A Reproduction and Discussion of "Exploring Hybrid Quantum-Classical Methods for Practical Time-Series Forecasting"

Cihan Alperen Bosnali, Dante Prins, Elio Di Nino, Richard Sun
Department of Electrical and Computer Engineering
The University of British Columbia

I. Introduction

Time-series forecasting is a tool used to predict future data points based on past observations and drive decisions based on the results. Univariate time-series forecasting is specifically predicting based on the historical data of a single variable [1] and is the focus of this report. To perform accurate predictions, time-series forecasting models learn patterns from historical data. A variety of classical time-series forecasting models are utilized by researchers in different fields. Some examples are the linear regression model and the neural network model [2].

There are also quantum computing methods that can be utilized for this task, two of which are investigated by Dimitrijevs et al. In the paper "Exploring Hybrid Quantum-Classical Methods for Practical Time-Series Forecasting." Dimitrijevs et al. attempt to investigate two questions related to the use of quantum computers for time-series forecasting:

1) Can we use Noisy Intermediate-Scale Quantum (NISQ) era [3] quantum computers for time-series forecasting?
2) How effective are the hybrid quantum-classical methods compared to classical methods for time-series forecasting?

Two hybrid methods are employed: Parametrized Quantum Circuit (PQC) and Variational Quantum Linear Solver (VQLS) [2] to investigate these questions.

II. Methods

A. Linear Regression

Linear regression solves the equation $Xw = y$ by maximizing some metric of accuracy, where X is an $n \times d$ matrix representing the n examples and y is an $n \times 1$ vector representing the predictions for each example [4]. Note that in general the model has d arbitrary features, but in time-series forecasting specifically, the features are a sliding window of the d data points prior to the current point [2]. The vector w is what linear regression solves for, and it represents the weight assigned to each feature with size $d \times 1$. Dimitrijevs et al. use a sliding window of size 12 for their results, so we did the same.

In order to construct and train a linear regression model, we need to define a loss function. The common choice and the one chosen by Dimitrijevs et al. is to use the linear least squares objective function, which is defined as follows [4]:

$$f(w) = \frac{1}{2}\sum_{i=1}^{n}(wx_i - y_i)^2 = \frac{1}{2}\|Xw - y\|^2 \tag{1}$$

In simple terms, this function reduces the sum of all the squared differences between each predicted value and its real value. Since we want to minimize this function, we can take the gradient with respect to w and find the global minimum by setting it equal to 0 [4]:

$$\begin{aligned} \nabla f(w) &= X^T Xw - X^T y \\ 0 &= X^T Xw - X^T y \\ X^T Xw &= X^T y \end{aligned} \tag{2}$$

This results in a set of d linear equations with d unknowns which we can directly solve with linear algebra [4]. Additionally, a y-intercept can be added such that the line of best fit does not need to pass through the origin. This is achieved by adding a feature with value 1 for every example and does not change how the w vector is found [5].

Linear regression can be expanded upon with the use of regularization terms. This is not something Dimitrijevs et al. detail outside of saying "no explicit regularization was applied to any of the models" and that "proper regularization would likely lead to good test set MSE scores" [2]. The two primary forms of regularization are L1- and L2-regularization which add $\lambda||w||_1$ and $\frac{\lambda}{2}||w||^2$ terms to the loss function (1), respectively. The benefit of L2-regularization is that favours making the values of w smaller for a slight increase in training error, but generally a lower test error [6]. L1-regularization operates similarly, but favours sparse solutions (where multiple weights are set to 0), which has the effect of selecting only the most important features. The disadvantage of L1-regularization is that it is non-differentiable so we must use an approximation of the L1-norm in combination with gradient descent to solve the equation whereas L2-regularization is still differentiable and allows for direct solving like before [7]. In both cases, λ can be varied to change the strength of the regularization and is often picked through cross-validation [6], like we did in our implementation.

B. Neural Network

Deep learning is used extensively for time-series forecasting, however, there are drawbacks since deep learning

models require large amounts of data and lack explainability of their trained parameters [8]. While advanced deep learning architectures such as transformers are the focus of the latest research in the field [8], this paper uses a simple neural network architecture as a baseline for comparison against their quantum-classical methods [2]. Dimitrijevs et al. argue that their neural network architecture is selected "to mirror the limitations of the currently available quantum devices" [2], but they do not provide an explanation of how they compared quantum and classical resources to arrive at this conclusion.

