Time Series forecasting with Quantum Neural Networks*

M.P. Cuéllar[1][0000−0002−9736−1608], M.C. Pegalajar[1][0000−0001−9408−6770], L.B.G. Ruiz[2][0000−0001−6716−5115], and C. Cano[1][0000−0002−0181−2444]

[1] Department of Computer Science and Artificial Intelligence, University of Granada (Spain) manupc@decsai.ugr.es / mcarmen@decsai.ugr.es
http://www.ugr.es
[2] Department of Software Engineering, University of Granada (Spain)
bacaruiz@decsai.ugr.es

Abstract. In this work we explore the use of Quantum Computing for Time Series forecasting. More specifically, we design Variational Quantum Circuits as the quantum analogy of feedforward Artificial Neural Networks, and use a quantum neural network pipeline to perform time series forecasting tasks. According to our experiments, our study suggests that Quantum Neural Networks are able to improve results in error prediction while maintaining a lower number of parameters than its classical machine learning counterpart.

Keywords: Quantum Neural Networks · Quantum Machine Learning · Time Series Forecasting.

1 Introduction

Quantum Computing (QC) [15] was born in 1982 after Richard Feynman pointed out the complexity of simulating a quantum system with a classic computer. Since then, QC has been growing as a research area until nowdays, where contemporary applications of QC are varied and include cryptography, finance, game theory, chemical modelling, or machine learning [5][10][12][17], to mention just a few. The recent reality of QC hardware and the existence of quantum computer simulators able to run in classical computers have contributed significantly to improve the state-of-the-art in QC, although quantum supremacy (understood as a significant speedup from exponential time to polynomial time) have not been achieved but for a handful of applications such as search in unordered sets in $\mathcal{O}(\sqrt{n})$ with Grover's search, finding if a function is balanced or not with Deutsch-Jozsa method, or integer factorization with Shor's algorithm [15], to mention the most sounded examples.

* This article was supported by the project QUANERGY (Ref. TED2021-129360B-I00), Ecological and Digital Transition R&D projects call 2022, Government of Spain, and Grant PID2021-128970OA-I00 funded by MCIN/AEI/10.13039/501100011033/FEDER.

In we focus on Quantum Machine Learning (QML) [4], all types of supervised [11], unsupervised [13], and reinforcement learning [1][6] tasks have been explored with this new computing paradigm. Most of the approaches show the benefits of QML to solve these tasks, and often they provide benefits in either time complexity and/or performance. In this work, we adopt the supervised learning paradigm under QC, and propose a Quantum Neural Network [8] to solve tasks of time series forecasting, with the goal of performing a preliminar evaluation of the scope of QC to tackle this type of problem.

A Time Series is a sequence of measurements or observations of a given phenomenon, sampled periodically and indexed in time. Time series forecasting is an ubiquitous problem to almost all areas in science, and attempts to predict future values of the data series $x(t+1)$ with historical time series data $x(t), x(t-1), x(t-2), ...$, and a hypothesis model f often parameterized with parameters θ, i.e. $x(t+t) = f(x(t), x(t-1), x(t-2), x(t-3), ..., \theta)$. Although the number of models used for time series forecasting is wide and the proposals come from different areas (statistics, electronics, computer science, economics, etc.), in this work we focus in the special case of neural networks for forecasting [16]. Both feedforward and recurrent neural network models have been extensively tested in a wide variety of problems, as it is described in the survey [3].

On the other hand, if we focus on the problem of time series forecasting with Quantum Computing, the reference literature is scarce and very few research articles have addressed the problem. In particular, the work [2] proposed a new neural network model whose computation units are inpired in quantum amplitude and phase operations, and applied the proposal for stock market forecasting. Lately, in [14] it is proposed an adaptation of the Quantum-inspired Optimization Algorithm (QOA) for fuzzy sets, and the approach was tested over the TAIFEX stock market time series and temperature time series, among others. We remark that these two approaches are not purely from QC, although they are inspired by elements of QC to build classical models. Pure QC approaches are the work [9], that proposes a hybrid classical/quantum neural network containing layers from both computing paradigms and trained the model to predict the Sun Spot time series; and the article [7] which develops a framework for quantum machine learning temporal tasks using reservoir computing and quantum neural networks (QNNs), applied to S&P 500 stock market time series prediction problems. In these cases, all papers in the literature report an increase in performance/accuracy of the QC models with respect to classic computing ones.

