User Manual

VeriQR is an object-oriented tool written in C++, which was chosen in part due to the prevalence of C++/Qt in the design of GUI (graphical user interfaces) programs.

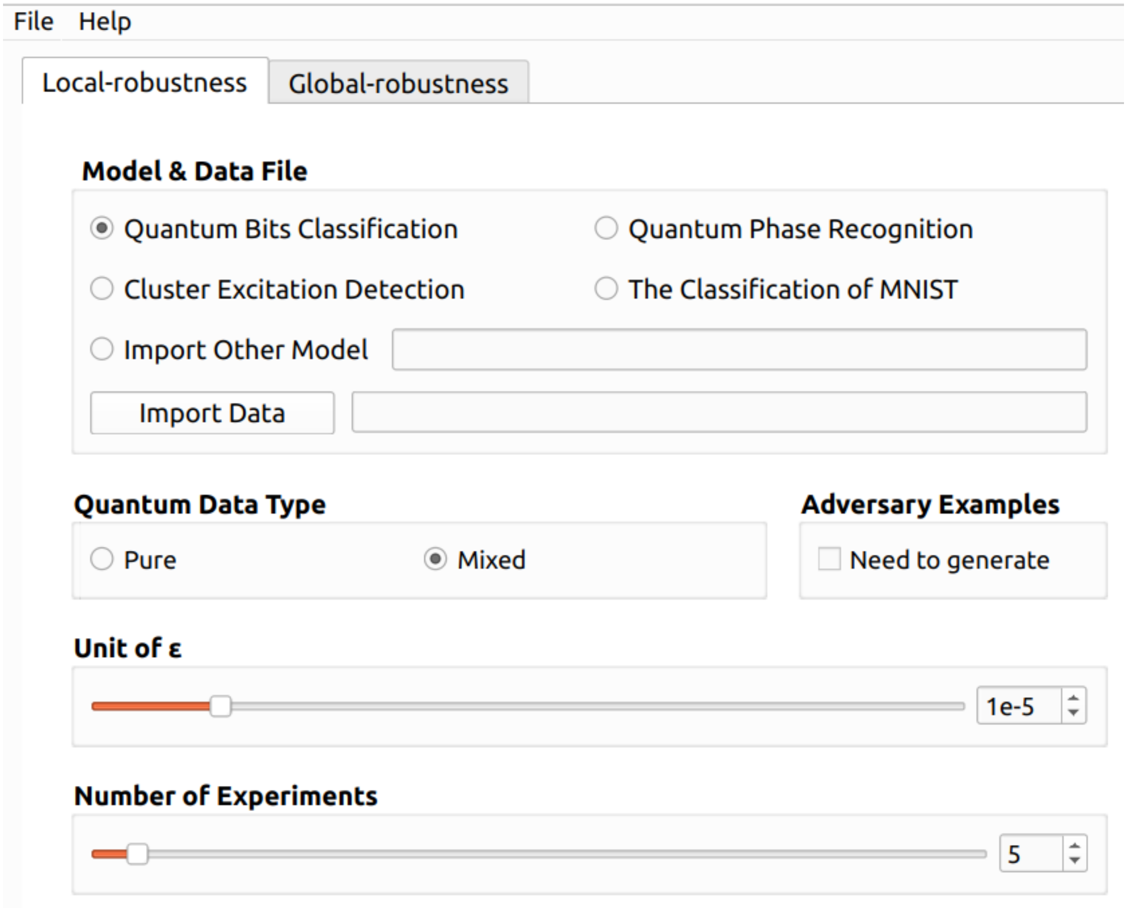
Our VeriQR application contains two parts:

- local-robustness verification (See Algorithm 1 in the paper [Robustness Verification of Quantum Classifiers]).
- global-robustness verification (See Algorithm 1 in the paper [Verifying Fairness in Quantum Machine Learning]).

Local-robustness Verification

Input

To perform local-robustness verification experiments on *VeriQR*, the inputs required for VeriQR include a quantum classifier, a measurement operator, a training dataset, a decimal parameter, the quantum data type and the number of experiments.