Dimitrijevs et al. provide the following information on their neural network: "two hidden layers, each with 12 units and ReLU activation" [2]. We used this information to implement the neural network with the Keras framework. They do not state the loss function, optimizer, batch size or the number of epochs used for neural network training. We decided to use Mean Squared Error (MSE) as our loss function since Dimitrijevs et al. chose it as their "primary evaluation metric used across all models" [2]. We arbitrarily chose the optimizer to be the Adam optimizer, batch size to be 8, and the number of epochs to be 300 after testing.

C. Parametrized Quantum Circuit

The Parametrized Quantum Circuit method, abbreviated as PQC, encodes model information and the solution ansatz as rotation gates on a input state [9], for example $|0\rangle^n$.

Dimitrijevs et al. encode a 12 wire circuit with an adjustable ansatz to make a time-series forecast prediction. Each of the wires are encoded with the previous time-series values using the ansatz gates, forming a feature window of size 12.

The PQC implementation in this paper has two main components, as seen in Fig. 1:

1) *Feature Map*: Consists of 1 layer of RY gates where the feature values are encoded as angles (ϕ). This is possible due to data normalization and scaling that was done during pre-processing.
2) *Parameterized Ansatz*: 2 layers of RX, RZ, and CNOT gates. The angles passed into the RX and RZ gates (θ) are tuned by an optimizer to minimize the cost function. The CNOT gates are to entangle the qubits of each wire so that when we take measurements on a wire 0, its results are the combination of all 12 features.

The cost function of this method is MSE, where error is defined as the difference between the expectation value of Z on the wire 0 and the actual forecast label (3). Notably, the paper left out the exact basis and wire we should take the expectation value of. We take the expectation value of Z on wire 0 as a prediction because it has a range of $[-1, 1]$ when measuring in the computational basis, which falls within the valid prediction range.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\langle Z_0\rangle - y_i) \tag{3}$$

Here (3), n denotes the total number of samples, and i is the incrementing index of each sample.

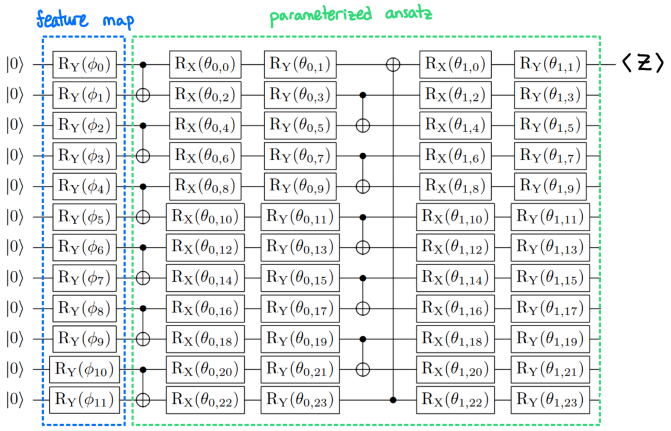

Fig. 1. PQC Diagram

When evaluating this method, two different optimizers were used:

1) *L-BFGS-B*: a gradient-based method optimized to be space-efficient for problems with large number of parameters. This is achieved by not storing the entire Hessian matrix of the gradient, but instead only a few vectors.
2) *COBLYA*: a non-gradient-based method using linear approximation to adjust the parameters.

D. Variational Quantum Linear Solver

A Variational Quantum Linear Solver or VQLS is a quantum solver [10] with a more rigorous derivation compared to the PQC ansatz.

We begin with the solution found earlier for the least-squares gradient (2), defining $M = X^T X$ and $b = X^T y$. Now the problem is redefined as $Mw = b$, which has an exact solution if M is full rank.

