ADVANCED QUANTUM TECHNOLOGIES
www.advquantumtech.com

Q-fid: Quantum Circuit Fidelity Improvement with LSTM Networks

Yikai Mao, Shaswot Shresthamali, and Masaaki Kondo*

The fidelity of quantum circuits (QC) is influenced by several factors, including hardware characteristics, calibration status, and the transpilation process, all of which impact their susceptibility to noise. However, existing methods struggle to estimate and compare the noise performance of different circuit layouts due to fluctuating error rates and the absence of a standardized fidelity metric. In this work, Q-fid is introduced, a Long Short-Term Memory (LSTM) based fidelity prediction system accompanied by a novel metric designed to quantify the fidelity of quantum circuits. Q-fid provides an intuitive way to predict the noise performance of Noisy Intermediate-Scale Quantum (NISQ) circuits. This approach frames fidelity prediction as a Time Series Forecasting problem to analyze the tokenized circuits, capturing the causal dependence of the gate sequences and their impact on overall fidelity. Additionally, the model is capable of dynamically adapting to changes in hardware characteristics, ensuring accurate fidelity predictions under varying conditions. Q-fid achieves a high prediction accuracy with an average RMSE of 0.0515, up to 24.7× more accurate than the Qiskit transpile tool `mapomatic`. By offering a reliable method for fidelity prediction, Q-fid empowers developers to optimize transpilation strategies, leading to more efficient and noise-resilient quantum circuit implementations.

1. Introduction

By using quantum computers that exploit the principles and phenomena of quantum mechanics, it may be possible to achieve superpolynomial or even exponential speedups for traditionally hard computing problems.[1] With quantum computers, in addition to representing the classical bits 0/1 using its computational basis $|0\rangle$ / $|1\rangle$, the qubits also have the ability to switch to any desired basis, elevating the calculation to a higher dimension. Moreover, by entangling multiple qubits, the computation can be performed simultaneously in an exponentially larger space.

However, present-day quantum computers have a limited amount of qubits with limited interconnectivity and very short coherence lifetimes (in order of milliseconds).[2,3] Furthermore, the gate operations and readout are very susceptible to external noise. To overcome the noise limitations of today's machines, Noisy Intermediate-Scale Quantum (NISQ)[4] offers a viable solution by using error-mitigation techniques to compensate for fragile qubits.[5–9]

To implement a quantum circuit on a real NISQ processor, the circuit must be mapped onto the available physical qubits according to the hardware's connectivity map, a process called transpilation.[10,11] There are several challenges during this process: First, the qubits in one processor are not identical. Each of them has unique physical properties that define their noise characteristics, and they can change depending on the time of operation. This causes the quantum circuits to exhibit different noise performances when they are placed on different qubits. Second, due to the limited qubit connectivity, some quantum circuits need to be modified from the original design before implementing on real hardware. For example, adding SWAP gates to bring two physically distant qubits together for a CNOT operation. Therefore, the accuracy of a quantum circuit can get compromised even if it is placed on high-quality qubits, due to the additional gate noise introduced by circuit transpilation.

Y. Mao, M. Kondo
Graduate School of Science and Technology
Keio University
Yokohama, Kanagawa 223-8522, Japan
E-mail: ykmao@acsl.ics.keio.ac.jp

S. Shresthamali
Graduate School of Information Science and Electrical Engineering
Kyushu University
Fukuoka, Nishi-ku 819-0395, Japan

S. Shresthamali
Kondo Laboratory
Keio Univerisity, Yokohama, Kanagawa 223-8522, Japan

M. Kondo
RIKEN Center for Computational Science
Kobe, Hyogo 650-0047, Japan

The ORCID identification number(s) for the author(s) of this article can be found under https://doi.org/10.1002/qute.202500022

DOI: 10.1002/qute.202500022

To overcome these challenges, we need a tool to accurately pick a high-fidelity circuit-to-qubit layout from all the transpilation options. This way we can efficiently retrieve high-quality measurements, without wasting quantum computing resources on testing error-prone layouts. Additionally, a promising direction for implementing large quantum circuits on NISQ processors is by breaking them down into smaller circuits.[12,13] In this case, picking a higher-fidelity layout for the smaller circuit is even more important because it will eventually affect the final output for the full circuit.

One such tool to find a high-fidelity layout is `mapomatic`, built within the Qiskit transpiler.[14] `mapomatic` estimates fidelity by accumulating the individual gate error rates, obtained from the Randomized Benchmarking (RB) experiments.[15] This approach has some severe limitations. For example, the RB experiments must be performed frequently to keep the error rates up to date. Also, the noise model of a full quantum circuit is more complex than the accumulation of the individual error rates.[16] Due to these limitations, `mapomatic` cannot estimate the fidelity accurately and it only gives the relative performance comparison for a set of transpilations.

In this work, we present a practical metric: d-R^2, that intuitively represents the fidelity of the output distribution from a quantum circuit. Here, fidelity refers to the computational accuracy of a quantum circuit/gate (a measurement of how closely the actual output of the quantum circuit matches the expected output), which is different from the qubit state fidelity (a quantification of the overlap between two states). Using this metric, we then develop an LSTM-based system: Q-fid, to accurately predict the fidelity of a quantum circuit. It eliminates the need for frequent RB experiments by actively learning qubit/gate operations from historical circuit execution data, without any separate input of system calibration parameters or error rates. Traditionally, a NISQ circuit requires hundreds to thousands of shots on a real processor to statistically estimate its correct output. With Q-fid, we can choose to only execute circuits that have higher fidelity and obtain the solution with fewer shots, thus saving precious quantum computing resources.

The contributions of this work are listed below:

1. We propose a simple and intuitive method to model a quantum circuit using text. This method can be applied to any gate-based quantum processor and enables feeding the quantum circuits directly into an LSTM neural network.
2. We present the discrete coefficient of determination (d-R^2) to evaluate the noisy output distribution of a quantum circuit. d-R^2 uses the uniform distribution as the worst-case output, offering a reasonable baseline for comparing NISQ algorithm fidelities.
3. We show that LSTM is effective in learning the error properties of a qubit and a quantum gate. The trained system, Q-fid, can predict the performance of a quantum circuit without any separate input of hardware calibration data or gate error rates.
4. We provide a framework to use LSTM for on-the-fly quantum circuit fidelity estimation, including the architecture of the neural network, how to build the dataset using Randomized Benchmarking, and the training workflow.
5. Experiments using real NISQ algorithms show that because Q-fid can accurately predict the d-R^2 score of quantum circuits, we can retrieve more high-fidelity, usable transpilations than `mapomatic` from a large set of transpiled circuits.

2. Background

2.1. NISQ Circuits and Processors

In a gate-based quantum computer, the quantum bits (qubits) are manipulated by a sequence of quantum gates as described by the quantum circuit (QC).[18] These gates change the state of the qubit and the transformations can be expressed by unitary matrices, i.e., the computation is reversible. **Figure 1** shows a sample implementation of the Bernstein–Vazirani Algorithm[19] divided into three stages. 1) State preparation: The qubits are initialized into the superposition state $|-++\rangle$ from $|000\rangle$. 2) Computation: Once initialized, the quantum computation (CNOT as in this example) is performed. 3) Measurements: The qubits are returned to their original basis and measured into the classical registers c_1c_0 to store the output.

NISQ processors have limited qubit availability, both in terms of qubit quantity and quality.[4] **Figure 2a** gives the architecture of the 7-qubit quantum processor *ibm_nairobi* along with three major forms of possible error: readout, single-qubit error, and CNOT error. The qubit quality is commonly characterized by their T_1/T_2 constants. T_1 is called the coherence time and T_2 is the decay time, which describes how long a qubit relaxes to the ground state and how long it can hold its phase. However, as shown in Figure 2(b), because qubits are very sensitive to multiple sources of noise, their T_1/T_2 constants can fluctuate significantly, and it is hard to predict which physical qubit is more stable than others at a given time.

2.2. Circuit Transpilation

Due to the limited connectivity of current NISQ processors, it is not always possible to map a QC onto the processor directly.[20] For example, the physical CNOT links exist only between the neighboring qubits in Figure 2a. So if a QC requires a CNOT gate between qubit 0 and qubit 3, additional operations must be added to the QC to compensate for the missed connection. This process of translating a hardware-agnostic QC description to implement in a given hardware platform is referred to as transpilation.

To perform a long-distance CNOT gate in a superconducting quantum computer like *ibm_nairobi*, the transpiler can insert SWAP gates to switch the position of the nonadjacent physical qubits to use the existing CNOT links.[21] However, in addition to connectivity restrictions, real quantum processors also have limited single-qubit gate availability. For example, *ibm_nairobi* only supports four single-qubit gates: ID, RZ, SX, X. Therefore, To perform a single-qubit operation like the Hadamard gate, the transpiler will also need to decompose the operation into the gates available in the processor.

2.3. Randomized Benchmarking

Randomized Benchmarking (RB)[15,22] is an experiment to estimate the error rates of the set of common quantum gates, usually called the Clifford gates. For IBM Q devices, this Clifford gate

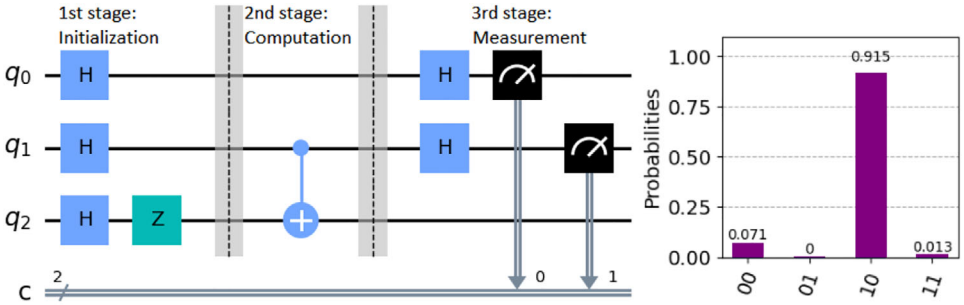

Figure 1. Bernstein–Vazirani Algorithm. The expected output is $q_1q_0 = |10\rangle$ (q_2 is not measured). Output distribution after running the circuit for 1,024 times on real hardware (*ibm_nairobi*) is plotted on the right, the correct state $|10\rangle$ has been measured 937 times, but the wrong measurements $|00\rangle$ and $|11\rangle$ also appears in the distribution due to noise.

