running on quantum hardware.

Since most contemporary data analysis problems use classical data, the Q-C strategy is the most studied nowadays in the research areas of Machine Learning and Artificial Intelligence. In this way, QML proposals in the literature have focused on adapting known ML algorithms to quantum computers, or on constructing a QML algorithm analogous to a classical ML method (Umer and Sharif 2022).

The usual pipeline for a QML solution is a hybrid approach that involves classical ML methods (running on Central Processing Units, CPUs) and QML algorithms (running on Quantum Processing Units, QPUs), and encompasses the following steps (Fig. 3):

1. **Dataset acquisition and preprocessing**: It is carried out on CPU, and contains the usual initial steps in ML, where data are acquired from sources and preprocessing is performed (handling missing values, data transformations/normalizations, etc.).
2. **Quantum Information Processing**: This step is executed in QPU, and it takes preprocessed classical data as input and performs the following tasks:
   (a) **Quantum Embedding**: Classical data are encoded into a quantum state using a quantum embedding technique, as basis encoding, tensor encoding, amplitude encoding, etc (Ganguly 2021). This step typically involves the design of a variational quantum algorithm to evolve a known initialized quantum state such as $|0\rangle$ to the desired quantum state that encodes the classical data.
   (b) **Antsatz**: It implements the quantum algorithm for the QML task, such as regression, classification, and clustering.
   (c) **Measurement**: The resulting quantum state from the QML algorithm is measured using a measurement strategy, and the results are fed back to the classical computer for the next stage.

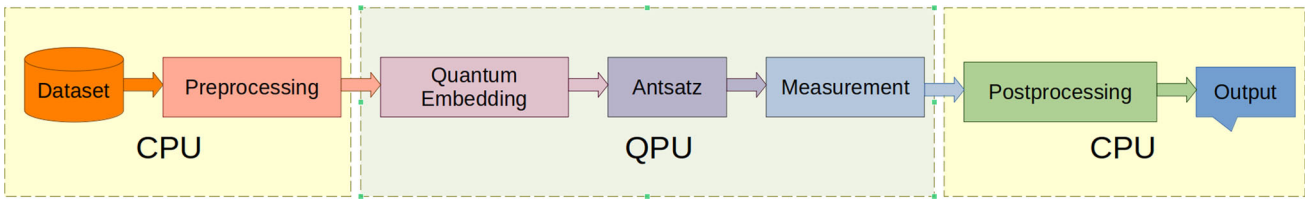

Fig. 3 Pipeline of Quantum Machine Learning processing of classic data

3. **Postprocessing and results**: The outputs of the Quantum Information Processing stage are postprocessed (if necessary) in the CPU, and the final results are provided.

In this work, we address the problem of Time Series Classification, where the values of the time series are classical numerical data. For this reason, we adopt the Q-C strategy and follow the methodology described in Fig. 3. Regarding the Quantum Information Processing stage, our proposal requires designing the mechanisms to encode a Time Series in a quantum state, the antsatz for the quantum classifier, and a strategy to measure quantum states.

3 Time series quantum encoding and classification

We adopt the common strategy in the literature in which a Variational Quantum Circuit interpreted as a feedforward Quantum Neural Network is used as a decision maker to solve the machine learning problem at hand (Macaluso et al. 2020; Andrés et al. 2022; Skolik et al. 2022). In particular, we extend this scheme and create a hybrid classical /quantum neural network where the quantum processing uses amplitude embedding of pre-processed time series. The main scheme of the proposal is shown in Fig. 4, and contains the following components:

- **Time Series preprocessing and resampling**. In this step, traditional preprocessing is performed on the time series data acquired on classical computers, including resampling, differentiation, scaling, etc. The resulting time series dataset is fed into the hybrid classical /quantum neural network.
- **Hybrid classic/quantum neural network processing**. The neural network is organized into three main blocks:
  - **Classic pre-quantum processing**. This step aims to perform linear/non-linear transformations of the time series, either with injections into higher dimensional spaces or projections to lower dimensional spaces, with the aim of preparing the time series and extracting relevant features for a better performance of the quantum classifier. It can contain none, one, or several classical fully-connected layers with activation functions typically used in Deep Learning.
  - **Quantum processing**. This stage starts with an encoding layer that takes as input the pre-processed time series of the pre-quantum classical layers and provides a representation of the quantum state of the time series using the amplitude embedding strategy. After that, one or more quantum layers containing Variational Quantum Circuits are executed. Finally, the measurement is performed to return the classification results. In the following sections, the layers of the quantum processing step are called the Variational Quantum Classifier.
  - **Classic post-quantum processing**. This last module also runs on a classic computer. It takes the outputs of the Variational Quantum Classifier and performs aggregations of the measurements to return the target label of the input time series.

In the following subsections, we describe in detail the components of the proposed hybrid classical/quantum neural network model separately, with a special focus on quantum and post-quantum processing.

In the following subsections, we describe in detail the components of the hybrid classic/quantum neural network model proposed separately, with a special focus on quantum and post-quantum processing.

3.1 Time series quantum amplitude embedding

A quantum state $|\psi\rangle$ containing n qubits is modeled as the complex linear combination of the 2^n vectors of the computational basis, i.e., $|\psi\rangle = \sum_{i=0}^{2^n-1} \alpha_i |i\rangle, \alpha_i \in \mathbb{C}$. Amplitude