Rather than directly solving the system of equations like in linear regression, we can convert this to a quantum optimization problem, by encoding both $|b\rangle$ and $|w\rangle$ as quantum states using amplitude embedding [11], and satisfying the equation $M|w\rangle = \lambda|b\rangle$, where λ accounts for the natural normalization of the quantum states. We may arrive at the solution $|w(\phi)\rangle$ using a cost function for variational optimization of ϕ. The authors [2] propose the cost function:

$$C(\phi) = 1 - |\langle b|M|w(\phi)\rangle|^2 \tag{4}$$

The premise is simple, for normalized quantum states, the maximum expectation value of $|\langle\gamma|\delta\rangle| \leqslant 1$ reaches 1 when $|\delta\rangle = |\gamma\rangle$. Therefore, our optimizer will minimize the cost using (4) when $|w(\phi)\rangle$ satisfies $M|w(\phi)\rangle = |b\rangle$.

However, this assumption is violated for two reasons. Firstly, the absolute value of $|\langle b|M|w\rangle|$ is equal to that of $|\langle b|M|(-|w\rangle)|$, but the state $-|w\rangle$ produces the wrong sign to solve $Mw = b$. To our knowledge, this mistake also appears in other documentation of VQLS [10]. The second violation is regarding the non-unitarity of M. Because $M = X^T X$ is Hermitian but not unitary, the state $M|w(\phi)\rangle$ will not

necessarily remain normalized. Therfore, the inner product $\langle b|M|w(\phi)\rangle$ in (4) has no obligation to be maximized when $M|w\rangle = \lambda|b\rangle$, so optimizing the cost function will not solve $Mw = b$. We may correct these mistakes with the modified cost function:

$$C(\phi) = 1 - \frac{\langle b|M|w(\phi)\rangle}{\|M|w(\phi)\rangle\|} \tag{5}$$

We see both the sign ambiguity and normalization mistakes have been corrected with this new cost function. The non-absolute value of the expectation may appear worrying due to the complex value of general quantum states. However, because amplitude embedding [11] produces quantum states with only $\pm$ phase, and the M matrix is necessarily real, no complex phase is ever introduced. In simulation, a real value filter can be applied to the cost function for the small imaginary component arising from computation imprecision.

Once $|w\rangle$ is obtained, the vector which solves $Mw = b$ is found by de-normalizing $w = \|w\| \cdot |w\rangle$. This is done like so:

$$\begin{aligned} Aw &= b \\ \|w\| \cdot A|w\rangle &= \|b\| \cdot |b\rangle \\ \|w\| \cdot \|A|w\rangle\| &= \|b\| \cdot \||b\rangle\| \\ \|w\| \cdot \|A|w\rangle\| &= \|b\| \\ \|w\| &= \frac{\|b\|}{\|A|w\rangle\|} \end{aligned}$$

It then follows that $w = \frac{\|b\|}{\|A|w\rangle\|}|w\rangle$.

Quantum computers operate on the application of unitary operators, however, and as previously mentioned, M is only Hermitian. Therefore, $\langle b|M|w(\phi)\rangle$ is not directly computable! To overcome this, we must decompose M in terms of unitaries, where we choose a Pauli basis, applying one Pauli gate to each qubit. For n qubits there are 4^n Pauli combinations:

$$\begin{aligned} M &= \sum_{i=0}^{4^n-1} \alpha_i M_i \\ M_i &= \bigotimes_{j=0}^{n-1} \sigma_{[i]_4^j} \end{aligned}$$

Where $\alpha_i = \frac{Tr(M_i M)}{2^n}$, is the coefficient or "overlap" of each Pauli decomposition basis matrix for M and $\bigotimes$ indicates the tensor product between each qubit's gate. The Pauli matrices are defined as usual, with $\sigma_{4n+0,1,2,3} = \sigma_{0,1,2,3}$. Finally $[i]_4^j$ indicates the j-th digit of i, encoded in base-4. This accomplishes an encoding of M_i with one Pauli for each qubit. For example, M_{12} for two qubits would be $\sigma_3 \otimes \sigma_0$ or $\sigma_Z \otimes I$.

Using this decomposition, and defining the amplitude embedding unitaries, $B|0\rangle = |b\rangle, W|0\rangle = |w\rangle$, we may compute the expectation of M using unitaries with (6).