Figure 2. a) Error map of *ibm_nairobi* on Dec. 9, 2022, generated by the IBM Q platform.[17] b) T_1/T_2 fluctuation of the 7 physical qubits inside *ibm_nairobi*. 100 data points were collected from Aug. 25 to Dec. 18, 2022. Note that data is not continuous due to scheduled/unscheduled system maintenance, for example around Nov. 1.

group includes [X, Z, P, H, CNOT, CZ, SWAP].[23] Based on the reversible principle of quantum gates, the RB experiment first generates a QC containing random quantum gates in the Clifford group, then it calculates a complementary gate sequence that can reverse the computation performed in the first QC. An example RB circuit is given in **Figure 3**. By applying the two generated QCs back-to-back, the full circuit is equivalent to an identity operator so ideally any qubits involved in this circuit should never change their states when measured at the end.

However, when executing an RB circuit on a NISQ processor, it is possible that the qubits cannot return to their initial state due to the noisy nature of the hardware. Therefore, by repeatedly running RB circuits and measuring the qubit outcomes, we can estimate the average fidelity of the processor, and use that information in turn to predict the gate error rate, either for 2-qubit gates or single-qubit gates.

2.4. Long Short-Term Memory Network

Neural Networks (NN) have demonstrated state-of-the-art performance in various tasks including Computer Vision (CV) and Natural Language Processing (NLP). Among numerous NN architectures, Long Short-term Memory (LSTM) networks[24] have been a popular choice for tasks related to time series processing, for example, weather forecasting and sentiment prediction.

We leverage the ability of LSTMs to learn temporal relationships in sequential data to estimate circuit fidelity. It is very easy to see that quantum circuits are essentially sequential gate operations applied on qubit(s). There is an inherent temporal sequence in the execution of the circuit on qubits. Furthermore, the fidelity of the entire circuit not only depends on the individual gate/qubit characteristics but also on how they interact with each other as the circuit progresses toward completion, much like words in a sentence or notes of a musical score. Since LSTMs are trained to recognize these temporal relationships, we use them to estimate circuit fidelity in our work. To our knowledge, this is the first work that uses LSTM for fidelity estimation.

As **Figure 4** shows, the x-axis can be used to indicate the timesteps of a QC from start to end, with each timestep modeled as a "QC layer" containing all the gates in the y-axis. The layers act on the same set of qubits and have fixed widths, so the QC can be described as a 2D time series with its width equal to the number of qubits in the circuit, and its length equal to the number of layers (sometimes referred to as the depth) of the circuit.

3. Proposed Q-Fid Framework

3.1. Framework Description

Q-fid is an LSTM network that takes a QC as input and predicts the fidelity of its output distribution. Inspired by one of the most popular LSTM applications, Sentiment Analysis,[25] the workflow of Q-fid is very similar to the workflow of many common NLP tasks. In these tasks, the LSTM takes a sentence as input and gives a prediction based on different objectives and contexts. For

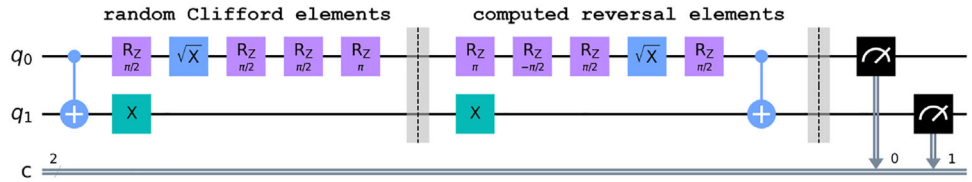

Figure 3. Example of a Randomized Benchmarking circuit. The first part is the randomly generated Clifford gates, followed by the calculated reversal gates. The final measurement should be |00⟩ if no errors occur.

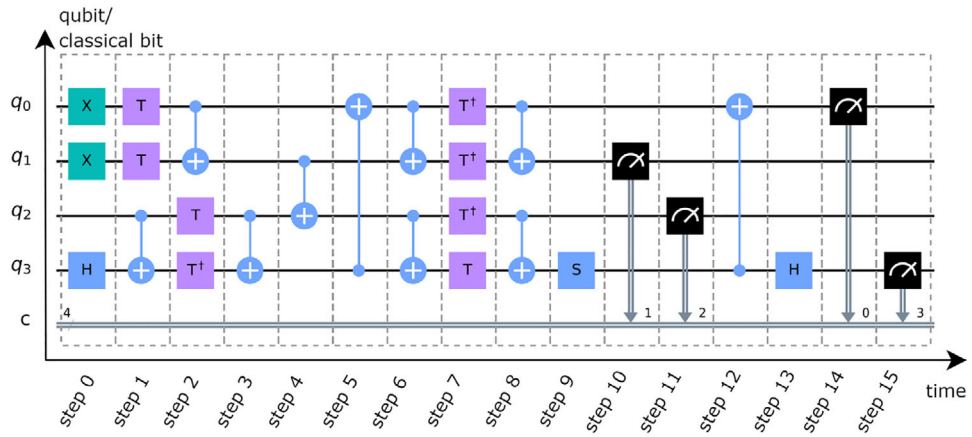

Figure 4. Quantum circuit placed in a coordinate system with the y-axis as qubit/classical bit, and the x-axis as time. It can be viewed as the qubits being manipulated through each timestep from the left to the right on the time axis.

example, translate the sentence or guess the sentiment. Although every individual word has its own definition, once combined together, they become elements of a single sentence where the meaning of the sentence must be inferred from the relationship between all of the words, i.e., the contexts.

This concept fits surprisingly well when applied to a QC: Although every individual quantum gate has its own operation and noise characteristics, once combined together, they become elements of a single circuit where the computation and fidelity of the qubits must be measured after all quantum gates have been executed. Similar to a sentence where different orders of the same words can express different sentiments, the same set of quantum gates can perform different computations and express different fidelity depending on their order in a circuit. Q-fid uses LSTM to catch this long-term temporal dependence and noise characteristics inside a QC.

At the core of the proposed Q-fid framework is an LSTM network with a lightweight architecture. The input layers first perform general pre-processing of the input QCs, then they are passed into the LSTM layer to extract long-term and short-term noise dependencies between the gates inside the QC. Finally, the output from the LSTM layer is passed into a series of Fully-Connected layers to generate the final prediction of the circuit fidelity. We use a data-driven approach to train the Q-fid system, and the noise characteristics of the hardware are approximated by the 700 000+ parameters inside the LSTM network. The number of input neurons is decided by the number of qubits of the quantum processor. In other words, a larger quantum processor will have a larger Q-fid neural network with more inputs attached to it. An overview of Q-fid's LSTM architecture is shown in **Figure 5** from (g) to (I).

A distinctive feature of Q-fid is that it does not require any explicit input of the processor's calibration data ($T1/T2$ frequency, etc.) to make predictions, since the LSTM infers the noise characteristics from the input QC during training. This hardware-agnostic feature gives Q-fid several advantages over the other calibration-based prediction systems. First, because the hardware descriptions are abstracted away, Q-fid can work with any gate-model quantum computer regardless of whether the qubits are superconducting or trapped ion. Second, since the user does not need to specify any device calibration data, Q-fid can be trained dynamically during regular workload and adapts to the ever-changing device characteristics, all without interruptions caused by calibration or maintenance jobs.

3.2. Discrete Coefficient of Determination (d-R^2)

The LSTM updates its parameters by comparing the differences between the true output probability distribution of the QC and the observed output distribution. We thus need a single metric that represents this difference so that we can feed it into the loss function of the LSTM network. In this paper, we apply a modified version of the Coefficient of Determination (R^2) as our metric for evaluating noisy QC fidelity. R^2 is commonly used in regression analysis to show the goodness of fit,[26] where $R^2 = 1$ indicates a perfect fit and $R^2 = 0$ indicates that the fitted line does not represent the original data at all.

Recent works[11,27] use the Probability of Successful Trials (PST) as a metric to quantify the performance of a noisy QC, which is defined as the ratio of the number of successful