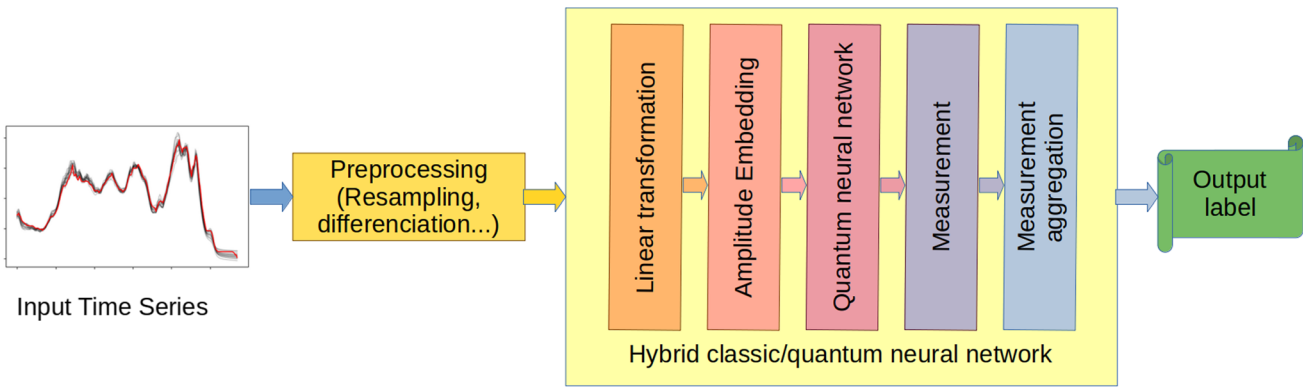

Fig. 4 Example of variational quantum circuit with parameterized gates $R_x(\theta_0)$ over qubit q_0 and $cR_x(\theta_1)$ over q_1 controlled by q_0

embedding encompasses a set of techniques for encoding multidimensional classical data into the amplitudes of the quantum state $\{\alpha_i\}$. Since all probability amplitudes $|\alpha_i|^2$ must sum 1, amplitude embedding methods are restricted to classical data with unitary L2-norm. If this is not the case for a classical input data pattern $v = (v_1, v_2, ..., v_{2^n})$, then the preprocessing described in the Eq. 2 is required to ensure the unitary norm, where $||v||$ stands for the L2-norm of v.

$$v_i \leftarrow \frac{v_i}{||v||} \tag{2}$$

Furthermore, if we want to use all the available amplitudes to represent the information, the classical input data must have dimension $T = 2^n$. In the case of $T < 2^n$, classical data can be injected into the 2^n dimensional space by means of zero-filling the remaining components from $T + 1$ to 2^n. On the other hand, if $T > 2^n$ then the data must be projected to a space of dimension 2^n or lower. In this work we assume a Quantum Variational Classifier with n qubits. In our experiments, the Time Series are resampled to a length of $T = 2^n$ using any user-defined resampler (linear, polynomial, etc.) and fed to the quantum embedding layer in the case that the pre-quantum processing module contains no layers. Otherwise, the time series is fed to the classical pre-quantum module whose output dimension must be 2^n.

Amplitude encoding can be achieved, in the case of classical multidimensional data $v \in \mathbb{R}^{2^n}$, using a Divide-and-Conquer procedure described in Araujo et al. (2021). This is performed by means of a sequence of Y-rotation gates $R_y(\theta)$ and (multiple) controlled Y-rotation gates $cR_y(\theta)$ (see Eq. 1). The idea behind the method is to redistribute the probabilities that the k-th qubit is $|0\rangle$ or $|1\rangle$ conditioned to all previous qubits $q_0, ...q_{k-1}$. Figure 5 shows a simple example for encoding an 8-dimensional data with the unitary norm. The goal of the example is to show the decomposition and distribution of amplitude probabilities for embedding the array $v = (0, 0.5, \sqrt{0.15}, 0, \sqrt{0.125}, \sqrt{0.125}, 0.5, \sqrt{0.1})$ to the quantum state $|\psi\rangle = 0|000\rangle + 0.5|001\rangle + \sqrt{0.15}|010\rangle + 0|011\rangle + \sqrt{0.125}|100\rangle + \sqrt{0.125}|101\rangle + 0.5|110\rangle + \sqrt{0.1}|111\rangle$ composed of three qubits $|q_0q_1q_2\rangle$ initialized to $|000\rangle$. To do so, we first calculate the probability of $p(q_0 = |1\rangle) = 0.6$ (root node) as the sum of the squared absolute amplitudes of basis vectors $|100\rangle, |101\rangle, |110\rangle, |111\rangle$, i.e., assigning the qubit $q_0 = |1\rangle$ and leaving the qubits q_1, q_2 free. The angle required for a Y-rotation gate to create the quantum state $\sqrt{0.6}|100\rangle + \sqrt{0.4}|000\rangle$ is calculated as $\theta = 2asin(\sqrt{p(q_0 = |1\rangle)})$. After that, we recursively decompose each amplitude with controlled rotation gates cRy_t^c where t stands for the target qubit and c for the set of control qubits. For example, in the case of the decomposition of $|100\rangle$ for the second qubit q_1, we first calculate the probability of $q_1 = |1\rangle$ constrained to $q_0 = |1\rangle$, i.e., $p(q_1 = |1\rangle|q_0 = |1\rangle) = 0.35/0.6$, and perform the conditional rotation of q_1 subject to $q_0 = |1\rangle$, cRy_1^0. The procedure is applied recursively for each node in the tree until we establish all probability amplitudes. The design of a quantum circuit that implements these operations is simple, for example, by means of executing operations as they appear in a pre-order tree traversal. Figure 6 shows the resulting circuit of Fig. 5 using this procedure. Gates X are used to switch qubits from $|0\rangle$ to $|1\rangle$ and vice versa to set conditions of control qubits.