It should also be noted that the authors of [2] incorrectly implied $B^\dagger$ as B in their work. Whether this was a publication mistake or an additional error in their simulation technique is unknown.

$$\langle b|M|w\rangle = \sum_{i=0}^{4^n-1} \alpha_i \langle b|M_i|w\rangle = \sum_{i=0}^{4^n-1} \alpha_i \langle 0|B^\dagger M_i W|0\rangle \tag{6}$$

The unitary measurement is accomplished using the Hadamard Test [12], in which the expectation $Re(\langle 0|U|0\rangle)$ is obtained with the circuit shown in Fig. 2.

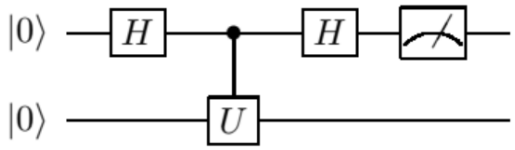

Fig. 2. Circuit diagram of the Hadamard Test, with the outcome measurement expectation on the first wire being $Re(\langle 0|U|0\rangle)$. Adapted from [2].

With this construction, we may use a variation solver to optimize the cost function (5). This obtains the unique solution to $Mw = b$, identical to the outcome of (2), which is the least-squares solution to $Xw = y$.

III. Results

A. Linear Regression

Dimitrijevs et al. present their linear regression results in Fig. 3. Notice that they chose to predict far into the future, but it is not clear whether they padded their feature windows with 0s at the end, if they predicted on top of predicted values, or did something else to achieve this. Regardless of which it was, the results are not meaningful and have no reference value, so we chose not to extend our results beyond the available data for all of our models.

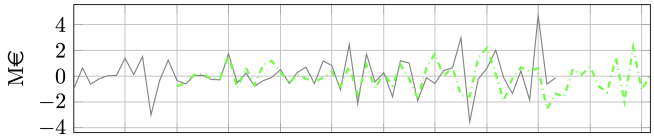

Fig. 3. Linear Regression Results from Dimitrijevs et al.

As part of the our linear regression implementation, we parameterized options for adding regularization and/or a y-intercept. When we initially compared our results with no regularization and no y-intercept to Dimitrijevs et al.'s results, we could see minor differences. However, when comparing our model with a y-intercept and no regularization, that is when we ended up with an identical result. Our suspicion is that Dimitrijevs et al. used a software library such as scikit-learn like we did, but did not realize that it adds a y-intercept by default so they failed to mention this in their paper. Our linear regression results are presented in Fig. 4.

Additionally, we trained models with both L1- and L2-regularization to see how they compared, which Dimitrijevs et al. did not do. These can be seen in Fig. 5 and 6, respectively. As discussed in Linear Regression, we can see that both L1- and L2-regularization favour lower weights resulting in flatter slopes, with L1-regularization going so far as to set all the

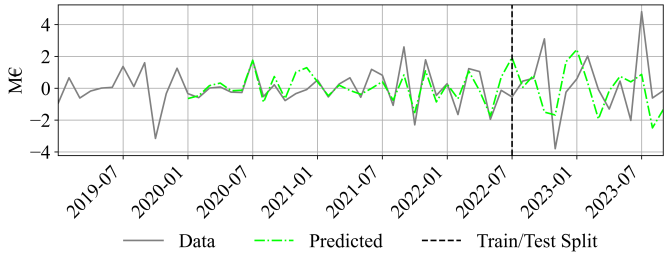

Fig. 4. Our Linear Regression (with Y-Intercept) Results

weights to 0, an indication that no trends could be found. Further discussion of this may be found in the Conclusion.