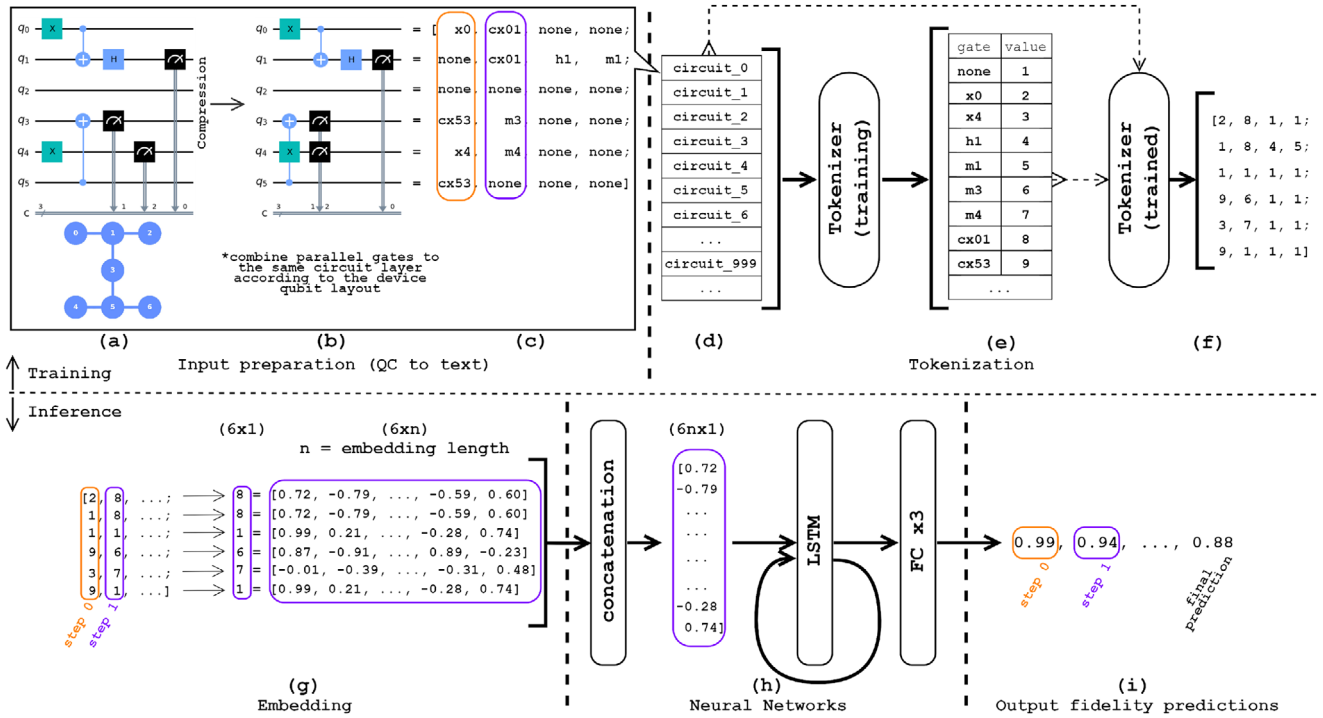

Figure 5. Overview of Q-fid's workflow. The first stage is input preparation (a, b, c), where the QCs are compressed and mapped to their corresponding text label. The compression causes the gates to stack visually but they are executed individually on hardware because the qubits are not physically connected (as shown in Figure S4a, Supporting Information). The second stage is tokenizer training (d, e), the tokenizer will look through all the QCs in the dataset and assign each text label with an integer according to their appearing frequency. After the tokenizer is trained, it will use its internal dictionary to translate all of the QCs in the dataset to an integer representation, which is called tokenization (f). Q-fid's neural network uses embedding (g) and LSTM (h) layers to encode the integer labels and then extract sequential and relationship information from the QC, each prediction (i) is associated with one timestep in the QC. After the final timestep, the output will be the fidelity prediction of the full QC, represented as a d-R^2 score.

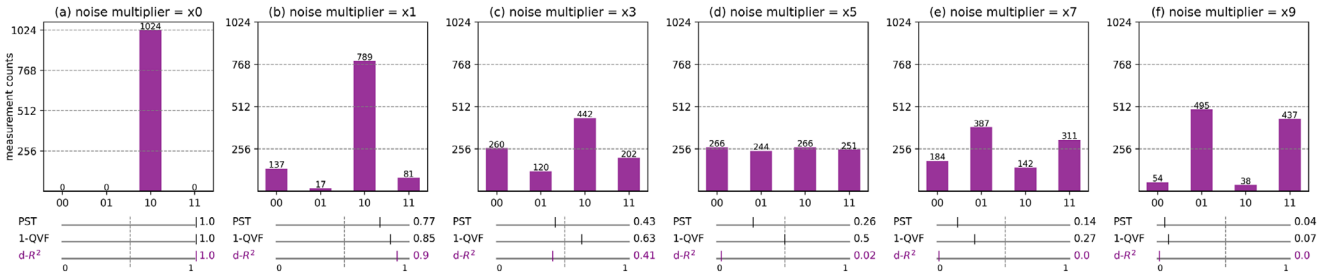

Figure 6. Output distributions after executing the QC in Figure 1 with 1,024 shots using the Qiskit Aer simulator with varying noise intensity. The base error rates for gates, measurements, and reset are set as 0.05n, 0.1n, and 0.03n. n is the noise multiplier where ×0 means noise-free and ×10 is the maximum possible value. The comparison of three QC fidelity metrics is shown below the bar graph, where 1 means perfect circuit fidelity. Panel (a) is the noise-free output, the correct output state should be |10⟩. Panels (b) to (f) are the outputs under different noise multiplier values. The output in Panel (d) is a uniform distribution, which gives zero information on which state might be the correct state. Panel (f) is considered as a faulty output, the circuit is not doing the intended quantum operations so the wrong states are appearing more than the correct state.

trials to the total number of trials. Although PST is easy to calculate, it does not give the user enough information to analyze the circuit. For example, **Figure** 6e shows a distribution where PST gives a fidelity score of 0.14. However, the user cannot distinguish whether this low fidelity is caused by |00⟩ or |10⟩, due to their similar measurement counts. Other metrics have been proposed to replace PST, for example, the Quantum Vulnerability Factor (QVF)[28] and the Total Variation Distance (TVD).[29] However, they do not offer a clear definition when multiple correct states are expected, which often happens for common NISQ algorithms like QAOA[30] and VQE.[31]

Our modified version of R^2 is called discrete-R^2 (d-R^2). Compared with the other metrics, d-R^2 has two important features that make it suitable for NISQ fidelity analysis. First, it takes all the states in the distribution into consideration. So in addition to checking how well the correct states are standing out, d-R^2 also penalizes wrong states when they should not appear in the output distribution, which makes it work well with algorithms

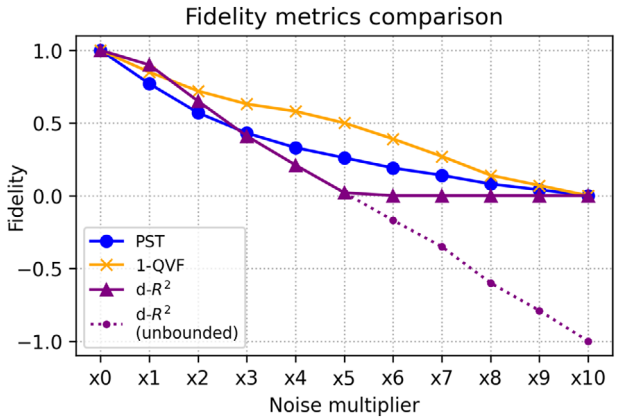

Figure 7. Fidelity metrics comparison when different noise multiplier values are applied on the circuit in Figure 1. Note that d-R^2 can go to the negative region if the bounding condition in Equation (1) is removed.

that output multiple correct states. Second, d-R^2 is a measurement of closeness to the uniform superposition, which gives the user d-$R^2 = 0$ as in Figure 6d. Because the output distribution of noisy QCs can rapidly converge to the uniform distribution,[32–35] we think a practical fidelity metric should clearly recognize this worst-case distribution for the user.

A uniform superposition as in Figure 6d does not give any useful information about which state(s) should be the correct state. However, PST and QVF will still show fidelity scores of 0.26 and 0.50. Since every possible output state has equal measurement opportunity, the user can just count the number of qubits in the circuit and randomly pick some states as the output, which invalidates the purpose of executing the quantum circuit in the first place. Sometimes the output distribution fits worse to the expected distribution than a uniform superposition, as in Figure 6e,f. These distributions are caused by defective hardware operations since the error rates for measurements in those QCs are already over 0.5 (0.7 in (e) and 0.9 in (f)). Because such output distribution is faulty and extremely misleading to the user, they should not be used to interpret the QC. We use the bounding condition $SSR < SST$ to limit d-$R^2 \geq 0$ for this practical purpose. However, if the user wants to quantify this faulty distribution, it is still possible to remove the bounding condition and have a negative d-R^2 score, as shown in **Figure 7**.

To calculate d-R^2, we first calculate two **S**um of **S**quares (SS):

$$SS_{residual} = SSR = \sum_{i=0}^{2^n} \left(Y_i - \gamma_i\right)^2$$
$$SS_{total} = SST = \sum_{i=0}^{2^n} \left(Y_i - mean(Y)\right)^2 \quad (1)$$

Here, n is the number of measured qubit(s), Y is the distribution containing all the measurement counts from the noise-free output, and γ is the distribution containing all the measurement counts from the noisy output. For example in Figure 6b, Y would be $(0, 0, 1024, 0)$, so $mean(Y)$ is 256, and γ would be $(137, 17, 789, 81)$. Note that when calculating the subtractions, the indices of Y and γ must be aligned so that the measured states match each other. Then d-R^2 can be obtained as:

$$\text{d-}R^2 = \begin{cases} 1 - \frac{SSR}{SST}, & \text{if } SSR < SST \\ 0, & \text{otherwise} \end{cases} \quad (2)$$

Compared with other metrics empirically, the fixed definition of d-$R^2 = 0$ as the uniform superposition gives finer granularity for calculations and allows the fidelity interpretations to stay invariant when applied to different circuits. Based on the observations during our experiments and the analysis in Figure 6, we offer the recommended interpretations for different d-R^2 values used in this paper in **Table 1**.

Table 1. discrete-R^2 values and proposed interpretations.

d-R^2	Interpretation
= 1	output is perfect, same as if there was no noise.
> 0.7 to ≤ 1	output is good, the circuit has high fidelity.
> 0.5 to ≤ 0.7	output quality is fair, contains noticeable noise.
> 0.3 to ≤ 0.5	output contains significant noise, interpret with caution.
> 0 to ≤ 0.3	output is extremely noisy, do not use.
= 0	output is no better than a uniform superposition.

When the QC is designed to output a uniform superposition, like the Quantum Random Number Generator (QRNG),[36] the output with no noise and the output with extreme noise can be the same (both are uniform superposition), since heavy noise will cause the output distribution to eventually converge into the uniform distribution.[32–35] In this case, the output is both perfectly correct and incorrect at the same time, so the definition of fidelity contradicts itself, causing d-R^2 to become undefined by design due to $SST = 0$.

3.3. Text-Based Representation of Quantum Circuits

We represent QC in text string formats that are suitable for feeding into the LSTM networks. The QC is first compressed to its true depth, because qubits that are not physically connected in hardware can execute gates in parallel. Then, Q-fid uses a simple and effective protocol to label the gates inside a QC: a short string describing the gate function followed by the qubit index describing where the gate is executed. For example, a Hadamard gate placed on qubit 2 is labeled as `h2`, a CNOT gate with control on qubit 0 and target on qubit 3 will be `cx03`. This representation also reflects the layout of the device. For *ibm_nairobi*, there will never be a label called `cx34`, because this connection does not exist in the hardware. This helps the LSTM network to learn the processor's layout implicitly. Our text format is very similar to many existing quantum programming languages including openQASM and Qiskit, so preparing the text representation of an existing QC for Q-fid is straightforward:

1) A tokenizer reviews the full QC dataset and builds a dictionary that maps each text label of a gate to a unique integer. The labels are ranked according to the frequency they appear in the circuit, with the most common appearing label mapped

to 1. A special label in Q-fid is `none`, which means the qubit is staying idle. This label is important because the qubit can decay and become noisier even when no gates are applied. An example of this workflow is shown in Figure 5 from (a) to (f).