In Fig. 6, the Ry operation in the first block corresponds to the rotation performed on the root node in Fig. 5. Then, the cRy of the second block comes from the next node visited in the preorder traversal with the first left-child criterion. The third block contains a multiple-controlled Ry gate that corresponds to the operation of node $\sqrt{0.35}|110\rangle$, and adjusts the probability of q_2 subject to the first qubits $q_0q_1 = |11\rangle$. After that, amplitudes of the basis vectors $|111\rangle, |110\rangle$ are set. The fourth block contains the operation of the node $0.5|100\rangle$ to establish the amplitudes of basis vectors $|101\rangle, |100\rangle$. Since it is partially conditioned on $q_1 = |0\rangle$, negating q_1 with a X-gate is required for the cRy gate. After performing the conditional rotation, q_1 returns to its previous value. The

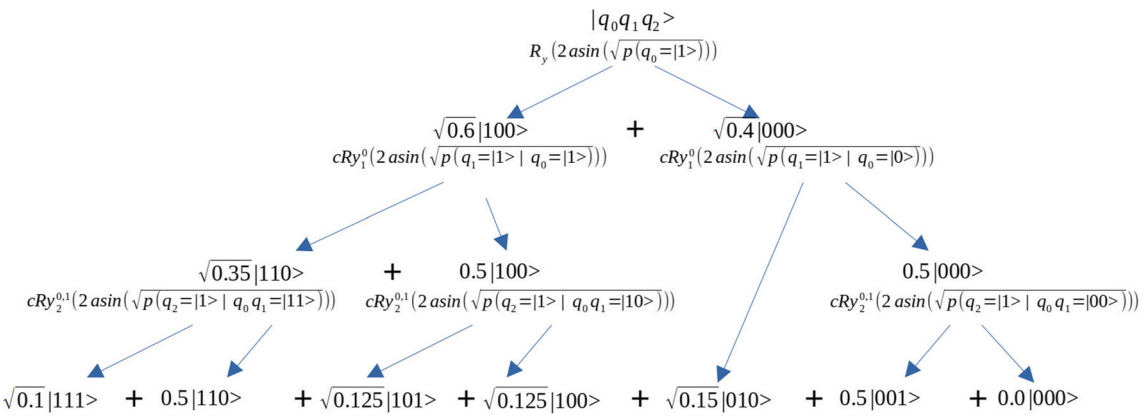

Fig. 5 Example of amplitude embedding and probability amplitudes decomposition for the data $v = (0, 0.5, \sqrt{0.15}, 0, \sqrt{0.125}, \sqrt{0.125}, 0.5, \sqrt{0.1})$

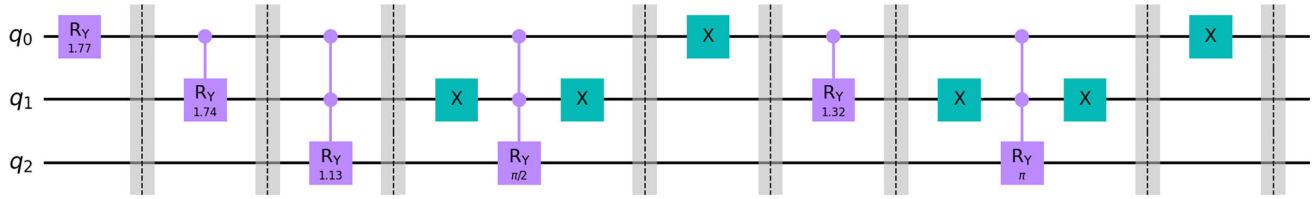

Fig. 6 Circuit obtained by preorder traversal of tree in Fig. 5

fifth block contains a single X-rotation gate, whose goal is to negate q_0 and to allow to visit the right branch of the root node. Then, a conditional rotation is performed on the node $\sqrt{0.4}|000\rangle$ in the sixth block of the circuit to obtain the target amplitudes $\sqrt{0.5}|010\rangle$ and $0|011\rangle$, and the seventh block performs a similar operation to the fourth block to set amplitudes in node $0.5|000\rangle$. The last block returns q_0 to its initial value, and finishes the amplitude embedding of the data.

As it can be verified, the height of a tree designed to create amplitude embedding is equal to the number of qubits n. This implies that a classical data of dimension T can be encoded into $log_2(T)$ qubits, but also that the number of operations to create the encoded quantum state grows exponentially with the number of qubits needed in the worst case. Therefore, there is a trade off between space and time complexity that must be studied for each problem whose data can be encoded using this technique.

In this work, we encode classical time series data into a quantum state using amplitude embedding. To benefit from all the computational power that can be encoded into n qubits, our proposal is constrained to time series of length $T = 2^n$ with unitary norm. As mentioned above, this can be achieved by time series resampling and preprocessing on a classical CPU with Eq. 2. Once these conditions are met, the time series $X(T) = \{x(0), x(1), ..., x(2^n - 1)\}, x(i) \in \mathbb{R}$, is encoded with the aforementioned procedure in the quantum state $|\psi\rangle = x(0)|0...0\rangle + x(1)|0...01\rangle + ... + x(2^n - 1)|1...1\rangle$. This quantum state is the input for the quantum classifier layers.

3.2 Variational quantum classifier

Unlike classical Neural Network design, there are not many standardized layer structures for building a Quantum Neural Network, such as fully connected linear layers or activation functions. However, there seems to be a consensus in the literature (Wang 2017; Bokhan et al. 2022; Andrés et al. 2022; Skolik et al. 2022; Rajesh et al. 2021) that a basic quantum layer should contain parameterized gates to alter the value of qubits (typically single-qubit rotation gates) followed by a mechanism of information transfer among qubits, generally implemented as entanglements organized by a given structure. In this work, a layer of a quantum neural network is implemented as a sequence of rotations R_x, R_y, R_z for each qubit, and entanglement with CNOT gates organized in a ring structure. This antsatz structure was selected after a previous trial-and-error experimentation to adjust the number and type of rotation gates, and the CNOT information transfer mechanism. The inclusion of rotation gates on all three axes is also justified to allow for a general antsatz that does not constrain the desired observable to be used for measurement. Figure 7 shows the structure of a layer for a circuit containing 4 qubits, where θ_{ij} is the free parameter to be optimized for the rotation gate j of the i-th qubit. A layer of a Quantum Neural Network with n qubits contains a fixed value of parameters equal to $3n$.

In this work, the **Quantum Processing** step described in Sect. 3 is composed of an encoding layer containing the quantum embedding procedure described in Subsection 3.1 followed by one or several layers (hyperparameter) containing the antsatz described here for a given number of qubits (also a hyperparameter). The output of the last layer is measured to obtain the results of the Variational Quantum Classifier.

