Optimising Quantum Calculations Reliability via Machine Learning: The IBM Case Study

1st Zakaria Abdelmoiz Dahi
Univ. Lille, Inria, CNRS
Centrale Lille, UMR 9189 CRIStAL
F-59000 Lille, France
abdelmoiz-zakaria.dahi@{inria,univ-lille}.fr

2nd Francisco Chicano
ITIS Software
University of Malaga
Malaga, Spain
chicano@uma.es

3rd Gabriel Luque
ITIS Software
University of Malaga
Malaga, Spain
gluque@uma.es

4th Iván Delgado Alba
ITIS Software
University of Malaga
Malaga, Spain
ivandelgadoalba@uma.es

***Abstract*—The current quantum technology depends on hardware/time-dependent features that render quantum calculations highly error-prone. Having faulty calculations generally requires redoing them, which induces an economic/computational overhead by using quantum machines/simulators. This can be problematic due to unaffordable fees, machine unavailability, or loss of quantum advantage. An alternative can be performing the calculations when the machines are in their most suitable state. This work presents a pipeline based on Machine Learning (ML), allowing users to choose the appropriate moment to perform a given computation based on the estimation of the Jensen-Shannon divergence between the noisy and ideal distributions of quantum sampling. This includes (I) an extract-transform-load data module, (II) an ML unit for quantum features forecasting and error prediction, and (III) a web-based visualisation unit. The pipeline was built/tested using 3.5 months of calibration data from three real 127-qubit IBM quantum machines. The results confirmed the applicability of the proposal in realistic scenarios.**

***Index Terms*—Quantum Computing, Machine Learning**

I. Introduction

Quantum Computing (QC) is a paradigm based on quantum mechanical properties, which are thought to provide an acceleration of the calculation compared to the classical paradigm. Eventually, this might allow making leaps towards advances in several domains, especially in optimisation and quantum simulation. Quantum calculation can be done exclusively on quantum machines or simulators. The simulation is time and computationally costly, where the memory and CPU requirements increase exponentially with the number of qubits. On the other hand, access to real quantum computers is subject to substantial fees and long queues. Actually, today's quantum technology is still in its Noisy Intermediate Scale Quantum era (NISQ), where the hardware is not yet stable and scalable enough, resulting in limited and, most importantly, error-prone computation [1].

Generally, in NISQ machines, two main factors rule the sensitivity of the calculations to errors: `(I)` the calculation complexity, and `(II)` the hardware features. The former, especially the noise-related ones, evolve over time according to several factors (e.g. environment and technology). So, depending on those factors, the noise will have a different impact on the reliability of the computation. In case the calculation error is not systematic, which is generally the case, one way of increasing the reliability of the calculation or correcting its associated error, although with no guarantee, is to repeat/perform calculations over different periods. This turns out to be problematic considering the limited and costly access to real quantum computers/simulators. A less brute-force alternative involves mitigating calculation errors either before or after performing the calculations. The feasibility of the second option is still challenging, considering the limited capacities of NISQ machines to implement efficient quantum error correction mechanisms [1]. Considering the pre-calculation error mitigation, a major approach is to design hardware-aware pre-calculation steps such as transpilation. The former adapts the quantum computation to fit within the hardware requirements. However, including noise-related aspects in the transpilation process makes it `(I)` slower to execute, which might erase any quantum acceleration achieved. `(II)` More difficult to handle, resulting, usually, in more complex quantum calculations (e.g. more gates to execute). Therefore, it does not provide any guarantee that the calculation will be 100% noise-free as the calculation robustness towards error generators (e.g. decoherence times) decreases proportionally to the calculation complexity.

This work focuses on pre-calculation error mitigation. To cope with the above-mentioned limitations, this proposal relies on the time evolution of the noise-related hardware features (e.g. qubits and gates) to mitigate the noise effects on the calculations' reliability. The idea is to execute the calculation during a time slot when the hardware is less error-prone. This has several advantages, such as `(I)` optimising the reliability of the computation, `(II)` making the transpilation process easier as the noise constraints are the weakest, `(III)` executing a more compact and efficient calculations, and `(IV)` removing the need for long/repetitive access/use to real quantum machine/simulators to repeat calculations with high computational and economic costs. Concretely, the contribution of the work stands in a complete machine learning-based pipeline to allow the users to choose the most appropriate time to execute, according to their preferences (e.g. tradeoff of error-robustness vs calculation time), their calculation with the least hardware noise. The former is based

on the estimation of the Jensen-Shannon divergence between the noisy and ideal distributions of quantum sampling. The pipeline is composed of three units: (I) the data module responsible for data extraction, processing, and persistence, (II) the forecasting and prediction module in charge of the forecasting of future machine features as well as the prediction of the calculations' errors. Finally, (III) the web-based visualisation module with easy hands-on the pipeline. The proposal was built and tested using data from three real 127-qubit IBM quantum machines, using 3.5 months of calibration data (from 27/02/2024 to 06/06/2024).

In the rest of the paper, Section II presents preliminary concepts of quantum computation/hardware and ML. Then, Section III introduces the proposal, Section IV evaluates its efficiency, and Section V concludes the paper.

II. Fundamental Concepts

This section presents the fundamentals of QC and ML.