In this manuscript, we design a pure QC neural network using Variational Quantum Circuits, and test the approach in classic benchmark time series prediction datasets. Since loops are not allowed in a quantum algorithm, the designed quantum neural network has a feedforward structure, and contribute to the existing literature by means of the proposal of how to encode time series data into quantum states, the QNN design and measurement of results, and a proof-of-concept experimentation to assess limitations and future possible ways to address the problem of scalability.

This work is structured as follows: Section 2 makes an overview of QC and QML to make this article self-contained. After that, Section 3 review the concept of Quantum Neural Network and describes our approach. Then, Section 4 describes the experimentation performed, and Section 5 concludes.

2 Quantum Computing and Machine Learning

This section contains a brief introduction to Quantum Computing and Quantum Machine Learning for article self-completeness. The main references used to write this section, and for further information, are [10][15] for QC and [5] for QML.

There is no need to argue that Quantum Computing is a completely different computer programming paradigm to the traditional (classic) computer programming. However, we may find some common elements in both: Classic computing is based on computation over binary digits (bits), and the output of a classic computing algorithm is a set of n bits with values in $\{0,1\}^n$ whose underlying mathematical model is $\mathbb{Z}_2^n$, i.e. the cartesian product of $\mathbb{Z}_2$ n times or $\times^n\mathbb{Z}_2$. Similarly, QC is based on operations over quantum binary digits (qubits), and the output of a QC system is also a set of n binary values in $\{|0\rangle, |1\rangle\}$. However, the underlying mathematical model of QC is a vector subspace of $\mathbb{C}^{2^n}$ obtained by means of the tensor product of $\mathbb{C}^2$ n times, i.e. $\bigotimes^n \mathbb{C}^2$. The values $|0\rangle = (1,0)^t$ and $|1\rangle = (0,1)^t$ are the orthonormal basis column vectors of the vector subspace of a system with one qubit, also refered to as the *computational basis*. For systems with a larger number of qubits, the computational basis is calculated by means of the tensor product of the qubit basis vectors, as for instance $\{|0\rangle \otimes |0\rangle = |00\rangle = (1,0,0,0)^t, |0\rangle \otimes |1\rangle = |01\rangle = (0,1,0,0)^t, |1\rangle \otimes |0\rangle = |10\rangle = (0,0,1,0)^t, |1\rangle \otimes |1\rangle = |11\rangle = (0,0,0,1)^t\}$ for the case of 2 qubits. It is worth noting that, when a new bit is included into a classical system, the dimension of the computational space increases by one as a consequence of the cartesian product operation; however, when a new qubit is included into a quantum system the dimension of the computational space doubles its size.

As a member of a vector subspace in $\mathbb{C}^2$, the value of an arbiratry qubit $|\psi\rangle$ can be modelled as a linear complex combination of the basis states $\{|0\rangle, |1\rangle\}$ as $|\psi\rangle = \alpha_0 |0\rangle + \alpha_1 |1\rangle, \alpha_i \in \mathbb{C}$ with the additional constraint that $\sum_i |\alpha_i|^2 = 1$, so that a qubit can potentially hold an infinite number of values. The coefficients α_i are called *amplitudes* and, when the user retrieves the output of a quantum algorithm through the *measurement* operator, the qubit collapses to value $|0\rangle$ with probability $|\alpha_0|^2$ or to value $|1\rangle$ with probability $|\alpha_1|^2$. This also holds for systems with a larger number of qubits, as for instance $|\psi\rangle = \alpha_0 |00\rangle + \alpha_1 |01\rangle + \alpha_2 |10\rangle + \alpha_3 |11\rangle$ for a 2-qubit system.