2) For a large QC dataset, the depth of the individual QCs can vary a lot. To improve the accuracy and efficiency of LSTM training, it is better to fix the number of time steps in the dataset. Once the maximum number of time steps is set, all of the integer-based QC vectors are either truncated or padded to the same depth. The reason for the tokenizer to start from 1 is that 0 is reserved as the padding element, so that the LSTM can safely skip it. We follow the common practices in NLP[37] to use pre-padding and post-truncation.
3) After all the text labels are converted to integers, every QC in the dataset is now a dense vector. However, the integers do not possess any relationship or similarity information between different gates, so it is hard for the LSTM to learn the effect of interconnected quantum gates. This problem is solved by using word embeddings, a technique of using a higher dimension fixed-length vector to replace the integer encoding.[38] The values of this embedding vector are trained with the LSTM in parallel, which helps Q-fid capture more detailed information from the QC. The embedding layers are shown in Figure 5g.

This tokenization process can represent a QC in human-readable text format and effectively retains vital circuit structural information, therefore making it possible for neural networks designed for time-series data to process a QC directly. More importantly, it enables existing NLP neural networks to perform analytical tasks on QCs without complex modification. The training of the tokenizer is expressed in Algorithm 1, and the inference process is described in Algorithm 2, all as part of the full Q-fid workflow.

Algorithm 1 Q-fid Training Workflow.

```
Q-fid Training
    Input: quantum circuit dataset
        ▷ circuits and corresponding noisy output
    Output: trained Q-fid system
Step 1: Input Preparation
    for each QC in dataset do
      noisy_fidelity(d-R^2) ← ideal noise-free output
        ▷ compare noisy/noise-free output to calculate training label
      compressed_QC ← compress(QC)
        ▷ combine parallel gates to the same circuit layer
      text_QC ← gate_to_text(compressed_QC)
        ▷ map gates to corresponding text label
    end for
Step 2: Tokenizer Training
    tokenizer ← initialize_tokenizer()
    for each text_QC in dataset do
      trained_tokenizer ← tokenizer.fit_on_texts(text_QC)
        ▷ train tokenizer on all circuits to generate dictionary
    end for
Step 3: Tokenization
    tokenized_QCs ← []
    for each text_QC in dataset do
      tokenized_QC ← trained_tokenizer.texts_to_int(text_QC)
      tokenized_QCs.append(tokenized_QC)
        ▷ translate QC to integer representation
    end for
Step 4: Q-fid Training
    for each tokenized_QC in tokenized_QCs do
      embedded_QC ← embedding(tokenized_QC)
        ▷ apply embeddings to integer labels
      lstm_output ← LSTM(embedded_QC.concat())
        ▷ process through LSTM
      fidelity_prediction ← fully_connected(lstm_output)
        ▷ final fidelity prediction
      model.fit(noisy_fidelity) ← fidelity_prediction
        ▷ fit model against noisy output fidelity score
    end for
    Return trained_Q_fid
```

3.4. Training Circuits and Dataset for Q-fid

Training Q-fid requires a large QC dataset that is also diversified in circuit depth and width, as the relationship between fidelity and circuit size is not necessarily linear. In this work, we utilize the RB circuits to efficiently create a QC dataset to train Q-fid. Because the gates in RB circuits are randomly generated, they span over all different types of gates, depths, widths, and qubit interactions. This provides the required diversity of width and depth for the dataset. In our generated *ibm_nairobi* dataset, all the available gates [X, RZ, SX, CNOT, Measurement] are sufficiently covered on all possible qubits, resulting in 40 total unique text labels plus the placeholder `none` label. The most common gate label in the dataset is `rz3` with 2, 341, 798 appearances, and the least common label is `m4` with 29, 528 appearances.

Using RB circuits in training provides many benefits, one of the advantages is that it greatly simplifies fidelity calculation. Every RB circuit is by definition equivalent to an identity operator, so if the qubits are initialized to $|0\ldots0\rangle$, we know the ideal output states must also be $|0\ldots0\rangle$. This means that in order to calculate the ground-truth training labels (d-R^2 scores) for an RB circuit, we only need to obtain one noisy output distribution of the circuit. In addition, most of the computing complexity for generating RB circuits comes from calculating the final circuit block that reverses the previous operations, and it can be calculated efficiently in polynomial time, proved by the Gottesman–Knill theorem.[39]

4. Experimental Section

4.1. Dataset

Two datasets were created based on two real IBM Quantum processors, *ibm_nairobi* and *ibmq_montreal*. The QCs were randomly generated according to the RB protocol, with the length of the RB sequence ranging from 1 to 5. Also, the number of active

Algorithm 2 Q-fid Inference Workflow.

Q-fid Inference
Input: Quantum circuits (QCs)
Output: Fidelity predictions of QC
Step 1: Input Preparation
for each QC in QCs **do**
  compressed_QC ← compress(QC)
  text_QC ← gate_to_text(compressed_QC)
end for
Step 2: Tokenization
tokenized_QCs ← []
for each text_QC in text_QCs **do**
  tokenized_QC ← trained_tokenizer.texts_to_int(text_QC)
  tokenized_QCs.append(tokenized_QC)
end for
Step 3: Q-fid Processing
for each tokenized_QC in tokenized_QCs **do**
  embedded_QC ← embedding(tokenized_QC)
  lstm_output ← LSTM(embedded_QC.concat())
  fidelity_prediction ← fully_connected(lstm_output)
end for
Return fidelity_prediction

qubits were changed when generating the RB circuits. For example, even though *ibm_nairobi* was a 7 qubit processor, RB circuits were generated that only require 1 qubit, 2 qubits, etc. The main reason for doing this was because placing the QC on different physical qubits on the same processor can yield different fidelity due to noise and manufacture variation. When the RB circuit only requires 1 active qubit, the circuit could be placed on seven different physical qubits on *ibm_nairobi*. This improves Q-fid's ability to learn the properties of individual qubits.

Due to the ever-changing nature of quantum errors, instead of depending on a fixed noise model to predict the fidelity of a quantum circuit, Q-fid was designed to actively learn the noisy output results from a quantum processor and incorporate the gate/qubit error parameters inside its hidden layers. During training, Q-fid needs the fidelity (d-R^2) score as the training label corresponding to the input circuit. This fidelity score is calculated by comparing the noisy output result to the theoretical noise-free result, which means that if the noise characteristics for a particular processor have changed, Q-fid will learn this change from the updated fidelity score and give new predictions based on the latest processor characteristics.

The *ibm_nairobi* QC dataset contains 103,500 circuits, and the *ibmq_montreal* QC dataset contains 100 000 circuits. All of the circuits are transpiled and converted to a text-based representation described in Section 3.3, then the noisy output results are captured using the Qiskit Aer simulator. We constrain the depth of the circuits to be less than or equal to 500 because circuits deeper than 500 have near-zero fidelity, as shown in **Figure 8**. After trimming, the *ibm_nairobi* QC dataset has 82 644 circuits, and the *ibmq_montreal* QC dataset has 91 386 circuits.

4.2. Model and Training

Two Q-fid models were built to test the two processors, the architecture of Q-fid for *ibm_nairobi* is shown in **Figure 9**. Each qubit has its own input neuron and they all have a length of 500, equal to the maximum QC depth of the dataset. The embedding layer transforms every gate label into a 64D dense vector and they are all concatenated together to represent one input timestep. The LSTM layer uses 256 memory units, followed by three Fully-Connected layers. ReLU activation and Sigmoid activation were used for the final layer. The model for *ibm_nairobi* contains 743 784 trainable parameters with seven input layers, and the model for *ibmq_montreal* contains 7 403 004 trainable parameters with 27 input layers. The model for *ibmq_montreal* has mostly the same architecture, only changing the number of input layers and adding the number of hidden units.

To demonstrate that no special modification is needed for existing LSTM architectures to analyze a quantum circuit, the original implementation of LSTM was picked with their standard forget/input/output gates by Hochreiter and Schmidhuber,[24] also implemented the text tokenizer and the embedding layer using the TensorFlow Keras API. The models run on a rack server with two Intel Xeon Gold 6354 processors and the Nvidia A100 GPU. The training workflow can be reproduced using the Jupyter Notebook available online mentioned in Section 8. The models were trained with a batch size of 32 and Adam optimizer for 20 epochs using the MSE loss function, the training automatically terminates if the loss does not improve for 5 continuous epochs. The training, validation, and test split ratio is 7:2:1, loss was evaluated on the validation set and pick the best-performing model on the test set. On average, each epoch took 81s to train and the inference time per circuit was 0.48ms. The training curve for the two models is shown in **Figure 10**, and **Figure 11** shows the scatter plot of real fidelity vs. predicted fidelity of Q-fid running on the test set.