A. Quantum Computing and Hardware

QC is based on quantum mechanical phenomena (e.g. entanglement and superposition), where the minimum piece of information is called a qubit, standing for Quantum Bit. The state of a qubit is represented by a vector in a complex-valued bi-dimensional vector space. The basis vectors are denoted with $|0\rangle$ and $|1\rangle$. An aribitrary quantum state has the form $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, where $\alpha, \beta \in \mathbb{C}$ and $|\alpha|^2$ and $|\beta|^2$ are the probability amplitudes of measuring 0 and 1, respectively. The quantum state of n qubits is represented by a vector in the tensor product $\mathcal{H}^{\otimes n}$ of the individual Hilbert vector spaces where each qubit's state evolves. Several quantum computing paradigms exist, although in this work we focus on the discrete gate-based one [2]. It describes quantum calculations as quantum circuits, which can be viewed as a series of quantum gates acting on a set of qubits. A quantum gate U is a unitary linear operator (matrix) that linearly maps a quantum state in a given vector space into another state within the same vector space. In that sense, a linear operator U acting over multiple vector spaces fulfills Equation (1).

$$U\big(\sum_i \alpha_i |\psi_i\rangle\big) = \sum_i \alpha_i U|\psi_i\rangle \tag{1}$$

Any unitary transformation U satisfies $UU^\dagger = U^\dagger U = I$, where I is the identity matrix and $U^\dagger$ is the conjugate transpose of U. Several single-qubit quantum transformations exist, such as the Hadamard gate, and two-qubit gates such as the CNOT (see Equations (2) and (3)) [3]. Some unitary transformations acting on n-qubit quantum systems can be represented by the Kronecker product of n unitary transformations, while some others cannot.

$$\texttt{H} = \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ 1/\sqrt{2} & -1/\sqrt{2} \end{bmatrix} \quad (2) \qquad \texttt{CNOT} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad (3)$$

Nowadays, quantum technology is still in an early stage of development, known as the NISQ era, where quantum machines have limited and noisy capacities, and where the hardware features have a direct impact on the feasibility and reliability of the calculation. Such features include the Quantum Processing Unit (QPU) topology, the set of implementable unitary transformations, and others. In this work, we focus on the IBM quantum machines, considering their wide use and open accessibility. More specifically, we emphasise the study of some qubit and gate features that are affected by noise [1]. These characteristics include gate features such as Gate_error probability and Gate_length (ns). Also, qubits features such as T1 (us), T2 (us), Frequency (GHz), Anharmonicity (GHz), Readout_error, Prob_meas0_prep1, Prob_meas1_prep0 probabilities, and Readout_length (ns).

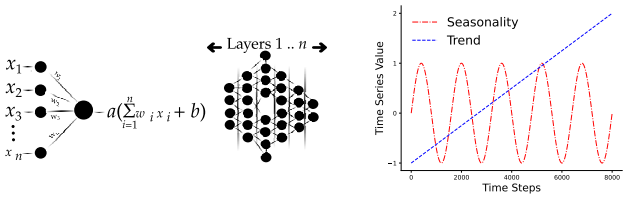
The gates' Error and Length express the probability of faultiness and duration of execution of the gates, respectively. T1 and T2 are the decoherence times, where T1 is the qubit relaxation time measuring how quickly a qubit in the excited state $|1\rangle$ spontaneously relaxes to the ground state $|0\rangle$. T2 is the dephasing time, measuring how fast a coherent superposition (one with a well-defined relative phase between the $|0\rangle$ and $|1\rangle$) loses this well-definedness. Essentially, it is what makes a qubit a quantum phenomenon, while a dephased qubit is just a probabilistic classical bit. IBM quantum machines are based on superconducting qubits, where qubits' interactions are performed via microwave resonators. To address each qubit separately, every qubit's resonator has a different frequency, which is represented by the Frequency feature. Regarding the Anharmonicity property, the used resonators are built to be anharmonic so that the difference of energy between the ground and the first excited state (i.e. $|0\rangle$ and $|1\rangle$) is different from the transition energy between the first and the second excited state. This allows treating it as a two-level quantum system to implement a qubit [4], [5]. The Readout_error is the probability that a measurement returns the wrong value. The Prob_meas0_prep1 and Prob_meas1_prep0 features are the probabilities that a measurement returns 0 or 1, immediately after preparing the $|1\rangle$ or the $|0\rangle$ states, respectively. Finally, the Readout_length is the time it takes to perform a measurement. This is one of the main limiting factors of current quantum hardware, because these times are often considerably longer than the T2, which makes intermediate measurements practically unfeasible. All these features evolve in time and differ according to the quantum machine being used and its surrounding environment.

B. Predictive vs Forecasting Machine Learning Models

Prediction-oriented machine learning requires data to build a function $f : X \mapsto \hat{Y}$ connecting the input data X to the predicted output $\hat{Y}$ [2]. Several models exist (e.g. XGBoost,

[1] IBM QPU Features: https://docs.quantum.ibm.com/guides/qpu-information

Multi-Layer Perceptrons (MLP), etc.), where the MLP is a neural network that stacks several layers of artificial neurons. The first layer receives the input data X, while each of the remaining layers receives the output of its previous one, up to the last that outputs the predicted data $\hat{Y}$. The input of a neuron is the weighted sum of the outputs of the neurons it is connected to from the previous layer, while its output is the activation function a of the input (see Figure 1(a)). This can be defined as $a(\sum_{i=1}^{n_{(j-1)}} w_i^{(j)} x_i^{(j-1)} + b)$, where $x_i^{(j-1)}$ and $n_{(j-1)}$ are the output of the i-th neuron and the number of neurons in the layer $j-1$, respectively. Model training aims at finding the weights $w_i^{(j)}$ that optimise a learning measure.