Operations in a classic algorithm are implemented at a fundamental level using a sequence of logic gates such as AND, NOT, OR, etc., whose mathematical model relies on the addition and product over the field with two elements $\mathbb{Z}_2$. Similarly, a quantum algorithm can also be implemented using quantum gates; however, these gates are a reversible linear transformation over the complex space $\mathbb{C}^{2^n}$ and are modelled as unitary matrices that multiply the quantum

state. For example, the σ_x gate/operation in equation 1 is the quantum analog to the NOT classical gate, and its application over qubits with values $|0\rangle$ or $|1\rangle$ is written as $\sigma_x |0\rangle = |1\rangle$ or $\sigma_x |1\rangle = |0\rangle$, respectively. Another example is the Hadamard (H) gate in equation 2, where $H |0\rangle$ moves the qubit into the standard superposition of the basis states $|\psi\rangle = \frac{\sqrt{2}}{2} |0\rangle + \frac{\sqrt{2}}{2} |1\rangle$. A final example of a gate with 2 inputs is shown in equation 3 and it corresponds to the Controlled-NOT (CNOT) gate which switches the second qubit from $|0\rangle$ to $|1\rangle$ or vice versa if the first qubit is $|1\rangle$, of leaves it unchanged otherwise. In the general case, for an arbitrary 2-qubit state $|\psi\rangle = \alpha_0 |00\rangle + \alpha_1 |01\rangle + \alpha_2 |10\rangle + \alpha_3 |11\rangle$, the CNOT exchanges the amplitudes of the last basis vectors so that $CNOT |\psi\rangle = \alpha_0 |00\rangle + \alpha_1 |01\rangle + \alpha_3 |10\rangle + \alpha_2 |11\rangle$.

$$\sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \tag{1}$$

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

$$CNOT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{3}$$

Of special interest to our work are the set of parameterized gates, i.e. gates whose behaviour depends of input parameters θ, such as the rotation gates $R_x(\theta), R_y(\theta), R_z(\theta)$, whose unitary matrices for one qubit are described in equations 4-6, respectively. These gates allow to change the output of a quantum algorithm depending on the parameter θ, and are the fundamental block to build the quantum neural network used in this work.

$$R_x(\theta) = \begin{pmatrix} cos(\theta/2) & -isin(\theta/2) \\ -isin(\theta/2) & cos(\theta/2) \end{pmatrix} \tag{4}$$

$$R_y(\theta) = \begin{pmatrix} cos(\theta/2) & -sin(\theta/2) \\ sin(\theta/2) & cos(\theta/2) \end{pmatrix} \tag{5}$$

$$R_z(\theta) = \begin{pmatrix} e^{-i\theta/2} & 0 \\ 0 & e^{i\theta/2} \end{pmatrix} \tag{6}$$

Quantum algorithms are described as a sequence of operations over a quantum state, as for instance the operations $CNOT((H |q_0\rangle) \otimes |q_1\rangle)$, and are implemented into quantum circuits. Figure 1 shows the implementation of the previous algorithm, which starts from state $|q_0 q_1\rangle = |00\rangle$ and measures state $|00\rangle$ or $|11\rangle$ with probability 0.5 into classical bits (line c). In a circuit, each horizontal line is assigned to a single qubit, and gates are organized sequentially until measurement. Thus, loops are not allowed in a quantum program. A special case of a quantum circuit is the Variational Quantum Circuit (VQC), whose

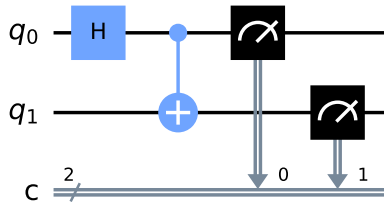


Fig. 1: Example of circuit implementing the algorithm $CNOT((H\,|q_0\rangle) \otimes |q_1\rangle)$.

main feature is that it contains parameterized gates such as the aforementioned $R_x(\theta), R_y(\theta), R_z(\theta)$.

Quantum Machine Learning has been a research area of growing interest for the last two decades; however, it has obtained a special focus in the last few years thanks to the advances in Quantum Computer Hardware and Quantum Computing simulators. The main goal of QML is to design and implement methods able to run in a quantum computer to solve the traditional supervised, unsupervised and reinforcement learning tasks of classic Machine Learning, taking advantages of quantum operations that are not present in a classic computer such as superposition, tunneling, entanglement, or quantum parallelism, coming from Quantum Computing (QC). In this work, we focus on the case of Quantum Neural Network (QNN) design. A QNN is the quantum analog of a classic neural network. Each layer of a QNN is a VQC containing parameterized gates, where the parameters are the quantum analog of the classic network weights. It also contains a mechanism to transfer information among the existing qubits as an analogy to a classic connection between neurons of different layers. Usually, this information transfer is implemented as entanglements using operators such as the CNOT gate.