5. Results and Discussion

Based on the number of qubits, connection complexity, and algorithm practicality, we picked 25 quantum circuits from the `QASMbench`[40] NISQ benchmark suite to demonstrate the performance of Q-fid. We compare our result with `mapomatic`, the latest and default mapping tool of the Qiksit transpiler. `mapomatic` predicts circuit fidelity and maps high-fidelity circuits onto the quantum processor, the prediction is made by accumulating individual gate error rates of the circuit,[41] which has the same prediction range and statistical meaning as d-R^2: 1 predicts a perfect output distribution equivalent to the noise-free simulation, and 0 predicts a distribution with maximum possible error rates, equivalent to the uniform distribution. The circuits are executed on the Qiskit Aer simulator[42] with a noise model that mimics *ibm_nairobi* and *ibmq_montreal*. Section 5.1 demonstrates Q-fid's prediction performance on *ibm_nairobi*. Section 5.2 demonstrates Q-fid's ability to correctly find high-fidelity transpilation layouts on *ibmq_montreal*. Section 5.3 demonstrates how Q-fid adapts to *ibm_nairobi*'s device variance on different dates. The four representative samples picked from the full result are: Hidden Subgroup problem (`hs4_n4`),[43] Quantum Ripple Carry

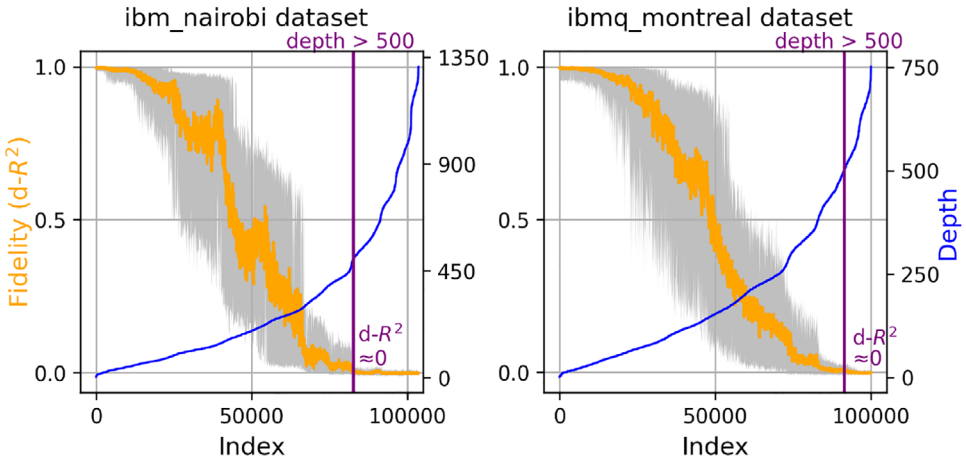

Figure 8. Two QC datasets showing the fidelity of a QC decreases with increasing depth. The x-axis represents the indices of RB circuits ordered from the shallowest to the deepest ones, with a vertical purple line indicating the cut-off point of *depth* = 500. The fidelity has a very large variance so the raw data is plotted using a gray envelope and an orange average line.

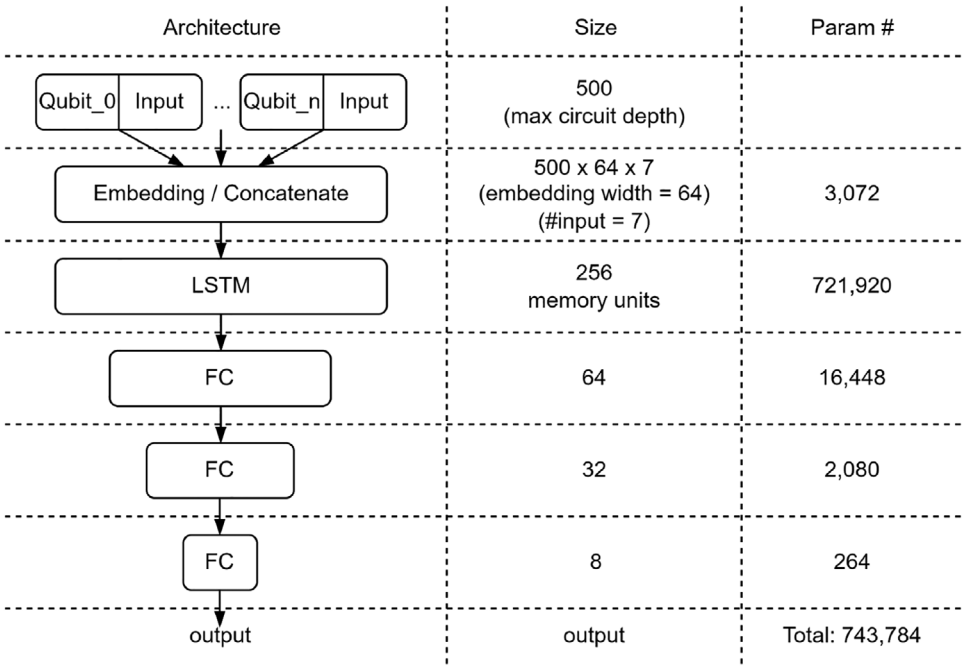

Figure 9. Neural network architecture of Q-fid. FC stands for Fully-Connected layer. The number of input layers depends on the number of qubits inside a processor. This figure shows the architecture for *ibm_nairobi* with 7 input neurons since it is a 7-qubit processor. The hardware qubit layout is shown in Figure S4 (Supporting Information).

Adder (`adder_n4`),[43] Variational Ansatz (`variational_n4`),[44] and Quantum Fourier Transform (`qft_n4`).[45]

5.1. Fidelity Prediction

Figure 12 demonstrates Q-fid's prediction performance on *ibm_nairobi*. For one given circuit, both Q-fid and `mapomatic` can only give one static fidelity prediction. Under this circumstance, comparing the performance between Q-fid and `mapomatic` using the mean hardware execution fidelity results from multiple trials shows the effect of quantum noise and how the fidelity prediction can provide a representative guideline. All the circuits used in this experiment are not optimized for physical qubit layout, so the logical qubits are placed on *ibm_nairobi* in numerical order from physical qubit 0 to 6 and stay on the same layout for all 50 trials.

Because most of the test circuits are placed on the same physical qubits, the variables in this experiment are the different quantum gates used in different algorithms. The results prove that Q-fid has learned the noise characteristics of different quantum gates on the same physical qubit from the training circuits. In contrast, the error rate based

mapomatic tends to give an underestimation, especially for high-fidelity circuits.

5.2. Transpilation Optimizations

Figure 13 shows Q-fid's ability to correctly find high-fidelity transpilation layouts on *ibmq_montreal*. Since *ibmq_montreal* is a 27-qubit processor, a circuit only using 4 physical qubits can have many layout options depending on different transpilations. For example, the hs4_n4 circuit has a depth of 34 and 13 CNOT gates, and it has a total of 2,728 different layouts shown on the x-axis, ranked from the highest fidelity layout to the lowest fidelity layout. mapomatic relies on the latest hardware calibration data to calculate the fidelity, and uses the relative fidelity differences to find the high-fidelity layouts. In comparison, Q-fid achieves similar performance without explicit hardware calibration data input, and gives better absolute fidelity prediction value so the user has more potentially good layouts to pick from. Additionally, from the full result in Figure S2 (Supporting Information), it shows that mapomatic's prediction is not sensitive to the layout variance of the deep and complex circuits, which makes it a less preferable metric for training neural networks.

In Section 5.1, the circuits are executed on fixed physical layouts, so the objective is to test if Q-fid can learn the effects of different quantum gates when they are applied to the same physical qubits. However, in this experiment, the same circuits are executed with different layouts, so the new objective here is to test if Q-fid can learn the noise characteristics when the same quantum gates are placed on different physical qubits.

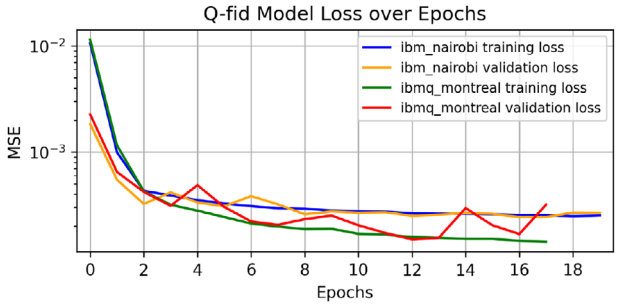

Figure 10. Training loss and validation loss for the two datasets. Training terminated on epoch 20 for *ibm_nairobi*, and on epoch 18 for *ibmq_montreal*. The final validation loss is 0.150E-3 for *ibm_nairobi*, and 0.243E-3 for *ibmq_montreal*.

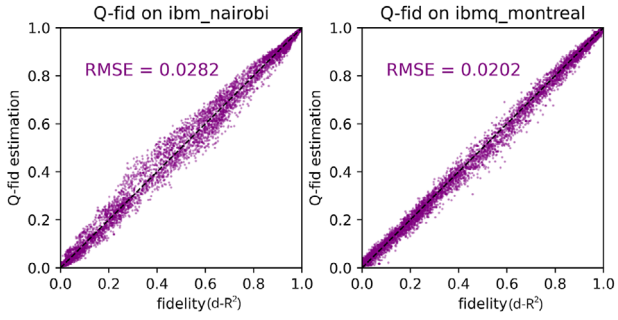

Figure 11. Scatter plot of real fidelity vs. predicted fidelity of Q-fid running on the test set. The test set for *ibm_nairobi* contains 8265 circuits, and the test set for *ibmq_montreal* contains 9140 circuits.

5.3. Noise-Adaptive Training

Figure 14 demonstrates how Q-fid adapts to *ibm_nairobi*'s device variance on different dates. The same experiment in section 5.1 is performed again, but the noise model of *ibm_nairobi* is changed to Nov. 18, 2022, three days later than the first set of predictions. The new noise model has a slightly worse fidelity performance than the previous one, which is shown to the right of the blue vertical line. For the new noise model, Q-fid's prediction is 5.31× more accurate than mapomatic on average, with the most accurate one being 42.0× better.

We envision that Q-fid can be constantly learning in parallel with the execution of QCs inside a quantum processor, so that it can always give predictions based on the most recent hardware characteristics. For this experiment, 100 additional RB circuits are randomly generated according to Section 4.1, then executed on *ibm_nairobi* with the new noise model. These new execution data go under the same training process as described in Section 4.2 to retrain Q-fid. For different processors, the number of circuits needed for retraining might vary. However, because Q-fid can use the output of any historical workload to update its internal parameters, it can be updated directly and continuously in parallel with the processor's normal workflow. In contrast, mapomatic needs to use the latest gate error rates to give updated predictions, which requires the processor to perform individual RB experiments for single-qubit gates, two-qubit gates, and the subsequent data-fitting calculations.[15] This process interrupts normal workflow and the prediction accuracy entirely depends on how recently the calibration jobs are performed.