3.3 Measurement and aggregation

The usual way in the existing literature to obtain the results of a Quantum Neural Network is through an observable applied to one or more qubits, for example the Pauli-Z σ_z observable (Ganguly 2021) on the computational basis. Formally, the mathematical modeling of the expectation of the value of an observable O over a quantum state $|\psi\rangle$ is $\langle\psi|O|\psi\rangle$ where $\langle\psi|$ stands for the complex conjugate transpose of $|\psi\rangle$, and O for

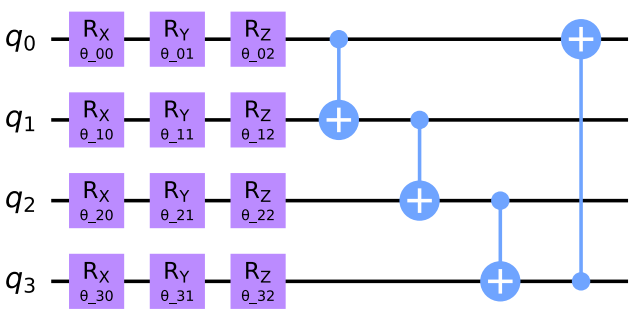

Fig. 7 Antsatz of the proposed quantum neural network layer

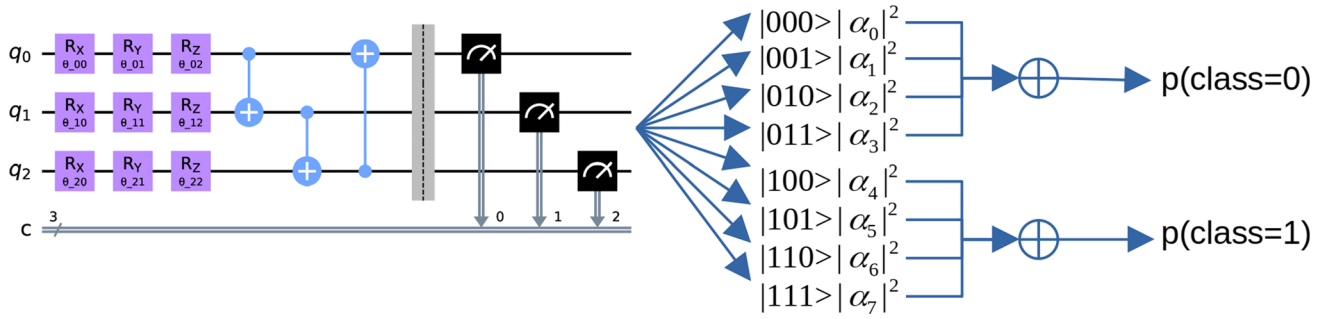

Fig. 8 Example of measurement aggregation with non-parameterized post-quantum processing for binary classification

the hermitian matrix of the observable operator. In this work, we select a slightly different approach in which we measure the probability of each basis vector of the system, i.e., an approximation of the values $\{|\alpha_i|^2\}$ for all basis vectors $|i\rangle$. In a quantum computer simulation software, this is easily performed with $O(1)$ complexity since the true quantum state value is known; however, this measurement in a real scenario with quantum hardware would require to run the quantum processing step multiple times, or replicating the circuit in different quantum registers running in parallel, to obtain the approximation of the probability amplitudes. The method has the advantage that it is possible to have a maximum number of 2^n outcomes, which means that the system can probide a maximum of 2^n different class labels for a quantum classifier with n qubits.

For the case where the number of class labels $C < 2^n$ (the usual situation), we propose to aggregate the measured values to take into consideration the information encoded by all entangled qubits. This aggregation is performed on the CPU (step **classic post-quantum processing** described in Sect. 3), and can be parameterized (e.g., a fully-connected linear layer followed by a softmax activation function), or not parameterized. In this work, we explore this second case and we propose to aggregate by addition values of $|\alpha_i|^2$ to provide the final probability of a class label prediction. Figure 8 shows an example of this scheme with the measurement on the variational classifier with 3 qubits in the last layer assuming a binary classification problem. The aggregation in the Figure is performed on consecutive qubits, and the scheme returns the classification probability for two classes $\{0, 1\}$. In the experiments, we name this module as the *aggregation* post-quantum processing step.

3.4 Training scheme

The proposed model is a hybrid classical/quantum neural network, so parameter optimization is performed on the CPU. In this section we consider the general case in which all pre-quantum, quantum, and post-quantum modules contain a set of parameters $\theta^1, \theta^2, \theta^3$, respectively. Figure 9 describes the learning procedure: First, the time series data are fed into the network to provide the probability of the class labels in a forward pass. After that, a loss function (binary cross-entropy for binary classifiers, cross-entropy for multiclass problems, etc.) is calculated and gradients are backpropagated through the layer structure. Finally, an optimizer (gradient descent, back-propagation, Adam, etc.) updates the parameters.

4 Experiments

Experiments have been conducted on multi-class and binary time series classification benchmark datasets to test the proposal. The datasets used are available on the website of the UEA & UCR Time Series Classification Repository (Dau et al. 2019) https://www.timeseriesclassification.com. Quantum neural networks do not allow loops in the internal structure and, for that reason, we compare the results with classical feedforward neural networks as the closest network structure we have found in classical neural networks. Since we performed the experiments in quantum computer simulation software, we constrained the experimentation to problems with a reduced number of classes (up to 7) since the classical operations required for the simulation of quantum

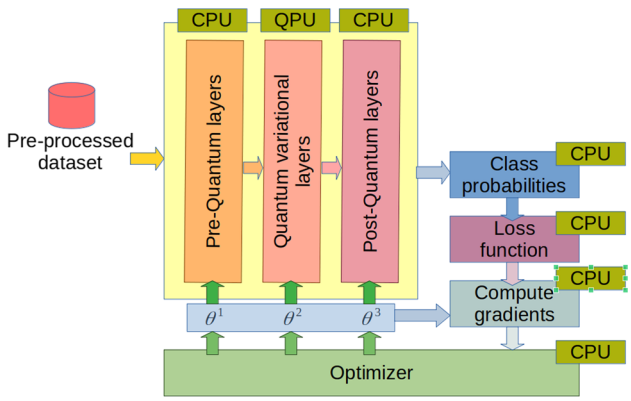

Fig. 9 Training scheme for the hybrid classic/quantum neural network

algorithms grow exponentially with the number of qubits, and linearly with the number of input/output data patterns.