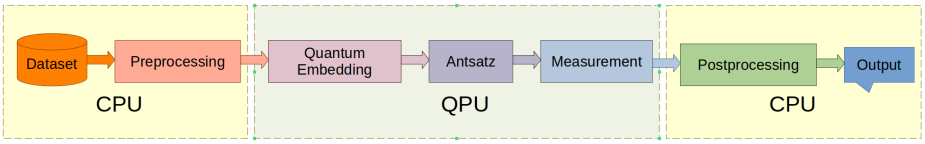


Fig. 2: Usual pipeline of Quantum Machine Learning, where CPU stands for operations performed on a classical computer and QPU operations over a quantum hardware.

The process to create a QML model usually involves the following steps (see Figure 2): First, the dataset is loaded and preprocessed into a classical CPU. After that, the classic data are encoded into quantum states on a quantum

hardware of QPU using a quantum embedding technique. Once the classic data has been represented into quantum states, the core model implemented in the antsatz is executed and its results measured into classical bits. Finally, these results are post-processed if necessary in CPU to provide the expected model output. In this work, we follow this general pipeline to study how a Quantum Neural Network can be used for time series forecasting.

3 Variational Quantum Circuits for forecasting

A Quantum Neural Network [8] can be tipically organized as a sequence of layers:

- The **input layer**, in charge of transforming the classic input data into a quantum state.
- The **antsatz**, containing a Variational Quantum Circuit whose structure is concatenated L times to create the quantum analog of L network layers.
- The **output layer**, which performs measurement operations over qubits to return the expected outcome.

The **input layer** is usually implemented as a parameterized variational circuit with rotation and controlled-rotation gates that help to set the desired quantum state for a given input classic data. This process is called the *quantum embedding* procedure, and it encompasses a set of techniques such as basis encoding, amplitude encoding, hamiltonian encoding, or tensor product encoding, to mention just a few. In this work, we use the tensor product encoding consisting of a single X-rotation gate for each qubit, where the gate parameter is the classic data scaled to $[-\pi, \pi]$. This is a simple and fast encoding technique which requires operations in $O(1)$ to perform the creation of a quantum state; however, it has the limitation that the number of qubits must increase with the number of input classic data linearly. In addition, quantum embedding can be influenced by the bias and scale of the input dataset and, for that reason, we have modified the tensor product classic scheme to include further learnable parameters to scale and bias the input data. Figure 3 shows an example of the input layer for a network containing 3 qubits. Values I_i are the classic data features, θ_i are the input scale parameter and b_i the bias parameter.

With respect to the **antsatz**, we may notice that the literature does not offer a set of fixed quantum layer structures as there are in the classic neural network domain (fully connected, recurrent, etc.). The number of possible gates used for quantum information transfer between qubits is wide, and the organization of these gates to make the data transfer has not been extensively studied yet. In this work, we use the Real Amplitudes antsatz which has been used previously in other domains with success such as policy estimation for quantum reinforcement learning and classification. The antsatz starts with full rotation X/Y/Z parameterized gates as the quantum analog of connection weights, followed by a set of CNOT gates organized with a ring structure for the qubit information transfer. Figure 4 shows the implementation of the described antsatz as the analogy of a quantum network layer for a 3-qubit network. Thus, a quantum network layer in

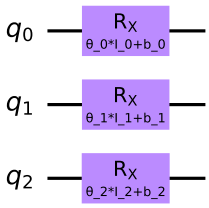


Fig. 3: Tensor product embedding for a classic data containing 3 features in $[-\pi, \pi]$.

our work contains a number of $3 * n$ parameters, where n stands for the number of qubits.

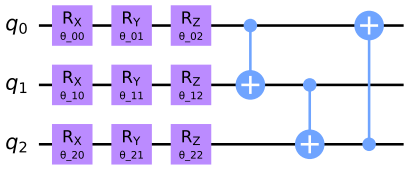


Fig. 4: *Real Amplitudes* antsatz for a 3-qubit network.

$$\sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \tag{7}$$

With respect to the **output layer**, *measurement* is often performed over a selected observable. A typical observable is the σ_z operator over the computational basis (see equation 7). The network output can be calculated by means of the expectation of the observable over a quantum state, i.e. $\langle\psi|\,\sigma_z\,|\psi\rangle$, where $\langle\psi|$ stands for the conjugate transpose of $|\psi\rangle$, so that the output is in the range $[-1, 1]$. This must be taken into account for the QML system design, since the output data must be scaled if the target patterns in our dataset have a different range. In this work, we use the expectation of the σ_z observable over the first qubit q_0 of our network as the network output. However, we append a final scale parameter and bias to be learned, so that the network output is less sensitive to dataset bias and scale. The whole designed model is depicted in Figure 5.