5.4. Summary

Our comparison shows that although mapomatic is doing well in following the trend of fidelity degradation due to heavy noise, it gives an underestimation which causes the user to mistakenly think the circuit is too noisy to execute, but the output result will be relatively acceptable. An overview of the result comparison between the mean noisy fidelity and the predictions from Q-fid/mapomatic is shown in **Figure 15**. The RMSE of Q-fid's prediction compared with the mean noisy fidelity ranges from 0.003 to 0.182 with an average of 0.0515. On the other hand, mapomatic's fidelity estimation has an average RMSE of 0.142, with a minimum RMSE equal to 0.0424 and a maximum RMSE of 0.284. On the Quantum Walks algorithm (quantumwalks_n2),[46] Q-fid's prediction is 24.7× more accurate than mapomatic. On *ibmq_montreal*, when finding the high-fidelity layouts, Q-fid correctly finds the top 10% of high-fidelity circuit layouts for all 25 algorithms. Within those top 10% of layouts, Q-fid's predicted fidelity has an average RMSE of 0.0252, up to 32.8× more accurate than mapomatic.

6. Related Work

Randomized Benchmarking is one of the earliest experiments developed to characterize quantum operation error rates,[15,22,47]

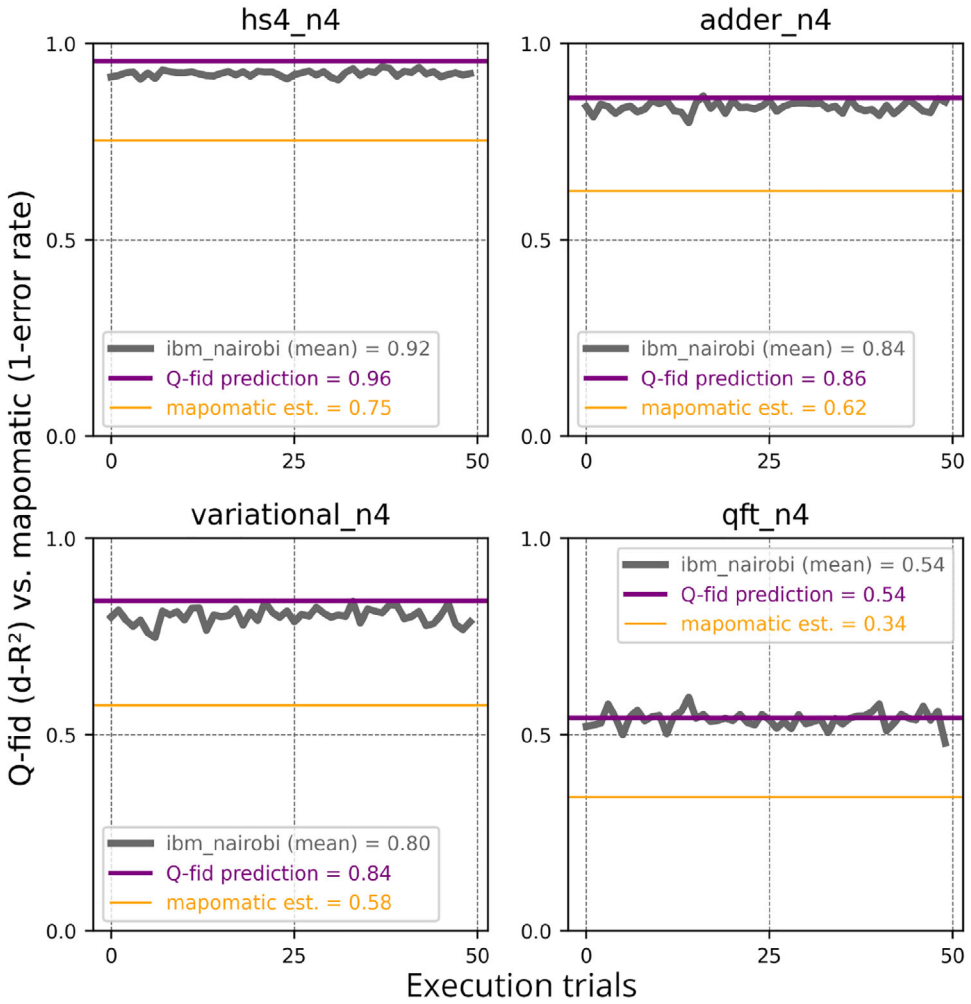

Figure 12. Q-fid's fidelity prediction (purple) compared with `mapomatic` (orange) on *ibm_nairobi*. Due to the probabilistic nature of quantum noise, the fidelity for the same circuit oscillates when executed with the same device noise model (gray). Each circuit is executed with 1024 shots to produce a noisy output distribution, then the mean fidelity is calculated from 50 noisy distributions (x-axis) as d-R^2. Q-fid accurately predicts the mean fidelity of each circuit.

and remains the most commonly used tool for this purpose today. The latest works focus on improving Randomized Benchmarking to support more quantum operations,[48] and make the experiment more flexible for larger circuits consisting of many qubits.[49,50]

A QC must be carefully implemented on real hardware to retrieve useful measurements. Early research concentrated on circuit compilation techniques to improve the fidelity of the output. For example, using gate scheduling to reduce the number of physical quantum gates or rerouting the CNOT connections to minimize SWAP gates.[51–54] Recently, the research direction has shifted to hardware-specific optimization and noise-aware qubit layout.[10,55–57] Due to the constantly changing device calibration data, quantum circuits have to be transpiled according to the architecture and noise characteristics of the processor in order to achieve the best performance.

Fidelity estimation is an emerging research field in quantum computing. Although it has been proved that estimating the final fidelity from the QC itself is hard in general, many works have demonstrated using quantum algorithms to attack this problem and achieving exponential speedup.[58,59] Other approaches like statistical estimation and polynomial fitting were also investigated.[60,61]

Using machine learning for fidelity estimation is still a very new area of research.[61] proposed a shallow neural network to directly estimate the fidelity of the quantum states, although the quantum states are not prepared with a QC. For our work, we focus on the holistic view of quantum states in circuit model quantum computing. In Ref. [62], the viability of using machine learning to predict the state fidelity from circuit representation was proved. This work utilizes a Convolutional Neural Network (CNN) for feature extraction and models the input circuit using integer encoding. The CNN architecture is heavier than our LSTM architecture, and the circuit-to-integer encoding also requires one extra step to implement. In Q-fid, this encoding is automatically handled by the text tokenizer.[63] also uses a CNN for feature extraction, but the number of parameters for a 3×3 circuit is almost 12× more compared with Q-fid's LSTM

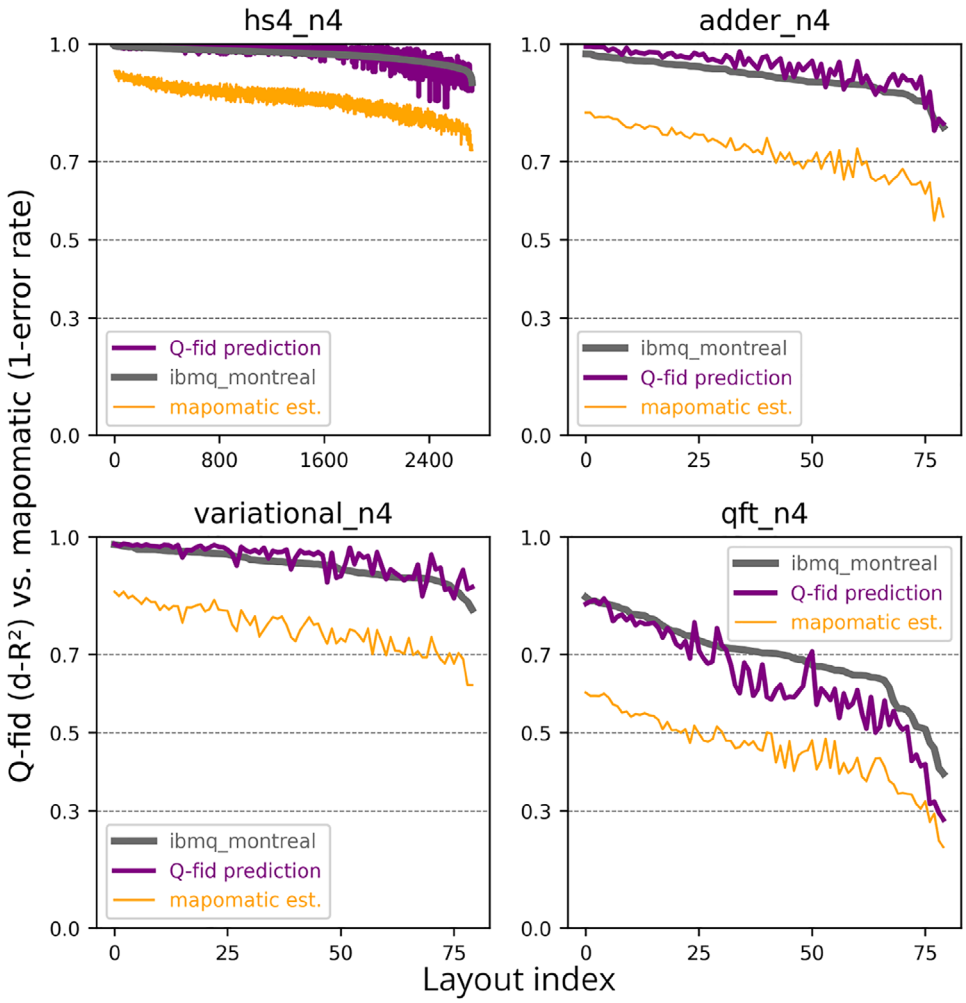

Figure 13. Q-fid's fidelity prediction (purple) for different circuit layouts on *ibmq_montreal*. The x-axis is the index of layouts for the quantum circuit, the y-axis shows different d-R^2 fidelity score regions listed in Table 1. The order of the data is sorted from the highest fidelity layout to the lowest fidelity layout according to the fidelity reported by *ibmq_montreal* (gray). Q-fid accurately tracks the fidelity variance of every layout for each circuit.

architecture, rendering future scalability issues. Recent works are exploring different learning methods.[64] proposes a multimodel deep learning method where they utilize two neural networks to learn from the measurement modality and circuit-encoded modality separately. In Q-fid, the QC is explicitly fed into the neural network and the measurement information is implicitly learned by the network through the corresponding d-R^2 score. Another work[65] implements the graph transformer for the same fidelity prediction task, but since the input QC is modeled like an image, a deep QC will produce a very long rectangle image and the performance needs to be investigated.

