Parameter Prediction for Variational Quantum Algorithms through Sequence Modeling*

Corrado Loglisci*, Vito Losavio, Berardina De Carolis, Marjana Skenduli and Donato Malerba

Department of Computer Science, Universita' degli Studi di Bari 'Aldo Moro', Bari, Italy

Abstract

NISQ devices represent a major step in quantum computing, though they face limitations such as short coherence times and error rates. Variational Quantum Algorithms (VQAs) help overcome these limitations by optimizing parameterized quantum circuits, but challenges like the barren plateau problem and the time spent in iterative procedures remain.

Predictive machine learning can improve VQAs by providing efficient parameter initialization and estimation. However, traditional ML models for parameter prediction assume static data and batch processing, which is problematic in dynamic environments. Sequential ML models address this by adapting the predictor to time-varying data and capturing correlations over sequences of data, improving the reliability of predictions of the parameters over time and reducing the time consumption that a conventional VQA would require.

Keywords

Variational quantum algorithms, Parameterized quantum circuits, Parameter estimation, Sequence modeling,

1. Introduction

In the rapidly advancing field of quantum technologies and quantum information processing, Noisy Intermediate-Scale Quantum (NISQ) devices are emerging as a key area of focus, having demonstrated quantum supremacy in a limited number of applications [1]. Although NISQ devices are not yet capable of performing fully error-corrected quantum computations, they mark a significant milestone in the development of quantum technology. Their limitations, such as short coherence times and higher error rates, have led to the creation of specialized algorithms like Variational Quantum Algorithms (VQAs). These algorithms are designed to be resilient against the imperfections of NISQ hardware. The interplay between the physical constraints of NISQ devices and the innovative design of VQAs represents a crucial area of research in quantum computing. VQAs are hybrid quantum-classical algorithms that incorporate parameterized quantum circuits (PQCs) alongside classical optimization techniques. PQCs consist of single- or multi-qubit gates characterized by tunable parameters, which are adjusted by classical optimizers in a manner similar to the training of neural networks.

The effectiveness and efficiency of these algorithms are highly dependent on how their parameters are initialized and optimized. For example, it has been shown that Parameterized Quantum Circuits (PQCs) can suffer from the barren plateau problem, where gradients vanish exponentially, leading to a flat optimization landscape. In such cases, optimizers struggle to improve the circuit parameters, making the training process extremely difficult. Moreover, this lack of efficiency is exacerbated by the limited access to real quantum computers. In practice, quantum hardware is not widely or readily available, as access is typically managed by large corporations, leading to long wait times and high costs for circuit execution and compilation [2]. Given these challenges, it is clear that designing effective strategies for parameter initialization and optimization is crucial. Starting the optimization process from a well-informed point in the parameter space, rather than from a random or uniform initialization

*AIQ*QIA 2024 2nd International Workshop on AI for Quantum and Quantum for AI*

*You can use this document as the template for preparing your publication. We recommend using the latest version of the ceurart style.

*Corresponding author.

corrado.loglisci@uniba.it (C. Loglisci)

, can significantly improve the performance of the VQAs by guiding them towards viable subspaces and increasing the likelihood of successfully navigating the optimization landscape.

To achieve that, it is important to rely on computational solutions capable of making non-intuitive decisions, exploring complex high-dimensional spaces, and identifying patterns and correlations that are not immediately obvious. It is also desirable that these solutions are robust to noise and errors while reducing the time and resources needed for empirical trial-and-error methods [3]. This is where machine learning (ML) algorithms can play a decisive role. ML models can systematically provide parameter estimations, efficiently search for optimal parameters, and offer substantial improvements in the execution of quantum algorithms.

The problem has only recently received significant attention from the research community. Most efforts have focused on using predictive ML models to initialize the parameters of quantum-classical optimizers, such as the Quantum Approximate Optimization Algorithm (QAOA) [4, 2]. These approaches typically rely on batch-style computation, where the data to train the model are all available at one time and remain unchanged throughout the process. The predicted parameter values for the optimizer reflect this assumption of data stability. However, in many real-world applications, data are not available all at once, and especially in dynamic and time-varying environments, data arrive in sequence, their properties may change [5, 6, 7] or may have some form of dependency or correlation with those of the data that have previously arrived [8]. So, the reliability of an ML model trained on the data available at the moment diminishes when it is later used on newly arrived data. To address these issues, time-variability of data properties and correlation between data points should be explicitly considered when training the model. Sequential ML models offer a solution by building predictors that account for the structure of sequential relationships between data, allowing to capture patterns and correlations over time or in ordered data [9].

