

## A. Artifact Appendix

### A.1 Abstract

This document provides artifact descriptions for the ASPLOS 2021 paper “QRAFT: Reverse Your Quantum Circuit and Know the Correct Program Output”. The artifacts can be divided into three categories: (1) Raw data: circuit metadata and output generated as a direct result of running quantum circuits. (2) Processed and Trained data: the data processed to be fed as input to the machine learning model training, as well as the output data of testing samples using the trained model. (3) Tools: code and scripts used for running circuits on quantum computers, processing the output, as well as training models and generating the final output (prediction of state probabilities). The artifacts, required hardware and software dependencies, experimentation, and evaluation characteristics are described below. The artifacts are publicly available at the following DOI link: <https://doi.org/10.5281/zenodo.4527305>.

### A.2 Artifact check-list (meta-information)

- **Program:** The paper uses six quantum algorithm benchmarks: Bernstein-Vazirani Algorithm (BV), Deutsch-Jozsa Algorithm (DJ), Grover’s Algorithm (GRV), Quantum Fourier Transform (QFT), Quantum Phase Estimation (QPE), and Simon’s Algorithm (SMN). The paper also generates random algorithm circuits for training. The Python (version 3.7) scripts to run these algorithms are provided in the “code\_execute” folder. Running these scripts on quantum computers requires access to the IBM Quantum Experience cloud.

The paper also uses different machine learning (ML) models for training. To determine which ML model performs the best, we use five ML models: ensemble of decision trees (EDT),  $k$  nearest neighbors ( $kNN$ ), simple feed-forward neural network (input is only fed into the first layer and each subsequent layer has a connection from only the previous layer) cascade-forward neural network (the input is directly fed into also the layers), and pattern-detection neural network (similar to feed-forward network but the output layer gives probability of membership to different classes). The hyperparameters of all models (e.g., number of trees and boosting/bagging method for the EDT, number of nearest neighbors for  $kNN$ ) are optimized using Bayesian optimization. The MATLAB (version R2019) scripts to run these algorithms are provided in the “code\_train” folder.

- **Compilation:** The Python-based Qiskit (version 0.23.1) transpiler (quantum circuit compiler) is used to generate optimal circuit maps. An optimal circuit map for a given algorithm on a given computer may vary at different times as the coherence times and operation errors vary temporally (these errors are noted when the computers are calibrated daily and this information is used when generating an optimal circuit map).
- **Transformations:** Not required.
- **Binary:** Not required.
- **Model:** See above under “Program”.
- **Data set:** The dataset under the “data\_experimental” folder includes the following. It includes a training dataset of  $\approx 1400$  random forward circuits of varying widths, depths, and operations across six different IBM quantum computers. Each circuit is run ten times in the forward mode and ten times in the forward + reverse mode to obtain the observed state probabilities and other related features (described next). All of the runs are organized by date. It also includes the runs for different algorithms (files are prefixed with the abbreviation of the algorithm) and runs for circuits without any optimizations (files are prefixed with “lq”).

The above raw data is processed using scripts in the “code\_process” folder and input data is generated for training the ML models (the processed data is in the “data\_processed” folder). Each circuit has multiple states and because a large fraction of the states tend to have zero probability, they can end up being not recorded in any of the trials of the forward and forward + reverse circuit. Note that including them in the training can inadvertently boost the accuracy of the prediction model even if it always predicts a true state probability of zero. Therefore, we eliminate such states from the training as they can be directly assumed to have zero true probability. After the filtering process, we end up with  $\approx 10,000$  states for training (also referred as samples). The “data\_trained” folder contains the output data of the test samples (state probability predictions) generated by different trained ML models.