As we are using Quantum Computing simulation software, the training of the proposed QNN is performed in CPU using classic algorithms such as the Adam optimizer. Gradient computation is performed in CPU using the classic propagation rules, while the gradient in the QPU is calculated using the parameter-

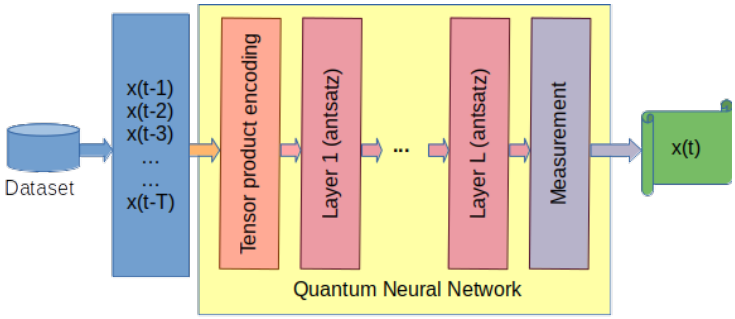


Fig. 5: Proposed Quantum Neural Network model.

shift rule. Figure 6 shows the training process pipeline, where θ^1 stand for the scale/bias parameters in the input layer, θ^2 are the parameters of the layers containing the antsatz, and θ^3 are the scale/bias parameters for the network outputs.

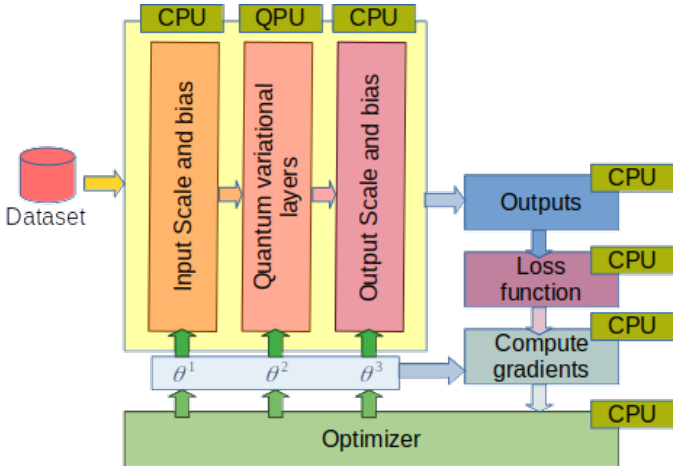


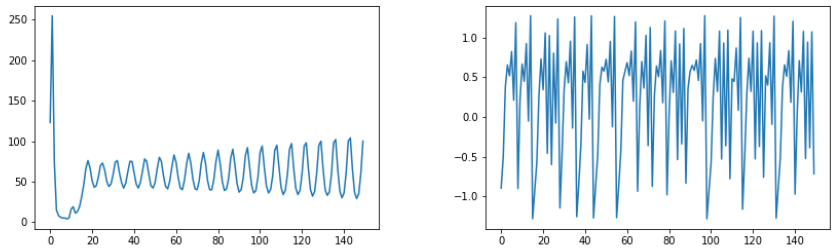
Fig. 6: Proposed Quantum Neural Network model.

The use of the proposed QNN model for time series forecasting is as follows: Since a QNN is a feedforward model, we first set a time horizon T, and the time series must be transformed to tabular data where the target is the time series value at time $t, x(t)$, and the inputs are the values $x(t-1), x(t-2), ..., x(t-T)$, as described in Figure 5.

4 Experiments

The experimentation in this paper is a proof-of-concept regarding the capabilities of QNNs to perform time series prediction. For that reason, we used two classic and well-tested time series: The **laser** time series from the far-infrared laser dataset A of the Sata Fe Time Series competition, and the synthetic **Henon** map time series. Figure 7 shows the time series used, containing 150 data points each. As a QNN has a feedforward structure, we compare the results with a classic Multilayer Perceptron (MLP) instead of more complex recurrent neural networks, in order to make the most fair comparison possible. The time series were divided into the first 75% of data for training/test, and the remaining 25% for validation. To ease QNN learning, we scaled the time series to the interval $[-1, 1]$ as a preprocessing step.