2. Contribution

In this work, we explore a paradigm shift in the design of VQAs by studying how ML can enhance the construction of VQA models on sequential data. Specifically, we study how predictive ML techniques can assist conventional optimizers in forecasting the parameter values of parameterized quantum circuits. The goal is to help optimizers to achieve convergence or provide optimal values with fewer iterations compared to using optimization alone. This benefit becomes even more significant when conventional optimizers are applied to sequential data, as they would need to be executed from scratch repeatedly, on newly acquired data, leading to time-consuming calculations. At each execution, the optimizer might even face the Barren plateau problem.

Our proposal stems from recognizing the intrinsic property of the correlation in sequential data. Addressing that it is crucial not only for designing the appropriate machine learning algorithm to train the forecasting model but also for improving the accuracy of the predicted parameters. This correlation typically manifests in two forms: *temporal locality* and *temporal recurrence* [10, 11]. Temporal locality suggests that recent data is more strongly correlated or predictive of future data compared to older data. Temporal recurrence refers to the regular repetition of certain patterns or sequences over time. While temporal locality emphasizes short-term influence, temporal recurrence focuses on patterns that reappear at regular intervals over longer periods. Consequently, this correlation is mirrored in the behavior of the optimizing function during exploration of the parameter space. Therefore, such function replicates its response when these recurring patterns in the data emerge. Based on this, we propose modeling the behavior of the optimizer based on the parameter values it generates while minimizing the loss function, rather than focusing directly on the input data. The loss function in this work measures the performance of a VQA, specifically in a binary classification task, by quantifying the error between the class provided by the VQA and actual label. We conjecture that training the forecasting model using the parameter values obtained during loss minimization could serve as a reliable strategy, offering valuable guidance to the optimizer for the future classification runs over sequential data.

3. Method

To combine both forms of correlation in a machine learning (ML) model capable of forecasting the parameters for future optimizer runs, we propose a computational solution based on sequence modeling neural networks for time-series forecasting, as illustrated in Figure 1. This solution is structured around two interrelated methods.

The first method addresses the binary classification task handled by a Variational Quantum Algorithm (VQA) and operates directly on sequential data. The data arrive at regular intervals but in a scattered manner, without a predefined order between labeled and unlabeled instances. As the data are received, they are collected into a fixed-size data container, storing labeled data and unlabeled data. Using this container helps alleviate the processing load caused by the data arrival rate. So, the first method is able to train the binary classifier (through the VQA) from the labelled data and assigned either labels to unlabelled data.

During training, the VQA assigns the values predicted by the second method (forecasting model) to its parameters. Then, it tries to refine them along a pre-defined number (I) of iterations on the labelled data. More precisely, we implemented two variants: the first one (afterwards, $For + It$) initializes the parameters based on the forecasting model and then refines them over I iterations, the second one (afterwards, $It + For + It$) runs a number of $\frac{I}{2}$ iterations of the optimizer, then, calls for the forecasting model, and, finally, once again runs a number of $\frac{I}{2}$ iterations. The binary classifier so learned is therefore applied to the unlabeled data.

As an initialization for the entire process, we consider a number (D) of training sessions on the labelled data accumulated in the container. Each initial training session runs for J optimizer iterations, where J is chosen to be greater than I in order to build reliable parameter values, which will be used to feed the forecasting model. Specifically, we chose K iterations (out of J) in which the loss function of the classifier (first method) is minimized. The rationale is to forecasting model learn the capabilities of the optimizing function in the minimization of the objective function. Eventually, we obtain K parameter configurations from each of the D training sessions, each configuration presents a number T of values equal to the parameterized gates of the VQA.

The forecasting model consists of T LSTM components, each assigned to one parameter and trained on the values generated during the training sessions. Each LSTM processes subsequences extracted from an overall sequence of $D \times K$ time steps long (K time steps for each of the D training sessions). Each subsequence contains M time steps, and the LSTM predics a single numeric values within the range [0;2π], based on the previous M time steps. Subsequences are formed by shifting the previous subsequence by one time step. The LSTM learns by minimizing the prediction error over these shifted subsequences.