4.1 Dataset description

The Table 1 summarizes the datasets used in our experimentation, and it contains the name of the dataset (column 1), the number of classes in the problem (column 2), the number of training and test samples (columns 3 and 4), and the length of the time series (column 5). Some datasets also contain unbalanced data, such as *Earthquakes* with 264 patterns for class 0 and 58 patterns for class 1; *OliveOil* with 13 patterns for class 3, 8 patterns for class 1, 5 patterns for class 0 and 4 patterns for class 2; and *DistalPhalanxTW* with 215 patterns for class 5, 82 patterns for class 2, 30 patterns for class 3, 29 patterns for class 0, 28 patterns for class 1, and 18 patterns for class 4. In these cases, we treat unbalanced data with the *class weighting* strategy to give the same importance to all classes during the learning of both classical Artificial Neural Networks and the hybrid classic/quantum proposal. The main criteria to select the datasets were that the UEA & UCR Time Series Classification Repository contains a reference to the best solution reported in the literature for comparison, and also that the size of the dataset allows an experimentation with different number of class labels in a desktop computer in affordable time. A brief description of each dataset is provided below to show the nature of the time series data.

The *Coffee* dataset contains spectrographs of two variants of coffee beans (Arabica and Robusta), and the goal is to distinguish between these two variants using its spectrograph. On the other hand, *Earthquakes* contains seismic measurements from the Northern California Earthquake Data Center averaged hourly from 1967 to 2003, and the objective is to predict whether or not an earthquake with value 5 or higher in the Ritcher scale will take place or not. The *Car* dataset aims to classify four different types of cars (Sedan, Pickup, Minivan, SUV) from 1-D time series created from a mapping of the cars' contours. *OliveOil* is a dataset containing spectrographs of extra virgin olive oils from four different countries, and the problem targets at predicting the country of origin of each oil from spectrograph information. The *DistalPhalanxTW* dataset gathers time series obtained from a mapping of the contour of bone image data, and the goal is to predict the Tanner-Whitehouse score which gathers information about the age of a person in 6 classes. Finally, the nature of the *Plane* dataset is similar to the *Car* dataset, and contains mappings of the contour of seven different planes to 1-D time series. The goal is to predict the type of aircraft based on this information.

4.2 Experimental settings

Experiments were performed on a desktop computer Intel(R) Core(TM) i5-9600K CPU at 3.70GHz with 32GB RAM. The classical feedforward neural networks were implemented in Pytorch 1.13, and the classic/quantum approach using Pytorch 1.13 for the classical part and Xanadu's Pennylane-Pytorch bridge for the quantum algorithms. The source code of all experiments is available at https://github.com/manupc/qtsclassification.

A preliminary experimentation stage was carried out following a trial-and-error procedure to find out the best hyperparameters for each model and problem, and Table 2 summarizes the pre-processing operations and network structure for each case. After this prior experimentation, we set the number of quantum layers for all datasets to 1 in the hybrid classical/quantum neural network. Additionally, the Adam algorithm was used for all experiments. The loss functions used to train the models were the Binary Cross-Entropy from logits for the binary classification problems, and the Cross-Entropy for the multiclass datasets. The remaining settings in Table 2 correspond to the time series preprocessing step, where two possible operations were considered: Resampling to a new time series length, and differentiation. The Pre-Quantum and Post-Quantum rows of QNN, and the Structure row of ANN, establish the layer structure of classical networks structures, where *Linear(x)* represents a fully-connected linear layer with x neurons, and *ReLU*/*SoftMax* stands for the corresponding activation functions.

We carried out 30 different executions of each problem and model with the settings described in Table 2, in order to analyze the results statistically.

Table 1 Datasets used for the experimentation

Name	# classes	# training samples	# test samples	Length
Coffee	2	28	28	286
Earthquakes	2	322	139	512
Car	4	60	60	577
OliveOil	4	30	30	570
DistalPhalanxTW	6	400	139	80
Plane	7	105	105	144

Table 2 Hyperparameter settings for the experimentation

Model	Parameter	Coffee	Earthquakes	Car	OliveOil	DistalPhalanxTW	Plane
Preprocessing	Resampling	256	16	256	256	–	128
	Differenciation	No	No	No	Yes	No	No
QNN	Pre-Quantum	Linear(256)	Linear(4)	Linear(16)	Linear(256)	Linear(100) ReLU Linear(64)	Linear(128)
	Qubits	8	2	4	8	6	7
	Post-Quantum	Linear(1)	Linear(1)	Aggregation	Aggregation	Linear(6) SoftMax	Aggregation
	Learning Rate	0.1	0.1	0.001	0.001	0.01	0.005
	Adam iterations	100	100	600	200	600	600
ANN	Structure	Linear(10) ReLU Linear(1)	Linear(20) ReLU Linear(20) ReLU Linear(1)	Linear(500) ReLU Linear(16) ReLU Linear(4) SoftMax	Linear(200) ReLU Linear(4) SoftMax	Linear(100) ReLU Linear(6) SoftMax	Linear(100) ReLU Linear(7) SoftMax
	Learning Rate	0.001	0.001	0.001	0.001	0.0001	0.0001
	Adam iterations	200	2000	1000	5000	7000	5000

4.3 Results and discussion

The results are mainly analyzed using the *accuracy* metric, since it is the value provided in the reference methods in the literature for the studied datasets (Dau et al. 2019). Table 3 summarizes the results obtained with the hybrid classical/quantum approach (QNN) and classical neural networks (ANN). Each dataset is analyzed from Column 2 to Column 7. Rows 2–3 show the average percentage accuracy obtained on training/test sets for both QNN and ANN. After that, rows 4–5 print the standard deviation of the accuracy in the training/test sets in our experiments. Then rows 6–7 and 8–9 describe the worst and best accuracy obtained in training/test, respectively, and rows 10–11 plot the average computational time of a training run, measured in seconds. The last row of the Table 3 prints the accuracy on the test set of the best-reported method for each dataset, obtained from the UEA & UCR Time Series Classification Repository website, which we use as reference for comparisons.