7. Conclusion and Outlook

We present the Q-fid system to accurately predict the fidelity of a quantum circuit running on a real NISQ processor. We show that the performance of NISQ processors is easily affected by external noise, so with Q-fid's fidelity prediction we can help save quantum computing resources by optimizing circuit layout and reducing execution shots. Q-fid uses LSTM to learn the noise properties of the qubit and the relationship between quantum gates, without the need for any separate input of hardware calibration data and gate error rates. A novel method to model the quantum circuits using text labels was presented, and the full training workflow was introduced. We apply the d-R^2 metric to intuitively quantify the fidelity of a noisy quantum circuit. Based on this metric, we also showed how to generate a training circuit dataset using the Randomized Benchmarking circuits. We compare Q-fid's performance with `mapomatic`, and the results prove that Q-fid can effectively learn the characteristics of different qubits, gates, and the structure of quantum circuits.

Future improvements of Q-fid can focus on the neural network component. The structure of the neural networks can be adjusted and optimized to use fewer layers of parameters, and different tokenizer configurations can be investigated to see how they affect Q-fid's prediction accuracy.[66,67] Because the QC is treated as text inputs in Q-fid, various new LSTM implementations or Recurrent Neural Network (RNN) architectures can also replace the standard LSTM used in this work to improve performance. The goal of Q-fid is to help enhance the usability of

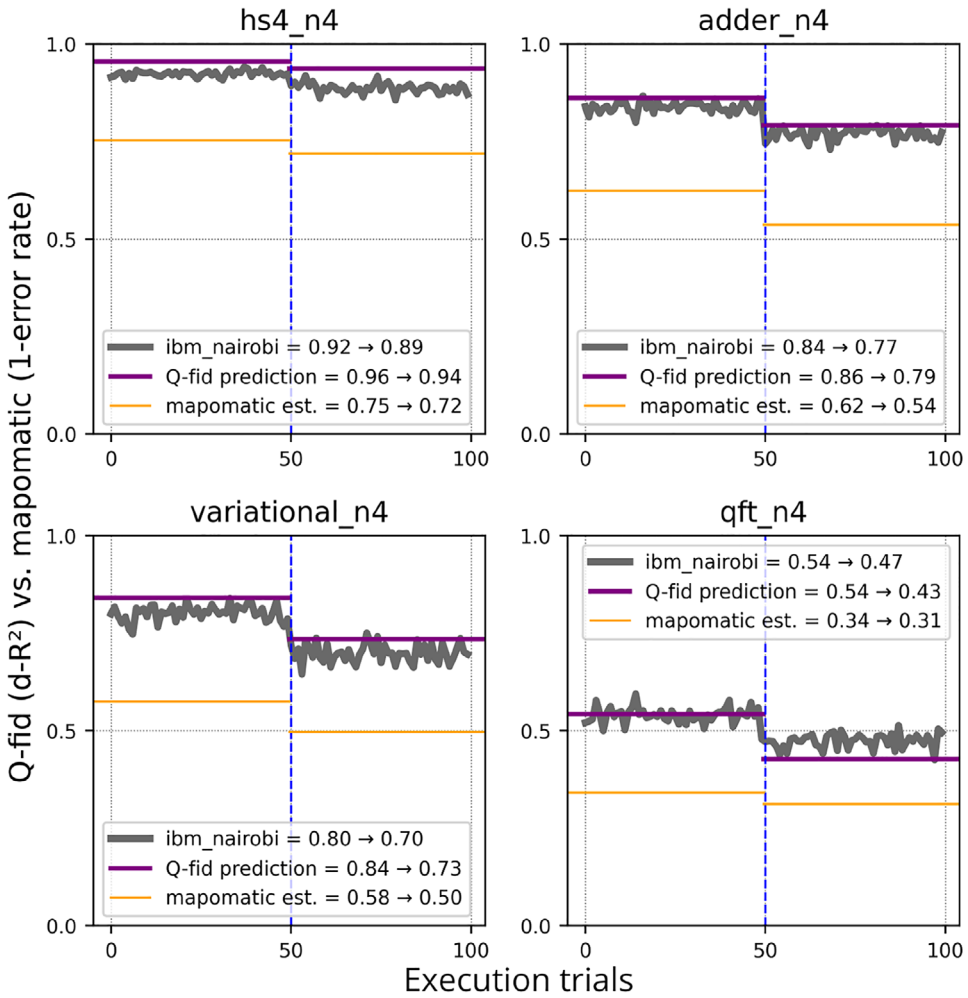

Figure 14. Q-fid's fidelity prediction (purple) compared with `mapomatic` (orange) on *ibm_nairobi*. Again, the fidelity oscillates (gray) due to the probabilistic nature of quantum noise. The first 50 trials (left of the blue line) are performed on Nov. 15, 2022, which is the same data as in Figure 12. The next 50 trials (right of the blue line) were performed on Nov. 18, 2022, which shows that *ibm_nairobi*'s performance is slightly worse due to different noise patterns. Q-fid adapts to this new noise pattern and it only takes 100 new RB circuits to retrain.

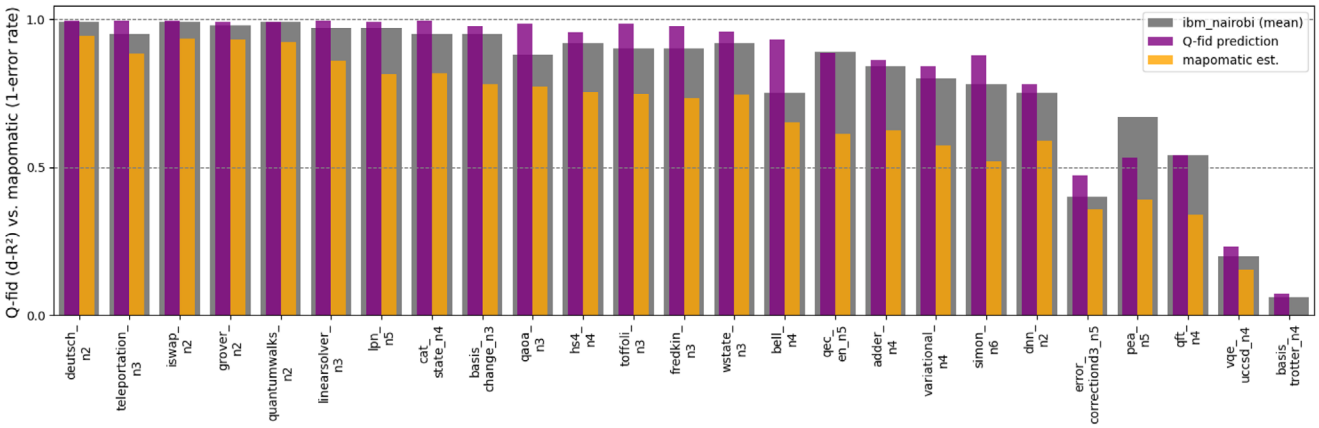

Figure 15. Result comparison between the mean noisy fidelity from *ibm_nairobi* and the predictions from Q-fid/`mapomatic`, the mean fidelity for each circuit is calculated from 50 noisy outputs as d-R^2. The RMSE of Q-fid's prediction compared with the mean noisy fidelity ranges from 0.003 to 0.182. On the other hand, `mapomatic`'s prediction has a minimum RMSE of 0.0424 and a maximum RMSE of 0.284.

current NISQ devices, where the number of qubits is limited and requires classical optimization to overcome noise problems. In the future when fault-tolerant qubits are realized, we will likely need more sophisticated metrics to evaluate the fidelity of QC outputs, due to the exponentially growing problem space accompanied by the increase of qubit counts.

8. Data and Code Availability

The full results and raw data from Section 5 are available in Figure S1, S2, S3 and Table S1, S2 (Supporting Information). The layout of the computing devices and their characteristics at the time of the experiments is available in Figure S4 and Table S3, S4, S5 (Supporting Information). The datasets used for training Q-fid are available at https://www.kaggle.com/datasets/ykmaoykmao/Q-fid-datasets. The code for generating the datasets is available at https://github.com/yikaimao/Q_fid.

Supporting Information

Supporting Information is available from the Wiley Online Library or from the author.

Acknowledgements

This work was supported by JST COI-NEXT (JPMJPF2221), JST SPRING (JPMJSP2123), and JSPS KAKENHI (JP24K20843). The authors acknowledge the use of IBM Quantum services for this work. The views expressed are those of the authors, and do not reflect the official policy or position of IBM or the IBM Quantum team.

Conflict of Interest

The authors declare no conflict of interest.

Data Availability Statement

The data that support the findings of this study are openly available in Kaggle at https://doi.org/10.34740/kaggle/ds/3474987, reference number 3474987.