(a) (b)

Fig. 1. (a) MLP principles and (b) trend and seasonality

On the other hand, a time series is a sequence of time-ordered data, recorded at regular or irregular time intervals, and generally used to describe how a phenomenon evolves over time [6]. Considering this fact, time series forecasting is the process of forecasting future values based on the inherent properties and characteristic patterns found in historical data [7]. Forecasting techniques can be classified as parametric and non-parametric. The first class needs to capture the distribution of the data to build predictive models. This makes the model dependent on hyperparameters to optimise the forecasting results. The choice of the forecasting technique should be made according to the data features, such as the stationarity where the mean/variance are constant over the time laps. This includes trends with constant inc/decreasing behaviour. Another aspect to be considered is the cyclic patterns in the data, known as the seasonality (see Figure 1(b)). As shown in [8], non-parametric models such as Long Short Term Memory neural networks (LSTM) have been proven to outperform parametric ones (e.g. ARIMA and SARIMAX) under similar conditions. So, this work focuses on non-parametric models such as the LSTM, as well as hybrid parametric/non-parametric models such as Neural Prophet.

C. Noise-Aware Software for Error Mitigation

This work is not concerned with the field of error mitigation and correction techniques that act directly on the quantum calculation description. But, instead, it focuses on the approaches that attempt to do so without modifying the calculation or introducing any error correction mechanism. In this perspective, the execution process of quantum calculations is composed of several processes. Considering, for instance, the transpilation [9], which itself is composed of several phases (e.g. `CNOT` routing, pulse scheduling, circuit optimisation, etc.). So far, the main efforts to minimise the effects of both qubit and gate errors have been done by designing transpilation techniques that consider the hardware properties. This can be noted, for instance, in the advanced options of the pass manager in the IBM Qiskit transpiler (`https://docs.quantum.ibm.com/api/qiskit/transpiler`). This approach has three main drawbacks: `(I)` the transpilation algorithms that consider noise properties are more complex, which requires more time and computation efforts. `(II)` Including noise properties in the transpilation modelling makes it more difficult to perform it efficiently. This implies that `(III)` there is no guarantee of ensuring the minimum level of noise-mitigation. If the error is not systematic, a possible solution is to perform trial executions of the same calculations under different noise constraints until an acceptable error deviation is obtained.

Using NISQ machines for performing trials of calculations on real quantum computers or simulators is likely to have an expensive economic and computational cost. So, unlike the first approach that relies on trying to mitigate the noise effect by considering the hardware properties, another alternative would be to forecast the time when the quantum machine would have the best hardware features to perform a reliable calculation. In that sense, very little has been done. For instance, in [10], authors use a simple support vector machine and a simplistic/limited/small benchmark testbed (e.g. circuits and quantum machines) to classify quantum noise fingerprints. This poses several issues, as the proposal does not provide a complete pipeline like ours, the models used are not suitable for time series, and the testbed is too limited to provide a reliable application. The authors in [5] use machine learning models, principally recurrent neural networks, to predict the error induced by a given calculation. This work suffers from several shortcomings. It does not consider the quantum machine noise fingerprint in the prediction model. It proposes an incomplete ML pipeline with no ETL module and visualisation. Lastly, the study is performed on a limited set of benchmarks as well as very small-sized quantum machines.

III. The Proposed Approach

Unlike the previous works in [10] and [5], the proposed approach is designed for the third-largest IBM quantum machines nowadays, and a wide range of calculations. It includes a complete pipeline based on machine learning for quantum machine features *forecasting*, and also quantum calculation error *prediction*. This section introduces the proposed approach, including `(I)` data module for data extraction, transformation, and loading of real IBM quantum machine calibrations. `(II)` the machine learning unit including the *forecasting* and *prediction* models, and `(III)` the web-based visualisation unit (see Figure 2). The source code of the proposal is made publicly available at: `https://github.com/Zakaria-Dahi/ML-Assisted_Quantum_Calculation`.

A. The Data Unit: Extract, Transform and Load

The data unit extracts the data, processes it, and stores it. The quantum machine calibration extraction was done for the

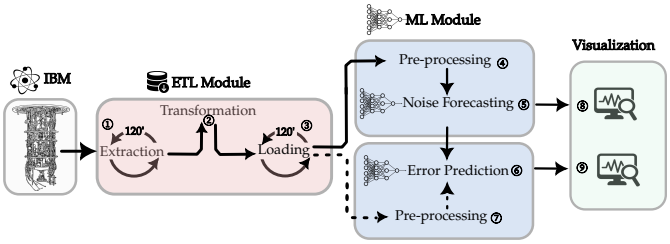


Fig. 2. The proposed quantum error-mitigation pipeline

three largest 127-qubit gate-based IBM quantum computers at the moment of performing the experiments, and the third largest machines nowadays: `IBM Brisbane`, `Kyoto`, and `Osaka`, although only the first one is still active. The data extraction unit extracts data every 60 minutes, processes it by saving it as tuples of {date, name, unit, value}. The data are stored locally as `JSON` files and remotely every 60 minutes on a public GitHub repository. This choice of storage format is thought to ease its use in both the ML and visualisation modules. Another reason is to ease replicability, comparisons with the present work, and their usage by other researchers and practitioners. In total, data has been extracted, transformed, and loaded for a period of 3.5 months (from 27/02/2024 to 06/06/2024). All the data is accessible in the above-given link.