A non-parametric Mann–Whitney U test with a confidence level of 95% was applied to check whether there are significant differences between the results of QNN and ANN in the accuracy distributions in the test sets. If so, then row 2 of Table 3 prints the symbol *(+)* if QNN performs better

Table 3 Summary of results in the experimentation

	Coffee	Earthquakes	Car	OliveOil	DistalPhalanxTW	Plane
QNN Mean (Tr/Ts)	100.00 / 100.00 (x)	88.47 / 77.17 (x)	100.00 / 88.00 (+)	99.78 / 96.22 (+)	82.16 / 66.21 (-)	100.00 / 99.52 (+)
ANN Mean (Tr/Ts)	98.33 / 97.86	92.46 / 76.91	93.89 / 77.00	93.78 / 88.89	80.50 / 69.06	98.44 / 95.78
QNN Sd (Tr/Ts)	0.00 / 0.00	2.52 / 1.38	0.00 / 1.52	0.83 / 1.13	3.51 / 4.27	0.00 / 0.48
ANN Sd (Tr/Ts)	8.98 / 9.65	3.46 / 1.10	4.80 / 3.56	1.66 / 1.57	0.86 / 0.32	3.07 / 6.95
QNN Worst (Tr/Ts)	100.00 / 100.00	80.43 / 74.82	100.00 / 85.00	96.67 / 93.33	72.50 / 58.27	100.00 / 99.05
ANN Worst (Tr/Ts)	50.00 / 46.43	78.88 / 74.82	68.33 / 58.33	93.33 / 86.67	80.25 / 68.35	90.48 / 78.10
QNN Best (Tr/Ts)	100.00 / 100.00	94.10 / 80.58	100.00 / 91.67	100.00 / 96.67	89.00 / 74.10	100.00 / 100.00
ANN Best (Tr/Ts)	100.00 / 100.00	94.10 / 79.14	95.00 / 78.33	100.00 / 90.00	85.00 / 70.50	100.00 / 99.05
QNN Avg. Time (s.)	28.54	263.81	615.20	244.54	5269.19	1657.38
ANN Avg. Time (s.)	0.13	4.72	2.79	6.91	13.26	10.45
Best reported solution	99.96	75.92	90.18	90.13	69.32	100

than ANN on a dataset, the symbol *(-)* if QNN is worse than ANN, and *(x)* if no statistical differences were found. We also provide boxplots of the results for he accuracy populations on the test sets in Fig. 10 to support the discussion.

If we analyze the average learning ability of QNN with respect to ANN, Table 3 points out that QNN was better in 3 problems, worse in 1 problem, and equivalent in two of them. On the other hand, if we focus on the standard deviation as a measure of the robustness of the models, we can see that the results of the QNN models have less variability than the ANNs in the test sets, for all problems except for *DistalPhalanxTW* and *Earthquakes*, which is a desirable feature in any machine learning model intended for deployment. In the case of *DistalPhalanxTW*, Fig. 10e verifies that the standard deviation of test accuracy for ANN is lower than in QNN, but also that the latter model is capable of achieving better results. This fact suggests that the dataset is difficult to learn and it is easy to fall into local optima solutions that, in the case of QNN, can be surpassed more easily than in ANN despite the fact that worse solutions can also be found. On the other hand, the analysis for the case of the *Earthquakes* dataset is similar. Although both models performed equivalently in the problem according to the Mann–Whitney *U* test, we can see in Fig. 10b that QNN found the best solution as an outlier. This fact, and also that the median of accuracy in the boxplot is higher for QNN though the first quartile is similar to that of ANN, could explain the small increase in variance for this model. Finally, we also highlight the variability of 0.0 obtained by QNN on the *Coffee* dataset, which means that the best solution with 100% accuracy was found in all runs, which also supports the previous analysis.

The study on the average accuracy and variability in the test set, with regards to QNN, may seem counterintuitive with respect to the existing analyses in the literature that indicate that the vanishing gradient problem is harder to solve in Quantum Neural Networks than in traditional ones. This is a well-documented fact in different articles (Andrés et al. 2022; Wang 2017; Skolik et al. 2022), and it is mainly due to the fact that QNNs are implemented in sequences of atomic operations rather than in parallel computations performed in the same layer in classical neural networks. Gradient propagation is more difficult from the output to the first layer. If we also consider that quantum operations are implemented as qubit rotation gates whose internal mathematics involve trigonometric calculus, we may derive that the gradient can become zero after a few propagations. However, this is not our case, since all experiments used a single quantum layer so that the vanishing gradient problem was not present in our study.

If we focus on the best possible accuracy in test sets found by each model, rows 8–9 in Table 3 highlight in bold the model with the best performance in a single training experiment, and row 2 highlights those in which the QNN proposal was better, on average, than the reported solution in the literature after the 30 experiment runs were finished. The QNN proposal was able to achieve the best results with respect to ANN in all problems, although the ANN approach also found the best solution for the *Coffee* dataset. The fact that QNNs