So, given a sequence $\langle x_1, x_2, \ldots, x_{D\times K}\rangle$ representing the entire time span from the training sessions, we extract overlapping subsequences of length M by shifting each subsequence by one time step:

$$\begin{aligned}
&\langle x_1, x_2, \ldots, x_M\rangle \rightarrow x_{M+1},\\
&\langle x_2, x_3, \ldots, x_{M+1}\rangle \rightarrow x_{M+2}, \ldots,\\
&\langle x_{(D\times K)-M}, \ldots, x_{(D\times K)-1}\rangle \rightarrow x_{(D\times K)}.
\end{aligned}$$

The LSTM is trained by minimizing the mean squared error (MSE), between the predicted values $\hat{y}_{s,t}$ and the true values $x_{s,t+M}$:

$$\frac{1}{D}\sum_{s=1}^{D}\sum_{t=1}^{K-M}\left(\hat{y}_{s,t} - x_{s,t+M}\right)^2$$

Generally, speaking a LSTM neural network consists of units each comprising three kinds of gates, *forget* gate, *input* gate, and *output* gate and keeps two model snapshots updated over processing time stepped data, *hidden state* and *cell state*. Forget gate filters out irrelevant parts of the previous cell state. Input gate decides what new information will be added to the cell state. Output gate determines the next hidden state based on the updated cell state. The cell state represents the long-term memory

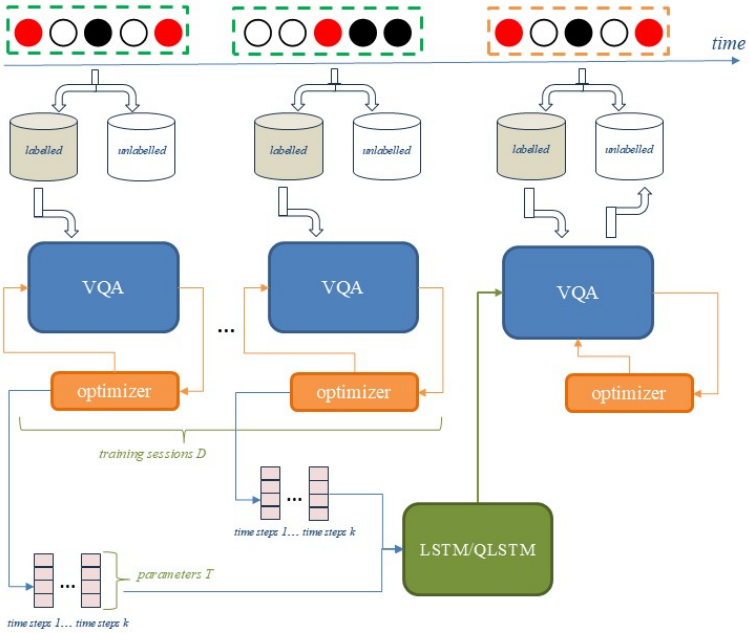


Figure 1: A graphical representation of the proposed computational solution. Two processes are carried out, one involving VQA executions over training sessions, the other one involving the parameter forecasting upon those produced by the VQA.

and is updated by combining the forget gate and input gate information. The hidden state represents the short-term memory, which is used to make predictions. They operate at each time-step as here formulated:

- Forget gate: $f_t = \sigma\left(x_t W_f + h_{t-1} U_f + b_f\right)$
- Input gate: $i_t = \sigma\left(x_t W_i + h_{t-1} U_i + b_i\right)$
- Cell state update: $c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t$ where $\odot$ denotes element-wise multiplication, $\tilde{c}_t = \tanh(W_c x_t + U_c h_{t-1} + b_c)$, h_{t-1} is the previous hidden state vector
- Output gate: $o_t = \sigma\left(x_t W_o + h_{t-1} U_o + b_o\right)$, with hidden state h_t updated as: $h_t = o_t \odot \tanh(c_t)$.

where,

- d is the number of units used in the whole model
- σ is the sigmoid activation function $\sigma(z) = \frac{1}{1+e^{-z}}$
- $\tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$ is the hyperbolic tangent activation function
- $f_t, i_t, o_t \in (0,1)^d$, $\tilde{c}_t, h_t (-1,1)^d$, $c_t \in \mathbb{R}^d$
- $W_f, W_i, W_o, W_c \in \mathbb{R}^{d \times 1}$ are weight matrices on the input values
- $U_f, U_i, U_o, U_c \in \mathbb{R}^{d \times d}$ are weight matrices on the hidden state
- $b_f, b_i, b_o, b_c \in \mathbb{R}^d$ are bias vectors learned during training.