The gathered data concerns both the gates' and qubits' calibration features described in Section II. However, the aim of this work is to design, for the first time in the literature, an ML pipeline for error mitigation based on the qubits' calibration. Thus, the transformation phase focused on the qubits' features only. Those features went through a statistical study to select the most relevant ones. In that sense, it was observed that features such as `Frequency`, `anharmonicity`, and `readout_length` were stable and constant through time. So, they have been discarded, and only the five remaining qubit features were considered: `T1, T2, prob_meas0_prep1, prob_meas1_prep0` and `readout_error`. This turns out to be quite relevant as these features symbolise the noise sensitivity of the qubits from different aspects.

B. The ML Unit: Forecasting and Prediction

The ML module includes two submodules. The first one is the *forecasting* module, which is in charge of forecasting the future values of the qubits' features. In this work, both the LSTM and Neural Prophet models have been used. A time window of 2 hours has been used to train those models. On the other hand, the *predictive* submodule has been used to predict the error produced when performing some quantum calculation using a given forecasted machine calibration. In this work, both MLP and XGBoost models have been used (see Section II-B). The goal of the predictive unit is to avoid the economic and computational overhead of executing the entire calculation on real quantum computers or simulators. In this work, the modelling of the task has been done by including both the qubits' calibration features and the description of the quantum calculation. For the qubit's calibration, five characteristics have been considered, which are the median values of the `T1, T2, prob_meas0_prep1, prob_meas1_prep0` and `readout_error` of all the qubits of the machine. On the other side, the quantum calculation has been represented by the number of `T, S, Hadamard` and `CNOT` gates necessary to describe the computation. The choice of these quantum gates is because they constitute a set of universal gates supported by IBM quantum machines and are sufficient to describe any quantum computation. All the features have been normalised using the Min-Max normalisation technique.

Going further into details, the *forecasting* model Neural Prophet is based on the Prophet model [11]. It includes a neural network that allows capturing more complex patterns in time series. Unlike autoregressive models, it does not make assumptions regarding the shape of the mapping function, which allows it to learn a nonlinear function to approximate continuous functions from the training data. On the other hand, its disadvantages are that it requires a large volume of data, hyperparameterisation is not straightforward, and it is less interpretable than parametric forecasting techniques.

Finally, the LSTM is a recurrent network thought to remember information over periods while resisting vanishing/exploding gradients. Unlike the recurrent neural networks that have one interacting layer in their repeating module, LSTM has four (see Equations (4)-(9)), where f_t, i_t, o_t and $\widetilde{c}_t$ are the activation vectors of forget, input, output, and cell input gates. c_t, h_t and x_t are the cell, hidden and input state vectors. Input and recurrent connections weights are stored in matrices of the form W_q and U_q. The σ_h and σ_c variables denote the hyperbolic tangent and sigmoid functions, respectively [12].

$$f_t = \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \quad (4) \qquad i_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \quad (5)$$

$$o_t = \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \quad (6) \qquad \widetilde{c}_t = \sigma_h(W_c x_t + U_c h_{t-1} + b_c) \quad (7)$$

$$c_t = f_t \circ c_{t-1} + i_t \circ \widetilde{c}_t \quad (8) \qquad h_t = o_t \circ \sigma_h(c_t) \quad (9)$$

C. The Visualisation Module: User Interface

The visualisation unit consists of a web application. The former is composed of several webpages that can be categorised in four classes: `(I)` the main page, `(II)` the error prediction, `(III)` qubits' calibrations forecasting, and `(IV)` the history of predictions and forecasting. On the principal webpage, all the functionalities are explained thoroughly. Afterwards, the second category allows predicting the error related to some given quantum calculation using some forecasted qubits' characteristics at a given date/time. The third class allows predicting the error of a given quantum calculation, given some qubits' calibration values introduced as a `JSON` file, and a description of the quantum calculation to be done. The last category of webpages allows visualising the history of the qubits' features, forecasting, and the calculation error predictions for a given quantum machine. All the webpages provide several

functionalities such as interactive filling, error messages in case of incorrect data, and drag-and-drop functionalities. For more details and real use, please consult the project repository in the previously given link.

IV. Experimental Analysis and Validation

This section presents the experiments performed to assess the feasibility and performance of the proposal, which include the experimental settings and the obtained results.

A. Settings and Benchmarks

The implementation was done using Python 3.9, running on a machine with Linux OS `Ubuntu 16.04.7 LTS (GNU/Linux 4.4.0-210-generic x86_64)`. The pipeline was designed and tested on the data of three 127-qubit IBM quantum computers: `Brisbane`, `Kyoto`, and `Osaka`. These were the largest IBM quantum computers when the study was conducted, and the third-largest IBM quantum computers nowadays. Quantum simulation using perfect and noisy machine models was done using Qiskit version `1.2.4`. The ML part has been implemented using `scikit-learn` and `Keras` libraries. The predictive and forecasting models of the ML unit have been executed with the following parameters: the MLP (Three layers: 64 neurons with a `relu` activation function, 32 neurons with a `relu` activation function, and 1 neuron with a linear activation function, `adam` optimizer, Mean Squared Error as the loss function, and Mean Absolute Error as an additional evaluation metric), XGBoost (Max depth: 6, learning rate: 0.1, `subsample`: 1, `colsample_bytree`: 1), Neural Prophet (Learning rate: 0.1, `n_forecasts`: 1), and the LSTM (100 neurons on the LSTM layer and 5 neurons on the dense layer). The number of trees, when applicable, is set to the default value used by the `scikit-learn` and `Keras` libraries.