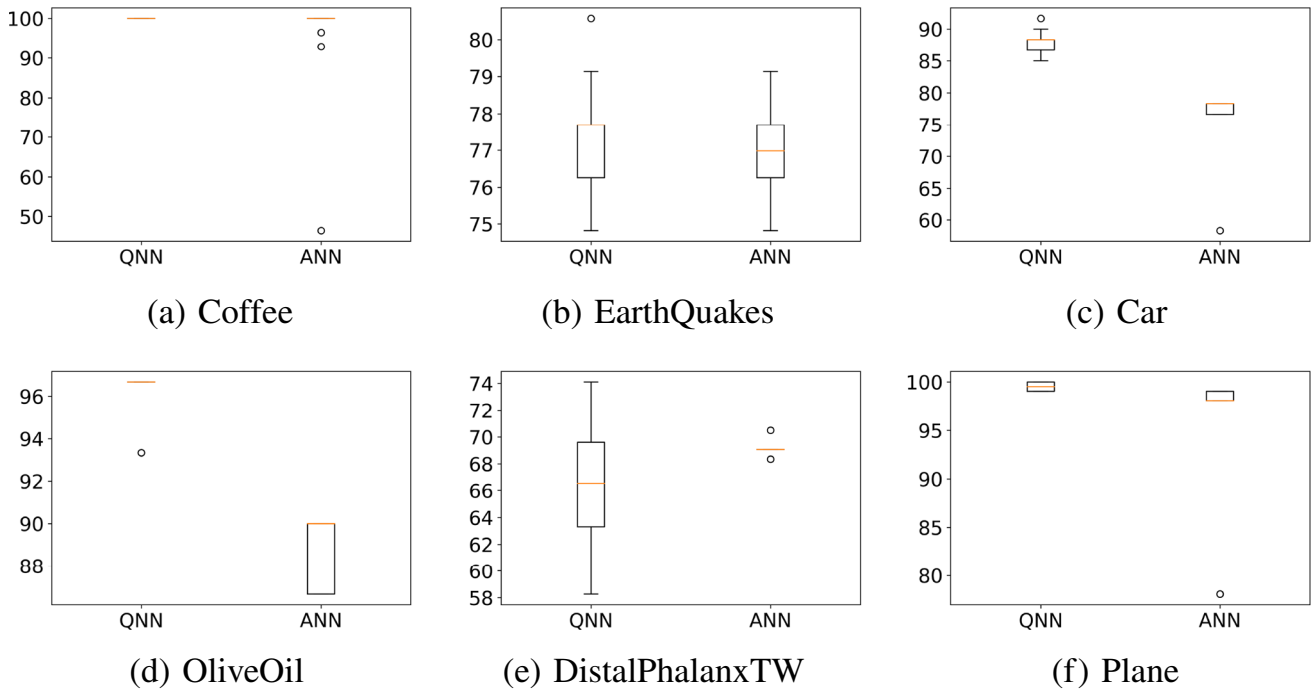

Fig. 10 Boxplots of QNN and ANN resulting accuracy populations in test sets

are able to achieve absolute better results in test accuracy, together with the results of the Mann–Whitney U test that stated that QNNs were equivalent or better than ANNs in five of six problems studied, suggests that these models may be a relevant alternative to be considered in the problem of Time Series Classification. However, as rows 10–11 in Table 3 suggest, this comes at the cost of a huge increase in the computational time required to train the QNN model which, in the case of *DistalPhalanxTW*, is over 500% with respect to the training time of classic ANNs. We remark that this computational time was measured over the quantum algorithm simulation Xanadu's Pennylane which is also noise-free. We expect that, in the next few years, quantum computers with a suitable reduced noise can be widely accessible to verify whether there is a true gain in computational time beyond simulation.

To conclude the analysis and discussion of the results, we compare the outcomes of the QNN and ANN models with the best accuracy in the test reported in the literature (last row of Table 3). From rows 8–9, we can observe that QNNs have the potential to provide better results than ANNs on all datasets except *Coffee*, where the best test solutions are optimal in both cases. Moreover, QNNs were able to improve the best average accuracy reported in the problems *Coffee*, *Earthquakes* and *OliveOil*. This fact reinforces the analysis of the potential of quantum computing, and QNNs in particular, to solve Time Series Classification problems, since these models can represent a Time Series compactly using amplitude embedding, but also improve the results on the studied datasets.

5 Conclusions

In this work, we have studied the problem of time series classification using quantum machine learning. In particular, a hybrid classical /quantum neural network was designed to solve the problem, and we used amplitude embedding to encode a time series of length T into $log_2(T)$ qubits as a compact data representation. The designed network also includes a novel method to perform measurement on the quantum classifier and proposes to aggregate the measurements of all qubits to take into account all information encoded in the quantum algorithm, to finally calculate the class labels. Experiments conducted on benchmark time series classification problems, with a varied number of target classes ranging from 2 to 7, suggest that the quantum neural network proposal is a competitive model capable of obtaining the best accuracy results with respect to both classical neural networks and literature reports, although the time complexity to train the models is much higher than for classical artificial neural networks. In future works, we plan to extend the proposed methodology to other problems with a larger number of data, by designing parallelization methods for quantum algorithms simulation, and also to explore the potential of quantum neural networks in other machine learning problems.

Author contribution All authors contributed equally to this work.

Funding This article was funded by the project QUANERGY (Ref. TED2021-129360B-I00), Ecological and Digital Transition R&D projects call 2022 by MCIN/AEI/10.13039/501100011033 and European Union NextGeneration EU/PRTR, and supported by Grant PID2021-128970OA-I00 funded by MCIN/AEI/10.13039/501100011033/FEDER.

Data Availibility Statement The datasets used are available on the website of the UEA & UCR Time Series Classification Repository at https://www.timeseriesclassification.com

References

Aïmeur E, Brassard G, Gambs S (2007) Quantum clustering algorithms. In: Proceedings of the 24th international conference on machine learning. ICML '07, pp 1–8. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/1273496.1273497

Andrés E, Cuéllar MP, Navarro G (2022) On the use of quantum reinforcement learning in energy-efficiency scenarios. Energies 15(16). https://doi.org/10.3390/en15166034

Araujo IF, Park DK, Petruccione F, Silva AJ (2021) A divide-and-conquer algorithm for quantum state preparation. Sci Rep 11:6329–6341. https://doi.org/10.1038/s41598-021-85474-1

Bagnall A, Lines J, Hills J, Bostrom A (2015) Time-series classification with cote: the collective of transformation-based ensembles. IEEE Trans Know Data Eng 27:1–1. https://doi.org/10.1109/TKDE.2015.2416723

Bausch J (2020) Recurrent quantum neural networks

Bokhan D, Mastiukova AS, Boev AS, Trubnikov DN, Fedorov AK (2022) Multiclass classification using quantum convolutional neural networks with hybrid quantum-classical learning. Front Phys 10. https://doi.org/10.3389/fphy.2022.1069985