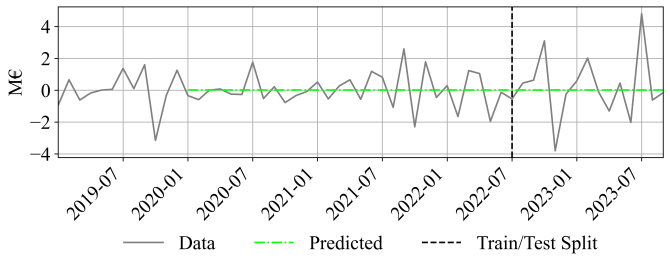

Fig. 5. Our L1-Regularized Linear Regression Results

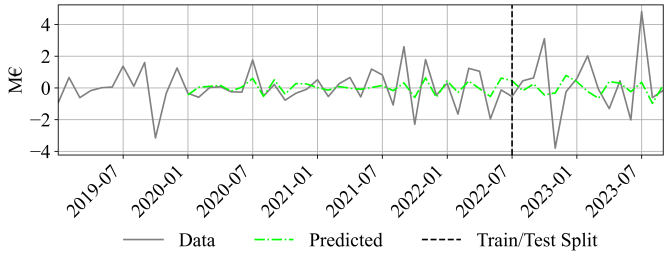

Fig. 6. Our L2-Regularized Linear Regression Results

B. Neural Network

Dimitrijevs et al. report the predictions of their neural network implementation on the dataset as shown in Fig. 7. We have achieved results extremely close to Dimitrijevs et al. as shown in Fig. 8. Specifically, we let the neural network overfit as was done in the paper.

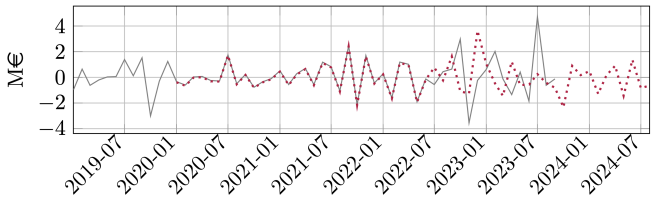

Fig. 7. Neural Network Results from Dimitrijevs et al.

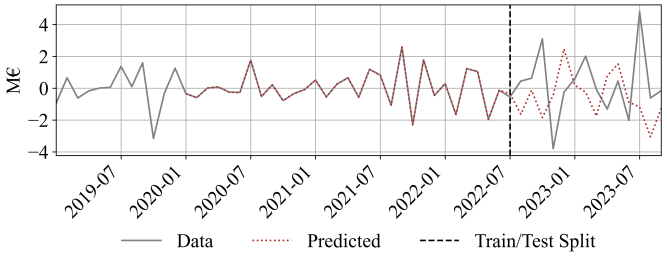

Fig. 8. Our Neural Network Results

C. Parametrized Quantum Circuit

Dimitrijevs et al. report time series forecasting results using PQC optimized with L-BFGS-B (Fig. 9) and COBYLA (Fig. 10). In our work, we implemented both optimization techniques, producing results that are comparable to those reported by Dimitrijevs et al. (see Fig. 11 for L-BFGS-B and Fig. 12 for COBYLA).

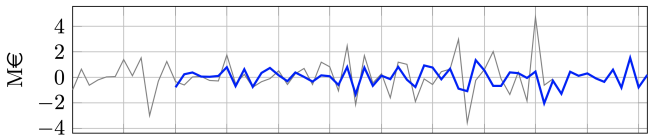

Fig. 9. PQC with L-BFGS-B Optimizer Results from Dimitrijevs et al.

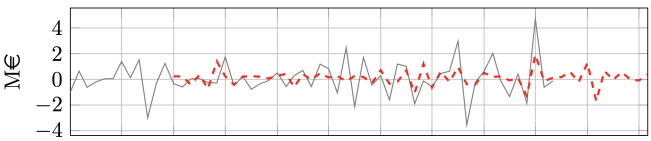

Fig. 10. PQC with COBYLA Optimizer Results from Dimitrijevs et al.

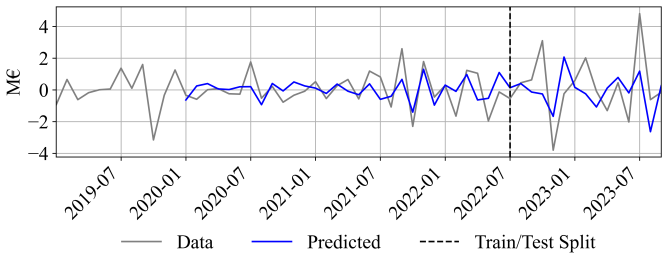

Fig. 11. Our PQC with L-BFGS-B Optimizer Results

Note that two sets of graphs do not show an exact correspondence. Although the L-BFGS-B algorithm is expected to yield a unique solution, the absence of the original dataset may introduce minor discrepancies in the computed results. This is reflected in the graphs (Fig. 9 and Fig. 11), which are largely consistent but not entirely identical. In contrast, the COBYLA algorithm, being non-gradient-based and inherently sensitive to initialization, demonstrates greater variability. In particular, Dimitrijevs et al. did not specify the initialization