After processing M time steps, the LSTM predicts the next numeric value $\hat{y} \in [0; 2\pi]$. This is done by applying a linear transformation to the hidden state at the last time step $t = M + 1$

$$\hat{y}_{M+1} = h_M W_y + b_y \tag{1}$$

where $W_y \in \mathbb{R}^{d \times 1}$ and $b_y \in \mathbb{R}^d$ are the output weight matrix and bias, respectively.

4. Experiments

The proposed solution has been implemented using IBM Qiskit [12], and is designed to work across a broad range of VQAs without loss of generality. For the current experiments, we used the quantum

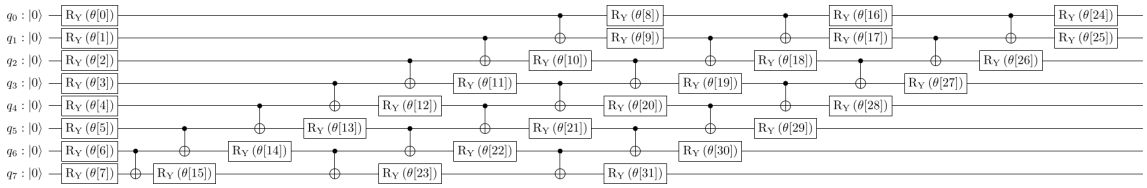


Figure 2: A graphical representation of the considered VQA. It presents 32 parameters distributed uniformly on 8 qubits.

circuit shown in Fig. 2, which is commonly used as an ansatz in quantum machine learning approaches. This circuit consists of 8 qubits and an equal number of gates per qubit. The design ensures that the parameterized gates are influenced not only by the encoding of classical data but also by two-qubit operations. The circuit includes 32 parameters to be optimized, corresponding to a forecasting model comprising 32 LSTMs.

The VQA leverages the pre-existing COBYLA optimizer for parameter tuning. For the forecasting model, we tested two implementations: a classical version using a standard LSTM [13] (referred to as LSTM) and a hybrid quantum-classical implementation [14] (referred to as QLSTM), both working on the gradient descent-based ADAM optimization algorithm. We conducted experiments using the *Aer* simulator on the real-world *Spambase* dataset, which contains 4600 data instances and 57 features[1]. A classical feature selection technique was applied to reduce the feature set to eight.

We simulated the scenario of incrementally incoming data by arranging the dataset in equally-sized data blocks, which were acquired one a time and stored in the container. Each data block consisted of 75% labelled data and 25% unlabelled data. We considered three different sizes of data blocks 20, 40, 60, corresponding to 230, 115, 77 data instances respectively. The number of initial training sessions D was chosen as the portion of 25% of all data blocks, consequently we have D to 4, 8, 12. We defined K (number of time-steps considered where the objective function takes lower values) as 4. The number M of time steps was chosen as {2,3,4} for D=4, {5,6,7,8} for D=8, {9,10,11,12} for D=12. Longer subsequences (higher M) are obtained with higher values of D. Orthogonally, the number of iterations J was set to 20, while the number of the iterations I is 8 when J is 20. To clarify, for the setting D=4 M=4 $For + It$ J=20, we trained the binary classifier over 4 training sessions each using 75% of 230 labelled data, the VQA works on 20 iterations during each training session, the forecasting model learns on subsequences of 5 (M+1) time steps, extracted from an overall sequence of 4×4 ($D \times K$) time-steps long. Once the D data blocks were processed, for the subsequent data blocks, the parameters of the VQA are either i) initialized using the forecasting model and refined through 8 iterations in case of $For + It$, or tuned over 4 iterations, whose K values feed the forecasting model and finally another refinement round of 4 iterations, in case of $It + For + It$.