- **Run-time environment:** No OS-specific artifacts.
- **Hardware:** Access to the IBM Quantum Experience cloud is required. Multiple superconducting-qubit quantum computers available are used for the evaluation. These include the five-qubit systems Vigo, Ourense, London, Burlington, and Essex, and the 15-qubit system Melbourne.
- **Run-time state:** The raw data artifacts are sensitive to the error rates of the quantum computers when the artifacts are executed. The error rates vary frequently from one run to another. For statistical significance and for capturing a wide array of error characteristics, the algorithms are run on multiple computers over multiple days.
- **Execution:** The original experiments of running and collecting data for training using random circuits, algorithm circuits, and no-optimization circuits on six quantum computers took over one month. The data processing and training steps for the Ensemble of Decision Trees model and other models requires 5-8 hours. The processing was performed on MacOS Catalina 10.15.7 and the training was performed on the MATLAB (MathWorks) cloud.
- **Metrics:** The main metric for evaluation is the state error. Consider a two-qubit algorithm with true probabilities  $p_t(|00\rangle) = 0.1$ ,  $p_t(|10\rangle) = 0.2$ ,  $p_t(|01\rangle) = 0.3$ , and  $p_t(|11\rangle) = 0.4$ . When the algorithm is run using a circuit map, the observed probabilities are  $p_o(|00\rangle) = 0.15$ ,  $p_o(|10\rangle) = 0.15$ ,  $p_o(|01\rangle) = 0.5$ , and  $p_o(|11\rangle) = 0.2$ . We define the state error as the absolute difference between the observed probability and the true probability. In this example, the state errors are  $e(|00\rangle) = |0.1 - 0.15| = 0.05$  or 5%,  $e(|10\rangle) = 5\%$ ,  $p_o(|01\rangle) = 20\%$ , and  $p_o(|11\rangle) = 20\%$ .  
Another metric we use is the program error. It is the sum of the state errors of all output states of an algorithm divided by two. In this case, the program error is 50%. We also define the dominant state error as the state error of the states with maximum true probability. This metric is useful because for many quantum algorithms, the aim is to identify the maximum true probability state. In this example, state  $|11\rangle$  has maximum probability and has an error of 20%. Therefore, the dominant state error is 20%. Note that for some algorithms, multiple states can have equal maximum true probability. In that case, the median error of all dominant states is quoted.
- **Output:** The output of the ML models is the predicted state probabilities (current output is stored in the “data\_trained” folder). It is in csv format. Please refer to the evaluation section (Sec. 3) of the paper for all expected results.
- **Experiments:** To collect the training and algorithm data, run the Python scripts in the “code\_execute” folder (generated data

in the “data\_experimental” folder. To process the raw data, run the Python scripts in the “code\_process” folder (generates input files for ML models in the “data\_processed” folder. To train the ML models, run the MATLAB scripts in the “code\_train” folder. Note that the MATLAB classification and neural network application widgets can also be used for training. Finally, after training, the output data for test samples will be generated in the “data\_trained” folder.

- **How much disk space required (approximately)?:**  $\approx 30$  MB.
- **How much time is needed to prepare workflow (approximately)?:** 1-2 hours
- **How much time is needed to complete experiments (approximately)?:** As mentioned above under “Execution”, random circuits are run over a period of one month and each algorithm is run with multiple optimal circuit maps across three days. The data processing and training steps for the Ensemble of Decision Trees model and other models requires 5-8 hours. The processing was performed on MacOS Catalina 10.15.7 and the training was performed on the MATLAB (MathWorks) cloud.
- **Publicly available?:** Yes.  
<https://doi.org/10.5281/zenodo.4527305>
- **Code licenses (if publicly available)?:** Open-source.
- **Data licenses (if publicly available)?:** Open-source.
- **Workflow framework used?:** No.
- **Archived (provide DOI)?:** Yes.  
<https://doi.org/10.5281/zenodo.4527305>.

### A.3 Description

#### A.3.1 How to access

The artifacts can be accessed using the following link:  
<https://doi.org/10.5281/zenodo.4527305>. They can be downloaded in a ZIP format.