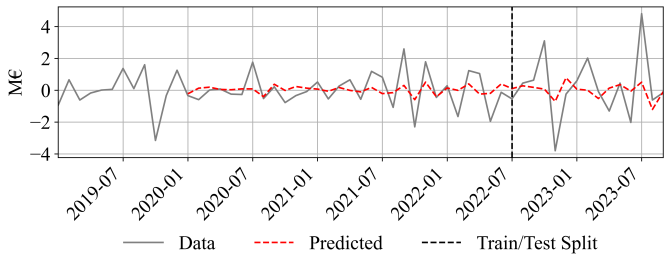

Fig. 12. Our PQC with COBYLA Optimizer Results

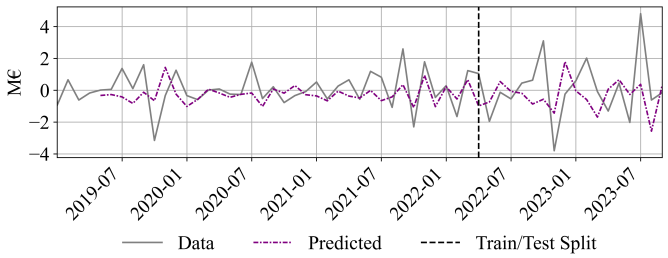

Fig. 13. Our VQLS Results

details, contributing to the large variations between the plots (Fig. 10 and Fig. 12). Our own experiments further confirmed that each execution of the COBYLA algorithm produced distinct outcomes, reinforcing the understanding that it does not guarantee a unique solution.

D. Variational Quantum Linear Solver

The authors [2], report VQLS producing "predictions [that] were indistinguishable from random values. It was also the case when [they] tested [the] approach on real data from Force AI." As mentioned above, VQLS achieves identical solutions to that of linear regression, where with proper execution of a variational solver, the solution $Mw = b$ is obtained. The authors appear oblivious of that fact, as they reported prediction results for linear regression, but claim the VQLS results were unsatisfactory. We hypothesize that they incorrectly inferred VQLS was producing poor time series forecasting, whereas the mistake was an incorrect implementation.

Regardless, we performed time-series forecasting using the same dataset to verify the correctness of our VQLS implementation as well to accomplish the author's intention to study its performance. We insist that this is redundant because the technique produces the same outcome as linear regression. This is re-iterated with the identical results of Fig. 13 and 14. The window size of these simulations was reduced to four, because the slow simulation speed of VQLS in the classical Pennylane simulator. No solution was reached in 24 hours of simulation time for 3- or 4-qubit simulations. This is a key drawback of VQLS, where the amount of unitary expectations to compute scales with 4^n qubits. This is coupled to a further slowdown as each simulation is performed with more quibts. Given these scaling challenges, it is difficult to see this technique becoming an attractive alternative to linear regression, which itself has many optimization methods [13]. The 2^n quantum state encoding density is overshadowed by the 4^n simulation requirement along with the much higher cost of quantum resources for the foreseeable future.

E. Loss Iterations

Dimitrijevs et al. report loss iterations for each method in Fig. 15. Our results in Fig. 16 were hard to analyze due to the large variance of PQC with COBYLA. As a result we have the same results with a log scale in Fig. 17 and without COBYLA entirely in Fig. 18. In both the paper's and our results, the final training loss follows the same ordering (from

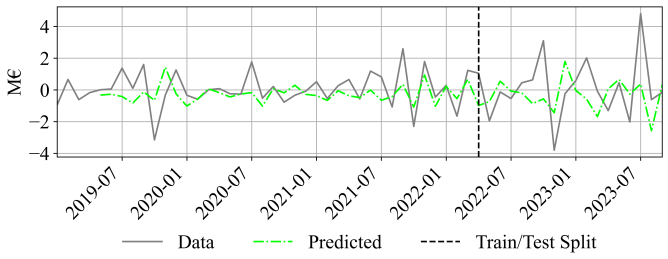

Fig. 14. Our Linear Regression Model with Equivalent Window Size

lowest to highest): Neural Network, Linear Regression, L-BFGS-B, and COBYLA. Note that VQLS's loss is included but it was trained with a smaller training window so its loss is not entirely comparable. Additionally, L-BFGS-B exhibits an initial exponential loss decrease in both plots, though it appears less prominent in ours due to the broader scale. COBYLA displays a different loss trajectory, likely due to sensitivity to initialization. Lastly, unlike Dimitrijevs et al., our plots extend the final loss value beyond the stopping point rather than truncating the curve.