Experiments were conducted to evaluate the accuracy in a binary classification task and measure the running times for different lengths, M, of the training sub-sequences. Table 1 presents the F1-scores, calculated as the average across individual values derived from the data blocks. The results show that the introduction of a forecasting model generally improves the predictive accuracy of binary classification for all values of M, except for M=10 (in bold).

It is important to note that not all predictors perform consistently. Specifically, the model **LSTM** *For+It* I=8 shows improvements for M values in the range {2, 3, 4, 5, 6, 7, 8}, whereas the model **QLSTM** *For+It* I=8 performs better for longer sub-sequences, specifically {9,11,12}. This suggests that the classical LSTM model performs well on short to medium-length sub-sequences, while the quantum LSTM model is more effective for those longer.

Another observation concerns the model variants *For+It* and *It+For+It*, which define how the VQA leverages the forecasting results. In all trials, the models using *It+For+It* consistently performed worse than their counterparts, both for LSTM and QLSTM. This underperformance may be attributed to the initial $\frac{I}{2}$ iterations, which, starting from a uniform setting, do not provide sufficient input for the

[1]https://archive.ics.uci.edu/ml/datasets/Spambase

	M		
	2	**3**	**4**
VQA	76,17(6,34)	76,17 (6,34)	76,17 6,34
LSTM $For + It$ I**=8**	**78,26** (6,46)	**78,53** (6,63)	**78,17** (6,36)
LSTM $It + For + It$ I**=8**	62,59 (5,68)	62,59 (5,68)	62,59 (5,68)
QLSTM $For + It$ I**=8**	63,13 (18,41)	66,44 (5,99)	68,17 (6,31)
QLSTM $It + For + It$ I**=8**	55,34 (27,81)	62,59 (5,68)	62,59 (5,68)

	M			
	5	**6**	**7**	**8**
VQA	74,64 (10,07)	74,64 (10,07)	74,64 (10,07)	74,64 (10,07)
LSTM $For + It$ I**=8**	**76,25** (7,55)	**75,8** (7,91)	**76,33** (8,31)	**77,59** (7,98)
LSTM $It + For + It$ I**=8**	63,3 (13,78)	68,21 (10,04)	70,8 (10,38)	70,8 (10,38)
QLSTM $For + It$ I**=8**	65,44 (19,46)	70 (13,88)	73,92 (26,83)	62,5 (19,33)
QLSTM $It + For + It$ I**=8**	67,23 (26,05)	65,44 (21,76)	70,8 (10,38)	70,8 (10,38)

	M			
	9	**10**	**11**	**12**
VQA	74,91 (11,54)	**74,91** (11,54)	74,91 (11,54)	74,91 (11,54)
LSTM $For + It$ I**=8**	71,78 (8,85)	67,65 (19,19)	60,85 (17,77)	58,48 (22,16)
LSTM $It + For + It$ I**=8**	72,22 (10,6)	67,05 (22)	59,51 (14,06)	59,51 (14,06)
QLSTM $For + It$ I**=8**	**75,89** (9,19)	73,37 (8,8)	**75,95** (8,73)	**75,29** (10,02)
QLSTM $It + For + It$ I**=8**	75,49 (8,74)	74,5 (10,57)	59,51 (14,06)	59,51 (14,06)

Table 1
Average and standard deviation of the F1-score computed for proposed method, with either LSTM or QLSTM, against a canonical approach with VQA.

forecasting model to enhance performance.

In Figure 3, we see the running times of three main approaches executed on a simulator average at different values of M (length of the subsequences). The predictors require more times only at the beginning, corresponding to the training sessions D. Contrarily, they ask for less time consumption for all the remaining data blocks, while saving time calculation by several hundreds. We observe that this happens with both LSTM models and QLSTM models.