Keywords

long short term memory, neural networks, quantum computing, quantum circuit

Received: January 11, 2025
Revised: March 4, 2025
Published online:

[1] D. R. Simon, *SIAM J. Comput.* **1997**, *26*, 1474.
[2] C. Wang, X. Li, H. Xu, Z. Li, J. Wang, Z. Yang, Z. Mi, X. Liang, T. Su, C. Yang, G. Wang, W. Wang, Y. Li, M. Chen, C. Li, K. Linghu, J. Han, Y. Zhang, Y. Feng, Y. Song, T. Ma, J. Zhang, R. Wang, P. Zhao, W. Liu, G. Xue, Y. Jin, H. Yu, *arXiv* **2021**.
[3] A. Ortu, A. Holzäpfel, J. Etesse, M. Afzelius, *npj Quantum Inf.* **2022**, *8*, 1.
[4] J. Preskill, *Quantum* **2018**, *2*, 79.
[5] Z. Cai, R. Babbush, S. C. Benjamin, S. Endo, W. J. Huggins, Y. Li, J. R. McClean, T. E. O'Brien, *arXiv* **2022**.
[6] K. Temme, S. Bravyi, J. M. Gambetta, *Phys. Rev. Lett.* **2017**, *119*, 180509.
[7] Y. Ueno, M. Kondo, M. Tanaka, Y. Suzuki, Y. Tabuchi, in *2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, **2022**, pp. 274–287.
[8] H.-L. Huang, X.-Y. Xu, C. Guo, G. Tian, S.-J. Wei, X. Sun, W.-S. Bao, G.-L. Long, *arXiv* **2022**.
[9] Y. Ueno, M. Kondo, M. Tanaka, Y. Suzuki, Y. Tabuchi, in *2021 58th ACM/IEEE Design Automation Conference (DAC)*, **2021**, pp. 451–456.
[10] M. Y. Siraichi, V. F. dos Santos, C. Collange, F. M. Q. Pereira, in *Proceedings of the 2018 International Symposium on Code Generation and Optimization*, ACM, **2018** https://doi.org/10.1145/3168822.
[11] S. S. Tannu, M. K. Qureshi, in *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, ACM, **2019** https://doi.org/10.1145/3297858.3304007.
[12] S. Bravyi, O. Dial, J. M. Gambetta, D. Gil, Z. Nazario, *J. Appl. Phys.* **2022**, *132*, 160902.
[13] C. Piveteau, D. Sutter, *arXiv* **2023**.
[14] P. D. Nation, M. Treinish, *arXiv* **2022**.
[15] E. Magesan, J. M. Gambetta, J. Emerson, *Phys. Rev. Lett.* **2011**, *106*, 18.
[16] T. Proctor, K. Rudinger, K. Young, E. Nielsen, R. Blume-Kohout, *Nat. Phys.* **2021**, *18*, 75.
[17] IBM, IBM Quantum, **2021**, https://quantum-computing.ibm.com/.
[18] M. A. Nielsen, I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition*, Cambridge University Press, **2010**.
[19] E. Bernstein, U. Vazirani, *SIAM J. Comput.* **1997**, *26*, 1411.
[20] T. Häner, D. S. Steiger, K. Svore, M. Troyer, *Quantum Sci. Technol.* **2018**, *3*, 020501.
[21] A. MATSUO, S. YAMASHITA, D. J. EGGER, *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.* **2023**, E106.A, 1424.
[22] E. Magesan, J. M. Gambetta, J. Emerson, *Phys. Rev. A* **2012**, *85*, 4.
[23] S. Bravyi, D. Maslov, *IEEE Trans. Inf. Theory* **2021**, *67*, 4546.
[24] S. Hochreiter, J. Schmidhuber, *Neural Comput.* **1997**, *9*, 1735.
[25] H. P. others, *IEEE/ACM Transactions on Audio, Speech, and Language Processing* **2016**, *24*, 694.
[26] D. Chicco, M. J. Warrens, G. Jurman, *PeerJ Comput. Sci.* **2021**, *7*, e623.
[27] J. Liu, H. Zhou, Reliability modeling of nisq- era quantum computers in *2020 IEEE International Symposium on Workload Characterization (IISWC)*, **2020**, pp. 94–105.
[28] D. Oliveira, E. Giusto, B. Baheri, Q. Guan, B. Montrucchio, P. Rech, *arXiv* **2021**.
[29] T. Patel, D. Silver, D. Tiwari, *arXiv* **2022**.
[30] E. Farhi, J. Goldstone, S. Gutmann, *arXiv* **2014**.
[31] A. Peruzzo, J. McClean, P. Shadbolt, M.-H. Yung, X.-Q. Zhou, P. J. Love, A. Aspuru-Guzik, J. L. O'Brien, *Nat. Commun.* **2014**, *5*, 1.
[32] A. Deshpande, P. Niroula, O. Shtanko, A. V. Gorshkov, B. Fefferman, M. J. Gullans, *PRX Quantum* **2022**, *3*, 040329.
[33] D. Aharonov, M. Ben-Or, R. Impagliazzo, N. Nisan, *arXiv* **1996**.
[34] S. Boixo, S. V. Isakov, V. N. Smelyanskiy, R. Babbush, N. Ding, Z. Jiang, M. J. Bremner, J. M. Martinis, H. Neven, *Nat. Phys.* **2018**, *14*, 595.
[35] S. Boixo, V. N. Smelyanskiy, H. Neven, *arXiv* **2017**.
[36] K. Tamura, Y. Shikano, in *International Symposium on Mathematics, Quantum Theory, and Cryptography*, **2020**, pp. 17–37.
[37] M. Dwarampudi, N. V. S. Reddy, *arXiv* **2019**.
[38] T. Mikolov, K. Chen, G. Corrado, J. Dean, *arXiv* **2013**.

[39] D. Gottesman, in *Group theoretical methods in physics*, **1998**.
[40] A. Li, S. Stein, S. Krishnamoorthy, J. Ang, *arXiv* **2020**.
[41] Qiskit_community, mapomatic, **2023**, https://github.com/qiskit-community/mapomatic.
[42] M. Treinish, J. Gambetta, S. Thomas, P. Nation, qiskit-bot, P. Kassebaum, D. M. Rodríguez, S. González, J. Lishman, S. Hu, L. Bello, J. Garrison, K. Krsulich, J. Huang, J. Yu, M. Marques, J. Gacon, D. McKay, E. Arellano, J. Gomez, L. Capelluto, Travis-S-IBM, A. Mitchell, A. Panigrahi, lerongil, R. I. Rahman, S. Wood, T. Itoko, A. Pozas-Kerstjens, C. J. Wood, Qiskit/qiskit: Qiskit 0.42.0, **2023**, https://zenodo.org/record/2573505.
[43] A. JavadiAbhari, S. Patil, D. Kudrow, J. Heckey, A. Lvov, F. T. Chong, M. Martonosi, ScaffCC: A Framework for Compilation and Analysis of Quantum Computing Programs, in *Proceedings of the 11th ACM Conference on Computing Frontiers*, ACM, **2014** https://doi.org/10.1145/2597917.2597939.
[44] J. R. McClean, J. McClean, N. Rubin, K. Sung, I. Kivlichan, X. Bonet-Monroig, Y. Cao, C. Dai, E. Fried, C. Gidney, B. Gimby, P. Gokhale, T. Häner, T. Hardikar, V. Havlíček, O. Higgott, C. Huang, J. Izaac, Z. Jiang, X. Liu, S. McArdle, M. Neeley, T. O'Brien, B. O'Gorman, I. Ozfidan, M. D. Radin, J. Romero, N. P. D. Sawaya, B. Senjean, K. Setia, *Quantum Sci. Technol.* **2020**, *5*, 034014.
[45] A. Cross, A. Javadi-Abhari, T. Alexander, N. Beaudrap, L. S. Bishop, S. Heidel, C. A. Ryan, P. Sivarajah, J. Smolin, J. Gambetta, B. Johnson, *ACM Transactions on Quantum Computing* **2022**, *3*, 1.
[46] https://github.com/raffmiceli, Qiskit code to simulate quantum walks on graphs with up to 4 nodes, https://github.com/raffmiceli/Quantum_Walks, **2020**, (accessed: 2023).
[47] E. Knill, et al., *Phys. Rev. A* **2008**, *77*, 1.
[48] D. C. McKay, S. Sheldon, J. A. Smolin, J. M. Chow, J. M. Gambetta, *Phys. Rev. Lett.* **2019**, *122*, 20.
[49] T. Proctor, S. Seritan, K. Rudinger, E. Nielsen, R. Blume-Kohout, K. Young, *Phys. Rev. Lett.* **2022**, *129*, 15.
[50] J. Helsen, X. Xue, L. M. K. Vandersypen, S. Wehner, *arXiv* **2019**.
[51] K. E. C. Booth, M. Do, J. C. Beck, E. Rieffel, D. Venturelli, J. Frank, *arXiv* **2018**.
[52] G. G. Guerreschi, J. Park, *Quantum Sci. Technol.* **2018**, *3*, 045003.
[53] A. Paler, A. Zulehner, R. Wille, *arXiv* **2021**.
[54] G. Li, Y. Ding, Y. Xie, *arXiv* **2019**.
[55] S. S. Tannu, M. K. Qureshi, *arXiv* **2018**.
[56] S. Nishio, Y. Pan, T. Satoh, H. Amano, R. V. Meter, *ACM J. Emerg. Technol. Comp. Syst.* **2020**, *16*, 1.
[57] S. S. Tannu, M. Qureshi, Ensemble of diverse mappings: Improving reliability of quantum computers by orchestrating dissimilar mistakes, in *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO '52. Association for Computing Machinery, New York, NY, USA, ISBN 9781450369381, **2019**, pp. 253–265.
[58] A. Gilyén, A. Poremba, *arXiv* **2022**.
[59] Q. Wang, Z. Zhang, K. Chen, J. Guan, W. Fang, J. Liu, M. Ying, *IEEE Trans. Inf. Theory* **2023**, *69*, 273.
[60] X.-D. Yu, J. Shang, O. Gühne, *Adv. Quantum Technol.* **2022**, *5*, 2100126.
[61] J. Liu, H. Zhou, Reliability modeling of nisq- era quantum computers, in *2020 IEEE International Symposium on Workload Characterization (IISWC)*, **2020**, pp. 94–105.
[62] N. Elsayed Amer, W. Gomaa, K. Kimura, K. Ueda, A. El-Mahdy, *EPJ Quant. Technol.* **2022**, *9*, 31.
[63] A. Vadali, R. Kshirsagar, P. Shyamsundar, G. N. Perdue, *Quantum Mach. Intell.* **2022**, *6*, 1.
[64] Y. Qian, Y. Du, Z. He, M.-H. Hsieh, D. Tao, *Phys. Rev. Lett.* **2024**, *133*, 130601.
[65] H. Wang, P. Liu, J. Cheng, Z. Liang, J. Gu, Z. Li, Y. Ding, W. Jiang, Y. Shi, X. Qian, D. Z. Pan, F. T. Chong, S. Han, *arXiv* **2022**.
[66] T. Hiraoka, T. Iwakura, *arXiv* **2024**.
[67] V. Zouhar, C. Meister, J. L. Gastaldi, L. Du, M. Sachan, R. Cotterell, *arXiv* **2023**.