#### A.3.2 Hardware dependencies

Multiple superconducting-qubit quantum computers available through IBM’s quantum computing cloud are used for the evaluation. These include the five-qubit systems Vigo, Ourense, London, Burlington, and Essex, and the 15-qubit system Melbourne.

Classical computations can be performed on any CPU hardware. In our experiments, the processing was performed on MacOS Catalina 10.15.7 and the training was performed on the MATLAB (MathWorks) cloud.

#### A.3.3 Software dependencies

No specific OS or proprietary software required. The experiments were using Python 3.7, Qiskit 0.23.1 (qiskit-aer 0.7.1, qiskit-aqua 0.8.1, qiskit-ibmq-provider 0.11.1, qiskit-ignis 0.5.1, qiskit-terra 0.16.1), and MATLAB R2019 (classification and neural network application widgets). The processing was performed on MacOS Catalina 10.15.7 (some code may require changing the forward slashes in file paths to backslashes if using Windows OS) and the training was performed on the MATLAB (MathWorks) cloud.

#### A.3.4 Data sets

The dataset under the “data\_experimental” folder includes the following. It includes a training dataset of  $\approx 1400$  random forward circuits of varying widths, depths, and operations across six different IBM quantum computers. Each circuit is run ten times in the forward mode and ten times in the forward + reverse mode to obtain the observed state probabilities and other related features (described next). All of the runs are organized by date. It also includes the runs for different algorithms (files are prefixed with the

abbreviation of the algorithm) and runs for circuits without any optimizations (files are prefixed with “lq”).

The above raw data is processed using scripts in the “code\_process” folder and the input data is generated for training the ML models (the processed data is in the “data\_processed” folder). Each circuit has multiple states and because a large fraction of the states tends to have zero probability, they can end up being not recorded in any of the trials of the forward and forward + reverse circuit. Note that including them in the training can inadvertently boost the accuracy of the prediction model even if it always predicts a true state probability of zero. Therefore, we eliminate such states from the training as they can be directly assumed to have zero true probability. After the filtering process, we end up with  $\approx 10,000$  states for training (also referred to as samples). The “data\_trained” folder contains the output data of the test samples generated by different trained ML models.

### A.3.5 Models

#### A.4 Installation

Installation of Python, Qiskit, and MATLAB is required. Please refer to the below links for installation directions.

- (1) Python: <https://www.python.org/downloads/>
- (2) Qiskit: <https://qiskit.org/documentation/install.html>
- (3) MATLAB: <https://www.mathworks.com/help/install/>

#### A.5 Experiment workflow

(1) To collect the training and algorithm data, run the Python scripts in the “code\_execute” folder (generated data in the “data\_experimental” folder.

(2) To process the raw data, run the Python scripts in the “code\_process” folder (generates input files for ML models in the “data\_processed” folder.

(3) To train the ML models, run the MATLAB scripts in the “code\_train” folder. Note that the MATLAB classification and neural network application widgets can also be used for training. Finally, after training, the output data for test samples will be generated in the “data\_trained” folder.

#### A.6 Evaluation and expected result

Please refer to the evaluation section (Sec. 4) of the paper for all expected results. The data and scripts are provided for all the results provided in the evaluation section: (1) median state error, dominant state error, and program error of the six algorithms. (2) State error of the testing samples for random circuits. (3) State error when the ML model is trained with only a part of the features. (4) Dependence of state probability error on different state and circuit properties such as the true state probability, state error, circuit width, circuit depth, state hamming weight, and a total number of gates of each type. (5) State error when circuits are executed on the 15-qubit Melbourne computer. (6) State error when circuits are executed without any Qiskit compiler optimizations.

#### A.7 Methodology

Submission, reviewing and badging methodology:

- <https://www.acm.org/publications/policies/artifact-review-badging>
- <http://cTuning.org/ae/submission-20201122.html>
- <http://cTuning.org/ae/reviewing-20201122.html>