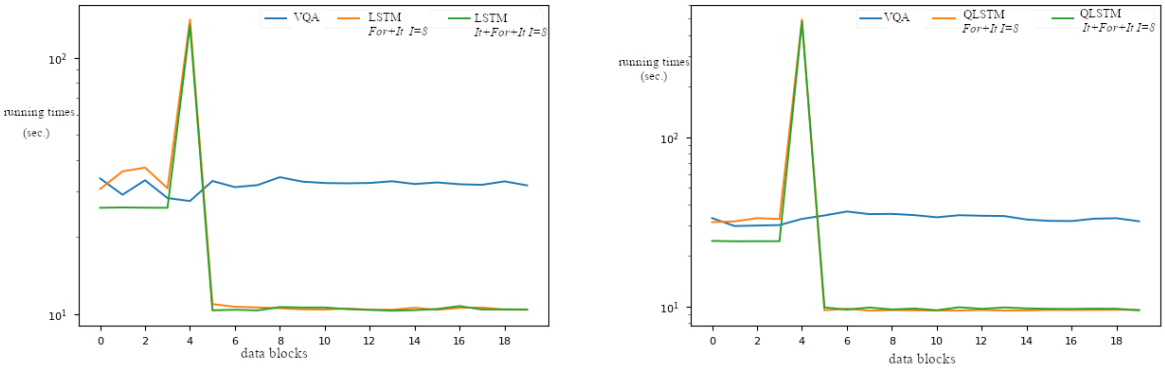
5. Conclusions

Alleviating the workload of VQAs is a challenging and attractive research area aimed at developing solutions that enable a broader range of practical applications demonstrating quantum advantages, including quantum machine learning. In this work, we explore a new perspective in VQA design, where the estimation of parameter values—traditionally handled using optimization techniques—can be supported by predictive machine learning. The usefulness of this approach becomes even more evident when the VQA is required to operate repeatedly, as in the case of continuous data flows. This work focuses on simulation-based systems, without considering noise-related issues.

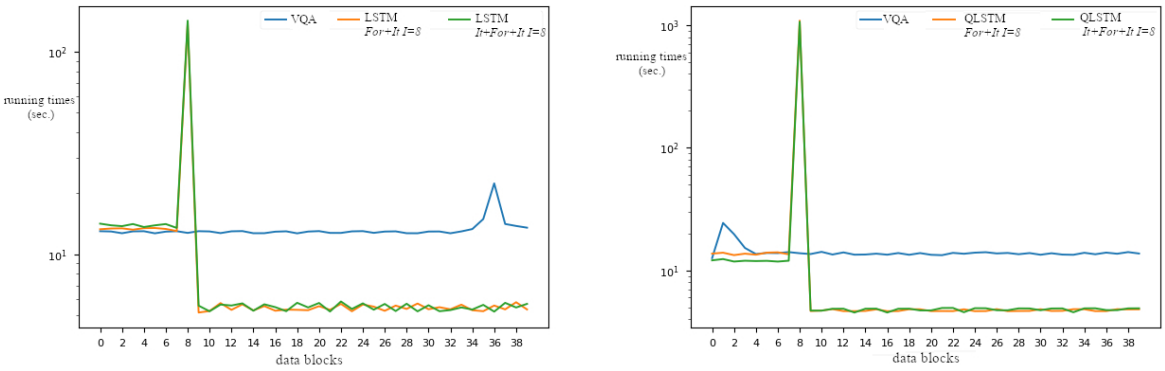
While our experiments are limited to a single case study and are not exhaustive, the results are encouraging. They suggest that integrating forecasting models into VQAs improves both the quality and effectiveness of the algorithms. However, not all predictors and experimental configurations guarantee this improvement, a topic we will further investigate. For instance, we plan to study the interactions between parameters in multivariate time-series data, such as with LSTM or QLSTM models.