The MLP model was implemented in tensorflow 2.7, containing 1 layers with 10 neurons, the *tanh* activation function in the hidden layer and the identity in the output layer. On the other hand, the QNN was implemented in Tensorflow Quantum 0.6.1 and, in both datasets, a single layer was included in the network structure. The training algorithm was Adam with a learning rate of 0.01 for both MLP and QNN, and 30 different executions were performed using 4-fold cross-validation, to make a statistical analysis of results over a desktop computer Intel(R) Core(TM) i5-9600K CPU at 3.70GHz with 32GB RAM with a NVidia GeForce RTX 2060 GPU. The source code for this experimentation is available at https://github.com/manupc/qnn_tsp.



(a) Laser (b) Henon

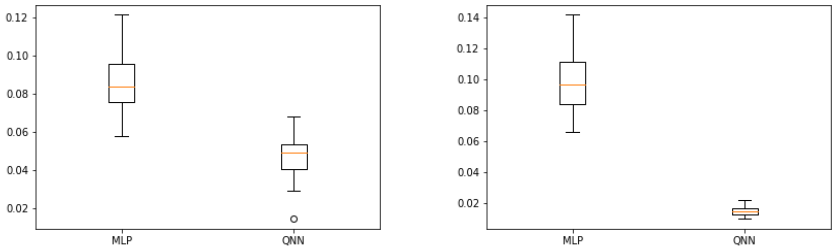
Fig. 7: Time Series datasets

Table 1 summarizes the results of the experiments performed. Column 1 prints the metric under study, Columns 2-3 the values of the corresponding metric for the **Laser** dataset obtained by MLP and QNN, respectively, and Columns 4-5 the values for the **Henon** dataset. On the other hand, rows 2-4 show the MSE obtained for the training, test and validation sets; and the rows 5-6 the minimum and maximum MSE obtained in the 30 experiments, respectively. Finally, the last row contains the average computational time in

seconds for each run. A Mann-Whitney U test was applied over the validation MSE of all 30 executions, and we remark results of QNN in row 3 with (+) if the test concluded that there are significant differences between MLP and QNN and the latter outperformed the former, which occurs in both cases. Boxplots in Figure 8 help to analyse these results.

Table 1: Summary of results.

Metric	**Laser**		**Henon**	
	MLP	**QNN**	**MLP**	**QNN**
Avg. Tr. MSE	0.0380	0.0168	0.0761	0.0044
Avg. Ts. MSE	0.0478	0.0199	0.0734	0.0063
Avg. Val. MSE	0.0863	0.0476 (+)	0.0977	0.0145 (+)
Min. Val. MSE	0.05774	0.01461	0.0655	0.0096
Max. Val. MSE	0.1215	0.0683	0.1416	0.02145
Avg. Time	10.36	30.92	10.44	30.89



(a) Laser (b) Henon

Fig. 8: Boxplots of MSE for Laser and Henon datasets in our experiments

According to Table 1, the QNN was able to outperform the classic MLP in the both datasets studied, both in training/test and validation data splits. Also, both the best solution ans worst were better with QNN. However, the average time required to perform the experimentation is x3 times slower in QNN, which makes the model less scalable than the classical counterpart for a large number of qubits. This could be expected, since the experiments were performed over a QC simulation software instead than in a true quantum hardware. Boxplots in Figure 8 support this conclusion, and also suggest that the robutsnetss of QNN is better than in MLP since the difference betweeen the first and third quartiles are lower in the former case.

If we analyze the results in terms of model complexity (number of parameters), the MLP with 10 hidden neurons contains (T+1)*10 parameters in the

hidden layer (weights and biases) and 11 parameters in the output layer. On the other hand, the number of parameters of the QNN is 2*T in the input layer (input scale and bias), 3*T parameters in the hidden layer, and 2 parameters (scale and bias) in the output layer. In the case of T=7, the MLP contains 91 parameters while the QNN has 37. Thus, the network model in the quantum proposal is significantly smaller than the classic counterpart. This fact suggests that QML can contribute not only with an improvement in accuracy of models, but also in model complexity.

Despite of these results, it is important to note that the computational space of a QNN increases by power of two everytime a qubit is included into the network. For that reason, we believe that the proposed model cannot be used under simulation for large time series. This fact opens the doors to future works where a more efficient way to embed time series could be analyzed to reduce the quantum network size.