The training dataset was built by randomly generating circuits with a depth and number of qubits equal to 5, 10, and 15. Also, we used circuits of 25%, 50%, and 75% of single-qubit gates, while the remaining gates were two-qubit gates. Using all the combinations of depth, number of qubits, and the percentage of single/two-qubit gates, a set of training circuits has been generated. Each of the calculations has been executed 5 times, and the arithmetic average of the Jensen-Shannon divergence has been computed to express the simulation error. This metric has been selected considering it is a well-established measure of difference between distributions. This fits both the ML models' purpose and the sampling-based quantum calculations. Also, unlike other metrics such as Kullback-Leibler, which is asymmetric, the Jensen-Shannon is symmetric. Actually, other QC works, such as the one devising the `TKet` quantum compiler [13], use the same experimental methodology as this present paper, and employ the same metric to assess it.

B. Results and Analysis

This section regroups the results of the three main groups of experiments performed: `(I)` data generation and training, `(II)` quantum machine calibration forecasting, `(III)` quantum error prediction.

1) Data Generation and Training: Table I shows the results of the data generation part of the proposed pipeline (see Section III-A), including the circuit configurations and the Jensen-Shannon divergence associated with it. The former has been used to represent the error (deviation) associated with a given circuit. To obtain it, first, the circuit is executed using a perfect simulator (i.e. no noise model used). Then, the same circuit is executed several times using a simulator with a noise model based on some extracted machine calibrations. Afterwards, the Jensen-Shannon divergence is computed using the first (perfect) and the remaining (noisy) executions.

TABLE I
Data generation: circuit features and error (i.e. Jensen-Shannon divergence)

Depth	Qubits	Gates	Brisbane	Kyoto	Osaka
5	5		0.0053	0.0063	0.0079
5	10		0.0070	0.0075	0.0083
5	15		0.0075	0.0083	0.0081
10	5		0.0357	0.0607	0.0556
10	10	25%	0.1508	0.1024	0.2181
10	15		0.2132	0.2439	0.2898
15	5		0.1641	0.2047	0.1845
15	10		0.3508	0.4819	0.4723
15	15		0.5792	0.6287	0.6566
5	5		0.0094	0.0046	0.0078
5	10		0.0077	0.0066	0.0093
5	15		0.0091	0.0081	0.0076
10	5		0.1124	0.0775	0.0987
10	10	50%	0.0894	0.1496	0.1753
10	15		0.2881	0.2418	0.2850
15	5		0.1297	0.1603	0.1733
15	10		0.5119	0.4344	0.3232
15	15		0.6646	0.6707	0.6358
5	5		0.0054	0.0064	0.0059
5	10		0.0074	0.0068	0.0091
5	15		0.0084	0.0086	0.0083
10	5		0.0499	0.0418	0.0630
10	10	75%	0.3019	0.1150	0.1215
10	15		0.2201	0.1969	0.2544
15	5		0.1610	0.2564	0.2204
15	10		0.6123	0.6363	0.4723
15	15		0.6442	0.5419	0.6358

It is worth noting that the divergence reported in Table I and Figures 3 and 4 is not obtained after training a model or whatsoever. It is a quantification of the deviation that can have a quantum calculation due to some hardware noise. As it can be seen in Table I and the Figures 3 and 4, one can observe that the circuit depth and the number of qubits influence the error induced. Also, one can note that the generated circuits have better error rates depending on the quantum machine being used, the depth, and the number of qubits. One can conclude that the increase in the complexity of the calculation, described here by the number of qubits used and the circuit depth, induces an increase in the error rate associated with the calculation. This emphasises the need for optimising quantum circuits to enhance their reliability and reduce the error associated with them, as well as the use of the most suitable machine to do so. At last, it is to be noted that, in the following sections, the generated dataset has been divided, randomly with no selection criteria, by dedicating 80% to training, and 20% to testing both the calibration forecasting and error prediction models. No validation phase/set has been used. Also, it is to be kept in mind that results in Table I,