Acknowledgment

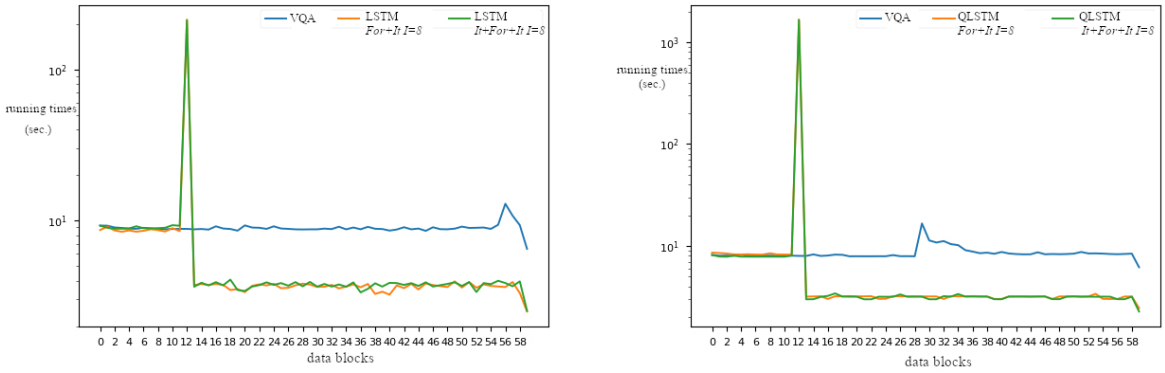
Corrado Loglisci and Marjana Skenduli acknowledge the financial support from the project "PNRR



(a) $M = 4$



(b) $M = 8$



(c) $M = 12$

Figure 3: Running times computed for proposed method, with either LSTM or QLSTM, against a canonical approach with VQA.

MUR project PE0000023-NQSTI" for this research.

References

[1] L. Madsen, F. Laudenbach, M. Askarani, F. Rortais, T. Vincent, J. Bulmer, F. Miatto, L. Neuhaus, L. Helt, M. Collins, A. Lita, T. Gerrits, S. W. Nam, V. Vaidya, M. Menotti, I. Dhand, Z. Vernon, N. Quesada, J. Lavoie, Quantum computational advantage with a programmable photonic processor (2022). doi:https://doi.org/10.1038/s41586-022-04725-x.

[2] Z. Liang, G. Liu, Z. Liu, J. Cheng, T. Hao, K. Liu, H. Ren, Z. Song, J. Liu, F. Ye, et al., Graph

learning for parameter prediction of quantum approximate optimization algorithm, arXiv preprint arXiv:2403.03310 (2024).
[3] J. Falla, Q. Langfitt, Y. Alexeev, I. Safro, Graph representation learning for parameter transferability in quantum approximate optimization algorithm, Quantum Machine Intelligence 6 (2024) 46.
[4] F. Meng, X. Zhou, Parameter generation of quantum approximate optimization algorithm with diffusion model, arXiv preprint arXiv:2407.12242 (2024).
[5] C. Loglisci, D. Malerba, S. Pascazio, Quarta: quantum supervised and unsupervised learning for binary classification in domain-incremental learning, Quantum Mach. Intell. 6 (2024) 68. doi:10.1007/s42484-024-00196-7.
[6] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, S. Wermter, Continual lifelong learning with neural networks: A review, Neural Networks 113 (2019) 54–71. doi:10.1016/j.neunet.2019.01.012.
[7] C. Loglisci, D. Malerba, Coupling quantum classification and quantum distance estimation in continual learning, in: AIQxQIA@AI*IA, volume 3586 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2023.
[8] H. Homayouni, S. Ghosh, I. Ray, S. Gondalia, J. Duggan, M. G. Kahn, An autocorrelation-based lstm-autoencoder for anomaly detection on time-series data, in: 2020 IEEE International Conference on Big Data (Big Data), 2020, pp. 5068–5077. doi:10.1109/BigData50022.2020.9378192.
[9] Z. Xie, Y. Yang, Y. Zhang, J. Wang, S. Du, Deep learning on multi-view sequential data: a survey, Artif. Intell. Rev. 56 (2023) 6661–6704.
[10] R. A. Rossi, Relational time series forecasting, The Knowledge Engineering Review 33 (2018) e1. doi:10.1017/S0269888918000024.
[11] C. Loglisci, I. Diliso, D. Malerba, A hybrid quantum-classical framework for binary classification in online learning, in: SEBD, volume 3478 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2023, pp. 88–99.
[12] M. S. Anis, et al, Qiskit: An open-source framework for quantum computing, 2021. doi:10.5281/zenodo.2573505.
[13] M. Abadi, et al., Tensorflow: Large-scale machine learning on heterogeneous systems, https://www.tensorflow.org/, 2015.
[14] S. Y.-C. Chen, S. Yoo, Y.-L. L. Fang, Quantum long short-term memory, 2020. URL: https://arxiv.org/abs/2009.01783. arXiv:2009.01783.