5 Conclusions

In this work we have studied how Quantum Neural Networks can be used to perform Time Series forecasting tasks in quantum computers. We have designed a quantum neural network composed of a tensor product encoding input layer, and one or several hidden layers using the Real Amplitudes antsatz. Experiments were conducted as a proof-of-concept over two time series. Results suggest that QNN have a great potential improving accuracy being compared with the MLP classical counterpart, although at a big cost of computational resources required under simulations. Future works must be conducted to extend the analysis to larger time series and to design more efficient ways of Quantum Embedding that allow to reduce the QNN size.

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

References

1. Andrés, E., Cuéllar, M.P., Navarro, G.: On the use of quantum reinforcement learning in energy-efficiency scenarios. Energies **15**(16) (2022). https://doi.org/10.3390/en15166034
2. Azevedo, C.R.B., Ferreira, T.A.E.: Time series forecasting with qubit neural networks. In: Proceedings of The Eleventh IASTED International Conference on Artificial Intelligence and Soft Computing. p. 13–18. ASC '07, ACTA Press, USA (2007)
3. Bryan, L., Stefan, Z.: Time-series forecasting with deep learning: a survey. Philosophical Transactions of the Royal Society A **379**, 20200209 (2021). https://doi.org/http://doi.org/10.1098/rsta.2020.0209

4. Ciliberto, C., Herbster, M., Ialongo, A., Pontil, M., Rocchetto, A., Severini, S., Wossnig, L.: Quantum machine learning: A classical perspective. Proceedings of the Royal Society A: Mathematical, Physical and Engineering Science **474** (07 2017). https://doi.org/10.1098/rspa.2017.0551
5. Ganguly, S.: Quantum Machine Learning: An applied approach (2021)
6. Jerbi, S., Gyurik, C., Marshall, S., Briegel, H., Dunjko, V.: Parametrized quantum policies for reinforcement learning. In: Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., Vaughan, J.W. (eds.) Advances in Neural Information Processing Systems. vol. 34, pp. 28362–28375. Curran Associates, Inc. (2021)
7. Kutvonen, A., Fujii, K., Sagawa, T.: Optimizing a quantum reservoir computer for time series prediction. Nature scientific reports **10**, 14687 (2020). https://doi.org/10.1038/s41598-020-71673-9
8. Kwak, Y., Yun, W.J., Jung, S., Kim, J.: Quantum neural networks: Concepts, applications, and challenges. In: 2021 Twelfth International Conference on Ubiquitous and Future Networks (ICUFN). pp. 413–416 (2021). https://doi.org/10.1109/ICUFN49451.2021.9528698
9. Li, X., Cheng, C.T., Wang, W.C., Yang, F.Y.: A study on sunspot number time series prediction using quantum neural networks. In: 2008 Second International Conference on Genetic and Evolutionary Computing. pp. 480–483 (2008). https://doi.org/10.1109/WGEC.2008.76
10. Norlén, H.: Quantum Computing in practice with Qiskit and IBM Quantum Experience. Packt (2020)
11. Schuld, M., Petruccione, F.: Supervised Learning with Quantum Computers. Springer Publishing Company, Incorporated, 1st edn. (2018)
12. Senekane, M.: Hands-on Quantum information processing with Python. Packt (2021)
13. Shrivastava, P., Soni, K.K., Rasool, A.: Classical equivalent quantum unsupervised learning algorithms. Procedia Computer Science **167**, 1849–1860 (2020). https://doi.org/https://doi.org/10.1016/j.procs.2020.03.204, international Conference on Computational Intelligence and Data Science
14. Singh, P.: Fqtsfm: A fuzzy-quantum time series forecasting model. Information Sciences **566**, 57–79 (2021). https://doi.org/https://doi.org/10.1016/j.ins.2021.02.024
15. Sutor, R.: Dancing with Qubits. Packt (11 2019)
16. Torres, J.F., Hadjout, D., Sebaa, A., Martínez-Álvarez, F., Troncoso, A.: Deep learning for time series forecasting: A survey. Big Data **9**(1), 3–21 (2021). https://doi.org/10.1089/big.2020.0159
17. Wittek, P.: Quantum Machine Learning: What Quantum Computing means to data mining. Elsevier (2014)