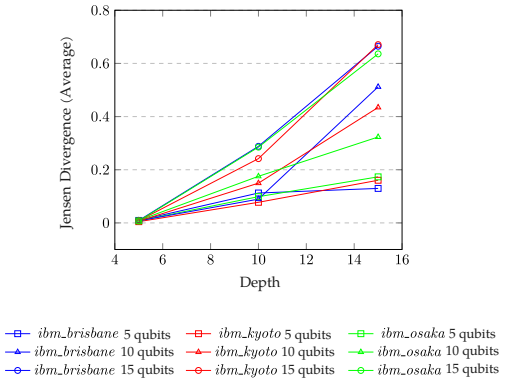


Fig. 3. *Jensen-Shannon* divergence vs depth: 50% of gates

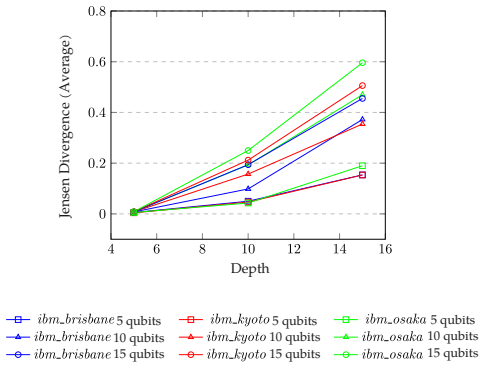


Fig. 4. *Jensen-Shannon* divergence vs depth: 75% of gates

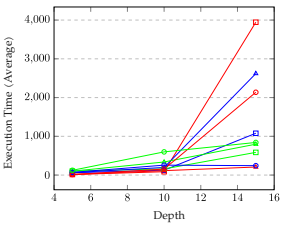


Fig. 5. 25%/75% 1/2-qubits gates

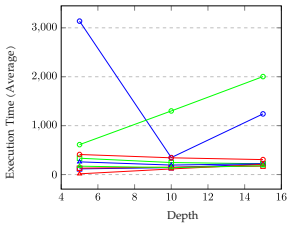


Fig. 6. 50%/50% 1/2-qubits gates

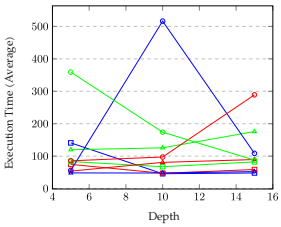


Fig. 7. 75%/25% 1/2-qubits gates

represents a snapshot of the generated data, where according to the calculation configuration (i.e. inputs) such as the circuit depth, number of qubits, and gates, a given quantum error (i.e. Jensen-Shannon divergence), due to machine noise, has been calculated for each of the studied quantum machines `Brisbane`, `Kyoto`, and `Osaka`.

TABLE II
AVERAGE RMSE ON TEST DATA: MODELS VS MACHINES

Model	Machine	RMSE
XGBoost	`Brisbane`	360.600
XGBoost	`Kyoto`	241.874
XGBoost	`Osaka`	270.868
Multilayer Perceptron	`Brisbane`	153.871
Multilayer Perceptron	`Kyoto`	867.243
Multilayer Perceptron	`Osaka`	371.319

Table II shows the results on the test data. Both the MLP neural network and XGBoost have been trained on the generated dataset (see Table I). The goal was to confirm the suitability of the model with respect to the data. As a training quality measure, the Root Mean Squared Error (RMSE) has been used (see Equation (10)), where y represents the real Jensen-Shannon divergence (e.g. the one reported in the Table I) that stands for the error associated with a quantum calculation done by a machine with a given calibration. On the other hand, $\hat{y}$ represents the predicted one by MLP or the XGBoost. Table II shows that the XGBoost is better than the MLP on all machines, except `Brisbane`. This atypical behavior goes along with the one detected in Figure 6. This outlines some peculiar hardware/calibration properties of the `Brisbane` machine.

$$\texttt{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y-\hat{y})^2} \tag{10}$$

TABLE III
EXECUTION TIME (SECONDS): `BRISBANE`, `KYOTO` AND `OSAKA`

Depth	Qubits	Gates	`Brisbane`	`Kyoto`	`Osaka`
5	5		60.502	14.581	104.265
5	10		61.674	14.660	100.275
5	15		70.142	13.966	123.202
10	5		125.679	86.479	158.783
10	10	25%	196.408	114.289	337.483
10	15		256.217	153.845	598.220
15	5		1078.784	3944.737	585.388
15	10		2619.743	207.544	797.007
15	15		242.911	2136.622	841.595
5	5		116.070	130.554	333.798
5	10		141.505	151.137	245.572
5	15		215.022	172.005	220.226
10	5		259.410	188.989	170.984
10	10	50%	193.951	190.273	142.342
10	15		221.731	150.830	179.428
15	5		3136.764	410.101	611.379
15	10		346.730	342.485	1300.353
15	15		1240.083	305.663	2002.806
5	5		140.848	74.582	83.399
5	10		45.805	47.236	67.336
5	15		48.145	58.753	82.308
10	5		48.309	53.866	119.722
10	10	75%	47.955	80.837	125.650
10	15		52.268	89.556	175.910
15	5		56.172	85.643	359.179
15	10		516.368	97.380	173.891
15	15		108.587	289.138	87.770

The results in Table III and Figures 5-7 show the results of the execution times (in seconds) of the previously reported executions for circuits of 25-75% single and two-qubit gates. The results confirm those obtained in Table I and Figures 3 and 4, where the complexity of the calculation (i.e. depth, number of qubits, and gates) is what controls the execution time. Indeed, as the complexity of the calculation increases, the execution time increases proportionally except in some peculiar cases, such as when using the `Brisbane` machine with a circuit requiring 15 qubits and a depth equal to 10. This atypical behavior might be explained by several factors. The most probable being a transpilation overheard due to some features of the `Brisbane` machine (e.g. calibration).

2) Quantum Machine Calibration Forecasting: These experiments investigate the forecasting part of the devised pipeline (see Section III-B). Table IV presents the results of the time series forecasting of the noise-related features of the `Brisbane`, `Kyoto`, and `Osaka` quantum machines. This includes a comparison between two state-of-the-art models, the Neural Prophet and the LSTM neural networks. The dataset has been split into training and testing ones (80%, 20%), where the testing has been done on the calibration obtained from the last three days of data extraction. The comparison has been made based on the RMSE metric (see Equation (10)). The forecasting has been made for each of the calibration metrics, including decoherence and dephasing times `T1` and `T2`, respectively, as well as the measurement and readout errors named `prob_meas0_prep1`, `prob_meas1_prep0`, and `readout_error`. The results in Table IV and the distribution of the calibration values in V show that the resulting forecasting error is minimal, where the best model was found to be the LSTM, although for the `Osaka` machine, the Neural Prophet was found to be the best. Considering these findings, one can conclude that the LSTM is, overall, the best for the forecasting of machine calibration. It is noteworthy to indicate that as the forecasting period is more distant in the future, the forecasting tends to be also far from the real calibration value, although the difference is low. So, one would advise forecasting calibration data over a period of time that is close in the future. This being said, considering the three-day lapse of time used in these experiments, one can note that the forecasting models do not lose much efficiency. The superiority of the LSTM over Neural Prophet might be explained by model (e.g. settings, features, etc.) and data-oriented factors (e.g. distribution).