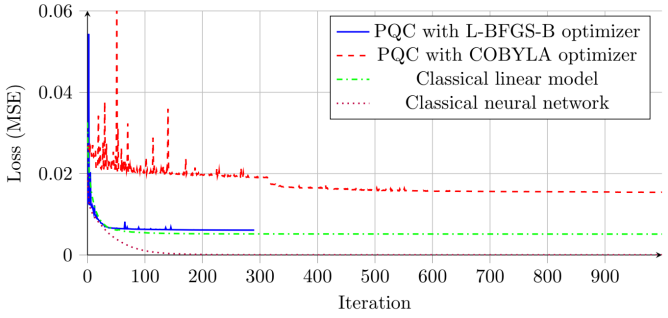

Fig. 15. Loss over Iterations from Dimitrijevs et al.

IV. Conclusion

Dimitrijevs et al. attempt to investigate the viability of quantum-classical models for time-series forecasting. They perform experiments with PQC and VQLS methods and compare them against classical linear regression and neural network methods.

After a brief inspection of their dataset, we can already conclude that Dimitrijevs et al. are in no position to make

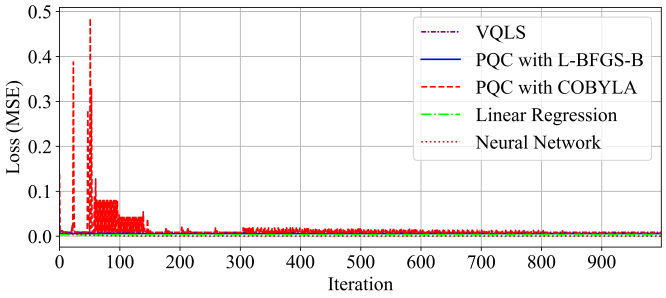

Fig. 16. Our Loss over Iterations

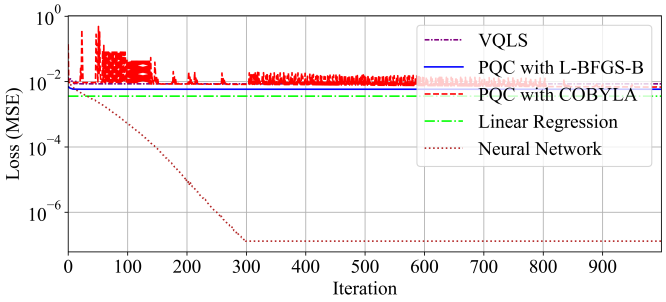

Fig. 17. Our Loss over Iterations (Log Scale)

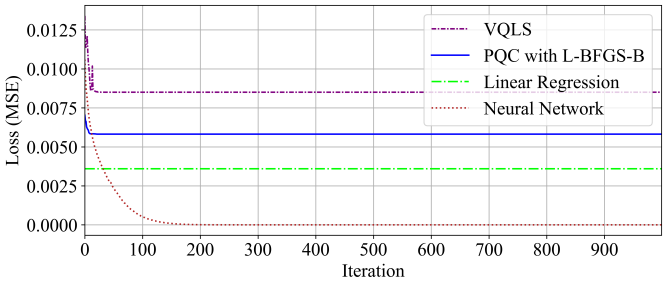

Fig. 18. Our Loss over Iterations (without COBYLA)

conclusions about the performance of quantum-classical models. With only 58 data points, their dataset does not contain nearly enough data to include meaningful patterns that can then be trained on [1]. To this point, none of the prediction models presented succeeded in remotely predicting the test data accurately. This is a failure of the dataset, not their choice of forecasters. To build on this point, our L1- and L2- results (Fig. 5 and 6) show how picking a flat line with this dataset produces the best test MSE because there are no extractable trends in the limited data.