Crooks GE (2019) Gradients of parameterized quantum gates using the parameter-shift rule and gate decomposition

Cuturi M (2011) Fast global alignment kernels. In: Proceedings of the 28th international conference on machine learning, ICML 2011, pp 929–936

Dau HA, Bagnall A, Kamgar K, Yeh C-CM, Zhu Y, Gharghabi S, Ratanamahatana CA, Keogh E (2019) The UCR time series archive. IEEE/CAA J Autom Sinica 6(6):1293–1305. https://doi.org/10.1109/JAS.2019.1911747

Deng H, Runger G, Tuv E, Vladimir M (2013) A time series forest for classification and feature extraction. Inf Sci 239:142–153. https://doi.org/10.1016/j.ins.2013.02.030

Deutsch D, Jozsa R (1992) Rapid solution of problems by quantum computation. In: Proceedings of the royal society a: mathematical, physical and engineering sciences, pp 553–558. University of Bristol, GBR

Dong D, Chen C, Li H, Tarn T-J (2008) Quantum reinforcement learning. IEEE Trans Sys Man Cybern Part B (Cybernetics) 38(5):1207–1220. https://doi.org/10.1109/TSMCB.2008.925743

Emmanoulopoulos D, Dimoska S (2022) Quantum machine learning in finance: time series forecasting. arXiv. arXiv:2202.00599

Farhi E, Goldstone J, Gutmann S (2014) A quantum approximate optimization algorithm

Fawaz HI, Forestier G, Weber J, Idoumghar L, Muller P-A (2018) Deep learning for time series classification: a review. CoRR. arXiv:1809.04356

Feynman RP (1982) Simulating physics with computers. Int J Theor Phys 21(6):467–488. https://doi.org/10.1007/BF02650179

Ganguly S (2021) Quantum machine learning: an applied approach. Apress, New York

Grover LK (1997) Quantum mechanics helps in searching for a needle in a haystack. Phys Rev Lett 79:325–328. https://doi.org/10.1103/PhysRevLett.79.325

Kate R (2015) Using dynamic time warping distances as features for improved time series classification. Data Mining and Knowl Discov 30. https://doi.org/10.1007/s10618-015-0418-x

Liu Z, Zhu Z, Gao J, Xu C (2021) Forecast methods for time series data: a survey. IEEE Access PP:1–1. https://doi.org/10.1109/ACCESS.2021.3091162

Macaluso A, Clissa L, Lodi S, Sartori C (2020) A variational algorithm for quantum neural networks. In: Krzhizhanovskaya VV, Závodszky G, Lees MH, Dongarra JJ, Sloot PMA, Brissos S, Teixeira J (eds) Computational Science - ICCS 2020. Springer, Cham, pp 591–604

Maheshwari D, Sierra-Sosa D, Garcia-Zapirain B (2022) Variational quantum classifier for binary classification: real vs synthetic dataset. IEEE Access 10:3705–3715. https://doi.org/10.1109/ACCESS.2021.3139323

Peruzzo A, McClean J, Shadbolt P, Yung M-H, Zhou X-Q, Love PJ, Aspuru-Guzik A, O'Brien JL (2014) A variational eigenvalue solver on a photonic quantum processor. Nat Commun 5(1). https://doi.org/10.1038/ncomms5213

Pushpak SN, Jain S (2021) An introduction to quantum machine learning techniques. In: 2021 9th International conference on reliability, infocom technologies and optimization (trends and future directions) (ICRITO), pp 1–6. https://doi.org/10.1109/ICRITO51393.2021.9596240

Rajesh V, Naik UP, Mohana (2021) Quantum convolutional neural networks (QCNN) using deep learning for computer vision applications In: 2021 International conference on recent trends . on electronics, information, communication & technology (RTEICT), pp 728–734. https://doi.org/10.1109/RTEICT52294.2021.9574030

Rivera-Ruiz MA, Mendez-Vazquez A, López-Romero JM (2022) Time series forecasting with quantum machine learning architectures. In: Pichardo Lagunas O, Martínez-Miranda J, Martínez Seis B (eds) Advances in computational intelligence. Springer, Cham, pp 66–82

Schäfer P (2015) The boss is concerned with time series classification in the presence of noise. Data Mining Knowl Discov 29. https://doi.org/10.1007/s10618-014-0377-7

Schmidl S, Wenig P, Papenbrock T (2022) Anomaly detection in time series: a comprehensive evaluation. Proc. VLDB Endow. 15(9):1779–1797. https://doi.org/10.14778/3538598.3538602

Schuld M, Sinayskiy I, Petruccione F (2014) Quantum computing for pattern classification. In: Pham D-N, Park S-B (eds) PRICAI 2014: trends in artificial intelligence. Springer, Cham, pp 208–220

Shor PW (1999) Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM Review 41(2):303–332. https://doi.org/10.1137/S0036144598347011

Skolik A, Jerbi S, Dunjko V (2022) Quantum agents in the Gym: a variational quantum algorithm for deep Q-learning. Quantum 6:720. https://doi.org/10.22331/q-2022-05-24-720

Sutor R (2019) Dancing with Qubits. Packt, Birmingham, UK

Umer MJ, Sharif MI (2022) A comprehensive survey on quantum machine learning and possible applications. Int J E-Health Med Commun 13(5):1–17. https://doi.org/10.4018/IJEHMC.315730

Wang G (2017) Quantum algorithm for linear regression. Phys Rev A 96:012335. https://doi.org/10.1103/PhysRevA.96.012335

Warren Liao T (2005) Clustering of time series data—a survey. Patt Recog 38(11):1857–1874. https://doi.org/10.1016/j.patcog.2005.01.025

Wittek P (2014) Quantum machine learning: what quantum computing means to data mining. Elsevier, Amsterdam, The Netherlands

Yarkoni S, Kleshchonok A, Dzerin Y, Neukart F, Hilbert M (2021) Semi-supervised time series classification method for quantum computing. Quantum Mach Intell 3(12):1–11. https://doi.org/10.1007/s42484-021-00042-0