3) Quantum Error Prediction: These experiments investigate the prediction part of the proposed pipeline (see Section III-B). It analyses the predictions of the error induced by the given circuit executed on some machine with a specific calibration. The calculation features are described in terms of the number of qubits (5 and 10), depth of the circuit (5 and 10), percentage of single and two-qubit gates (25% and 75%), number of `T`, phase, Hadamard, and `CNOT` gates. On the other hand, machine calibration considered in this work consists mainly of qubit features. This includes decoherence times (relaxation and dephasing) named `T1` and `T2`, respectively, as well as measurement and readout and measurement error probabilities named `prob_meas0_prep1`, `prob_meas1_prep0`, and `readout_out`. The first step consisted in obtaining the data with the calibrations of the `Kyoto` machine over several time steps (see Table VI) that generated a total of 8 circuits for each configuration (i.e. the possible combinations of circuit depth, number of qubits and the percentage of single/two-qubit gates) as shown in Table VII. Then, we execute the calculations to obtain both Kullback-Leibler and Jensen-Shannon divergences, as well as the simulation time required. However, one should stress that the XGBoost and the MLP neural network have been trained using the Jensen-Shannon divergence, considering it is more robust than the Kullback-Leibler when dealing with asymmetric distributions.

Table VIII represents the results of the calculation error prediction, where each row represents a different calculation (i.e. circuit configuration). As it can be seen in Table VIII, both the XGBoost and the MLP neural networks achieve similar predictions in general. One can observe that the difference regarding the real values of the error is minor for the MLP model, although there is no substantial difference regarding the XGBoost. As a conclusion of the experiments, one can

TABLE IV
RMSE RESULTS: BRISBANE, KYOTO, AND OSAKA

Calibration	Neural Prophet	LSTM
Brisbane Quantum Machine		
T1	18.591	**4.651**
T2	**6.174**	9.143
prob_meas0_prep1	4.034e-04	**2.198e-04**
Prob_meas1_prep0	1.297e-03	**1.136e-03**
Readout_error	1.752e-03	**3.964e-04**
Kyoto Quantum Machine		
T1	3.530	**3.205**
T2	19.760	**3.719**
Prob_meas0_prep1	**3.429e-04**	6.289e-04
Prob_meas1_prep0	1.481e-03	**4.440e-04**
Readout_error	1.958e-03	**3.551e-04**
Osaka Quantum Machine		
T1	13.611	**12.722**
T2	32.166	**7.924**
Prob_meas0_prep1	**1.500e-3**	2.173e-3
Prob_meas1_prep0	**1.646e-3**	3.413e-3
Readout_error	**1.927e-3**	2.356e-3

TABLE V
RANGE OF THE CALIBRATION VALUES

Calibration	Mean	Median	Deviation
Brisbane Quantum Machine			
T1	234.180	241.582	78.911
T2	156.293	145.716	94.270
Prob_meas0_prep1	0.0319	0.016	0.047
Prob_meas1_prep0	0.030	0.012	0.050
Readout_error	0.031	0.014	0.047
Osaka Quantum Machine			
T1	271.316	276.440	94.088
T2	147.139	131.168	98.502
Prob_meas0_prep1	0.053	0.025	0.084
Prob_meas1_prep0	0.050	0.020	0.088
Readout_error	0.051	0.024	0.080
Kyoto Quantum Machine			
T1	211.999	216.543	75.011
T2	130.746	112.932	91.253
Prob_meas0_prep1	0.041	0.015	0.069
Prob_meas1_prep0	0.040	0.016	0.060
Readout_error	0.041	0.060	0.061

TABLE VI
SNAPSHOT OF THE CALIBRATION DATA USED IN THE THIRD GROUP OF EXPERIMENTS

T1	T2	prob_meas0_prep1	prob_meas1_prep0	readout error
216.54	112.93	0.0150	0.0163	0.0158
220.66	109.59	0.0145	0.0143	0.0144
222.65	118.05	0.0156	0.0144	0.0149
219.32	104.13	0.0153	0.0146	0.0159
226.70	104.91	0.0140	0.0146	0.0151
223.51	105.32	0.0150	0.0155	0.0160
225.72	104.50	0.0135	0.0135	0.0148
221.22	103.21	0.0144	0.0148	0.0146
218.47	100.54	0.0140	0.0146	0.0151
220.55	96.85	0.0150	0.0162	0.0153

TABLE VII
SNAPSHOT OF THE CIRCUITS USED IN THE THIRD GROUP OF EXPERIMENTS

Qubits	Depth	Gates	T	Phase	H	CNOT
5	5	25%	1.0	1.0	1.0	7.0
5	5		1.0	2.0	1.0	6.0
10	5		0.0	0.0	1.0	16.0
10	10		11.0	16.0	19.0	13.0
5	10		2.0	3.0	1.0	16.0
10	5	75%	11.0	6.0	4.0	7.0
5	5		6.0	1.0	2.0	5.0
5	10		10.0	11.0	10.0	6.0
5	5		0.0	1.0	0.0	8.0
10	10		22.0	15.0	15.0	16.0

determine that the MLP model can infer relations that are more complex than what XGBoost can do for the prediction of quantum calculation error, given some specific quantum machine features.