Their work is not relevant to the current discourse on the topic of time-series forecasting as no advantage is shown against classical models, and furthermore, the datasets are insufficient to perform any meaningful comparison. The work has little relevance to the quantum computing field, as PQC being used as a time-series forecasting model was not properly studied given their insufficient dataset. The authors also incorrectly derived and implemented Variational Quantum Linear Solver. As we have shown analytically and through data (Fig. 13), a correct VQLS implementation will produce identical results to linear regression. Hence, we can claim VQLS is a valid time-series forecasting method, although its inefficiency compared to classical linear regression is highlighted.

While the paper's results are replicable, Dimitrijevs et al.'s choice of dataset along with an erroneous VQLS implementation invalidate all claims they make on the suitability of quantum-classical methods for time-series forecasting. We recommend a new study be conducted with a much larger and more robust dataset. Furthermore, given that the outcomes of VQLS are equivalent to linear regression, that study may be conducted without any VQLS simulation. A more detailed resource estimation and scaling derivation should also be performed to estimate the regime in which PQC and VQLS are efficient alternatives to the classical techniques studied in this work.

References

[1] Chris Chatfield. *Time-Series Forecasting*. New York, NY: Chapman and Hall/CRC, 2000, p. 280. DOI: 10.1201/9781420036206.

[2] Maksims Dimitrijevs, Mārtiņš Kālis, and Iļja Repko. *Exploring Hybrid Quantum-Classical Methods for Practical Time-Series Forecasting*. 2024. arXiv: 2412.05615 [quant-ph]. URL: https://arxiv.org/abs/2412.05615.

[3] John Preskill. "Quantum Computing in the NISQ era and beyond". In: *Quantum* 2 (Aug. 2018), p. 79. ISSN: 2521-327X. DOI: 10.22331/q-2018-08-06-79. URL: https://doi.org/10.22331/q-2018-08-06-79.

[4] Jiarui Ding and Mi Jung Park. *Linear Regression [Lecture Notes]. CPSC 340/540 - Machine Learning and Data Mining (Spring 2025)*. Jan. 2025. URL: https://www.students.cs.ubc.ca/~cs-340/slides/L11.pdf.

[5] Jiarui Ding and Mi Jung Park. *Nonlinear Regression [Lecture Notes]. CPSC 340/540 - Machine Learning and Data Mining (Spring 2025)*. Jan. 2025. URL: https://www.students.cs.ubc.ca/~cs-340/slides/L12.pdf.

[6] Jiarui Ding and Mi Jung Park. *Regularization [Lecture Notes]. CPSC 340/540 - Machine Learning and Data Mining (Spring 2025)*. Feb. 2025. URL: https://www.students.cs.ubc.ca/~cs-340/slides/L16.pdf.

[7] Jiarui Ding and Mi Jung Park. *More Regularization [Lecture Notes]. CPSC 340/540 - Machine Learning and Data Mining (Spring 2025)*. Feb. 2025. URL: https://www.students.cs.ubc.ca/~cs-340/slides/L17.pdf.

[8] John A. Miller et al. *A Survey of Deep Learning and Foundation Models for Time Series Forecasting*. 2024. arXiv: 2401.13912 [cs.LG]. URL: https://arxiv.org/abs/2401.13912.

[9] Marcello Benedetti et al. "Parameterized quantum circuits as machine learning models". In: *Quantum Science and Technology* 4.4 (Nov. 2019), p. 043001. ISSN: 2058-9565. DOI: 10.1088/2058-9565/ab4eb5.

[10] Andrea Mari. *Variational Quantum Linear Solver*. URL: https://pennylane.ai/qml/demos/tutorial_vqls.

[11] PennyLane Documentation. *Amplitude Embedding*. URL: https://docs.pennylane.ai/en/stable/code/api/pennylane.AmplitudeEmbedding.html.

[12] Michael Nielsen and Isaac Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2010. DOI: 10.1017/CBO9780511976667.

[13] John P. Cunningham and Zoubin Ghahramani. "Linear Dimensionality Reduction: Survey, Insights, and Generalizations". In: *Journal of Machine Learning Research* 16.89 (2015), pp. 2859–2900. URL: http://jmlr.org/papers/v16/cunningham15a.html.