TABLE VIII
RESULTS OF THE ERROR PREDICTION

Multilayer Perceptron	XGBoost	Real Data
0.0778	**0.0734**	0.0363
0.0748	0.0698	0.1198
0.2654	0.2207	0.2838
0.5505	0.5355	0.6931
0.1375	0.1330	0.1546
0.2740	0.2494	0.2740
0.0955	0.0692	0.1512
0.1655	0.1454	0.3388
0.0795	0.0675	0.0848
0.5894	0.5480	0.6469

V. CONCLUSIONS AND PERSPECTIVES

This paper proposes a complete pipeline based on machine learning for both quantum machine calibration *forecasting* and quantum error *prediction*. The contributions of the paper include (I) a data module that extracts, transforms, and stores the data, (II) a machine learning unit for both quantum machine features forecasting and computational error prediction, and (III) a web service-based interface for user-friendly use. The pipeline has been built and trained using three 127-qubit IBM quantum machines, where data was extracted over 3.5 months. The machine learning model studies several families of techniques using parametric and non-parametric ones, including those based on neural networks and others. They include Neural Prophet, LSTM, multiperceptron neural networks, and XGBoost. The results obtained are satisfactory, which strengthens confidence in the application of the proposal in real-life scenarios.

Future steps include (I) extending the training data by considering a longer period of extractions and also the calibrations of machines of different manufacturers besides IBM. (II) Extending the machine learning module by studying other parametric and non-parametric forecasting and prediction methods. This includes Gated Recurrent Unit (GRU), and also SARIMAX by including event information such as patterns of quantum machine maintenance days. (III) Extend the validation to other universal quantum gate sets, and emphasise a particular class of quantum algorithms, such as the quantum variational ones. Lastly, we plan on integrating the pipeline within the Qiskit SDK.

ACKNOWLEDGEMENTS

The authors acknowledge that this research is partially funded by the University of Malaga (PAR 4/2023), as well as the AIM-Zero ID2024-158752OB-I00 research project.

REFERENCES

[1] U. Aseguinolaza, N. Sobrino, G. Sobrino, J. Jornet-Somoza, and J. Borge, "Error estimation in current noisy quantum computers," *Quantum Information Processing*, vol. 23, no. 5, 2024.

[2] G. Welsch and P. Kowalczyk, "Designing explainable predictive machine learning artifacts: Methodology and practical demonstration," 2023. [Online]. Available: https://arxiv.org/abs/2306.11771

[3] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.

[4] O. Mukkula, "Master thesis: Optimizing coherence and anharmonicity of superconducting qubit," 2023, available at https://helda.helsinki.fi/items/01154831-44bc-432d-9d25-df0c933dfd4f.

[5] X. Li, X. Cheng, X. Chen, Z. Guan, P. Zhu, and H. Gu, "Quantum circuit output prediction based on time-series neural network integration," in *Proceedings of the 3rd International Conference on Cryptography, Network Security and Communication Technology*, ser. CNSCT '24. New York, NY, USA: ACM, 2024, p. 538–542.

[6] A. Casolaro, V. Capone, G. Iannuzzo, and F. Camastra, "Deep learning for time series forecasting: Advances and open problems," *Information*, vol. 14, no. 11, 2023.

[7] X. Kong, Z. Chen, W. Liu, K. Ning, L. Zhang, S. Muhammad Marier, Y. Liu, Y. Chen, and F. Xia, "Deep learning for time series forecasting: a survey," *Int. Journal of Machine Learning and Cybernetics*, 2025.

[8] L. Schmid, M. Roidl, A. Kirchheim, and M. Pauly, "Comparing statistical and machine learning methods for time series forecasting in data-driven logistics—a simulation study," *Entropy*, vol. 27, no. 1, 2025.

[9] Z. A. Dahi, F. Chicano, G. Luque, and E. Alba, "Genetic algorithm for qubits initialisation in noisy intermediate-scale quantum machines: the ibm case study," in *Proceedings of the Genetic and Evolutionary Computation Conference*, ser. GECCO '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 1164–1172.

[10] S. Martina, S. Gherardini, L. Buffoni, and F. Caruso, "Noise fingerprints in quantum computers: Machine learning software tools," *Software Impacts*, vol. 12, p. 100260, 2022.

[11] O. Triebe, H. Hewamalage, P. Pilyugina, N. P. Laptev, C. Bergmeir, and R. Rajagopal, "Neuralprophet: Explainable forecasting at scale," *ArXiv*, vol. abs/2111.15397, 2021.

[12] J. Á. M. Martínez, Z. A. Dahi, F. Chicano, G. Luque, and E. Alba, "Time series forecasting for parking occupancy: Case study of malaga and birmingham cities," in *Optimization and Learning - 6th International Conference, OLA 2023, Malaga, Spain, May 3-5, 2023, Proceedings*, ser. Communications in Computer and Information Science, vol. 1824. Springer, 2023, pp. 368–379.

[13] S. Sivarajah, S. Dilkes, A. Cowtan, W. Simmons, A. Edgington, and R. Duncan, "t—ket⟩: a retargetable compiler for nisq devices," *Quantum Science and Technology*, vol. 6, no. 1, p. 014003, nov 2020. [Online]. Available: https://dx.doi.org/10.1088/2058-9565/ab8e92