- The quantum classifier that users input should be well-trained, which consists of a quantum circuit with a measurement at the end. VeriQR accepts quantum classifiers of the following formats:

  (i) A `NumPy data` file (in .npz format) which the quantum circuits, the measurement operator, and the training dataset are packaged together into. This kind of NumPy data files can be directly obtained by the data of the classifiers trained on the platform --- [Tensorflow Quantum] of Google. VeriQR provides four quantum classifiers in the required format, including quantum bits classification, quantum phase recognition and cluster excitation detection from real world intractable physical problems, and the classification of MNIST.

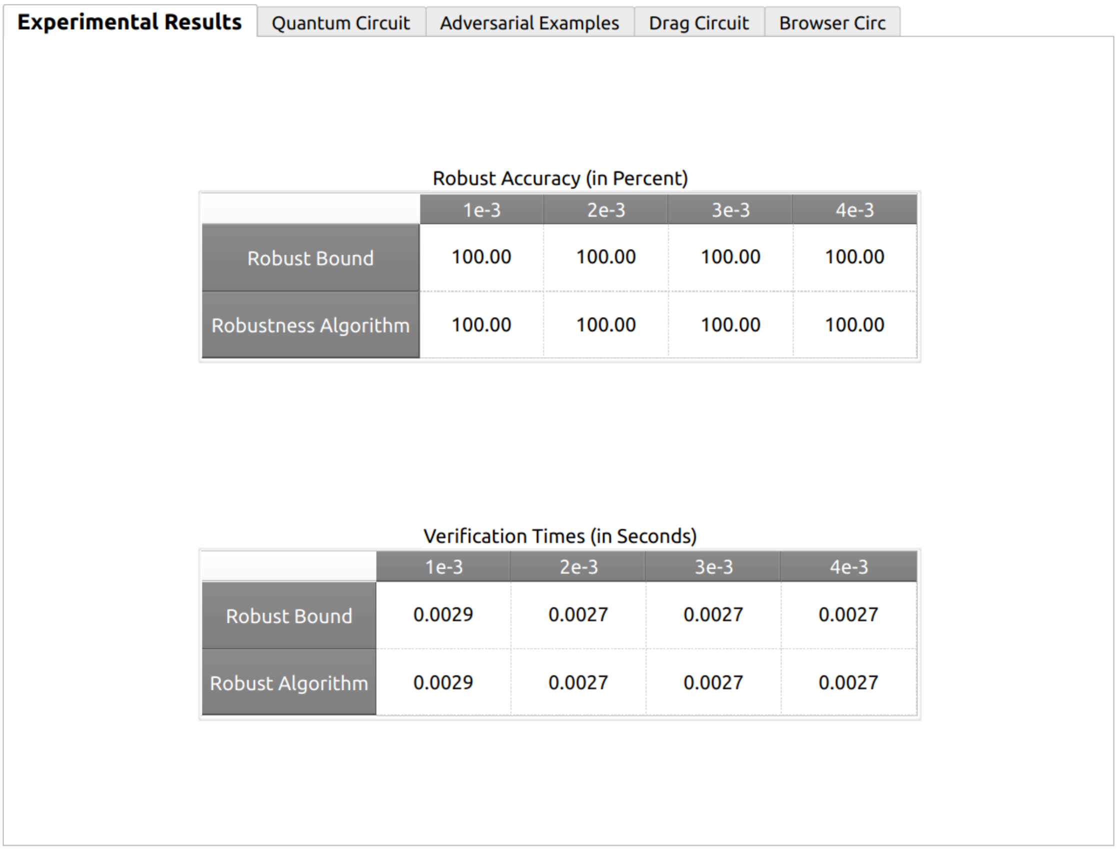
(ii) A `OpenQASM 2.0` file (in .qasm format) which represents the quantum circuit corresponding to a quantum classifier. Quantum models trained with other hybrid quantum-classical machine learning frameworks such as MindSpore, Cirq and Qiskit can be translated into this intermediate representation. For example, VeriQR provides script for the translation of MindSpore models into the .qasm format. In this case, a `NumPy data` file which contains the measurement operator and the training dataset is also required.

- The decimal parameter is the unit of the robust threshold value ϵ, which together with the number of robustness verification experiments forms ϵ for each experiment. For example, for the case where the decimal precision and the number of experiments are `1e-3` and `3`, respectively, the `1e-3`, `2e-3`, `3e-3`-robustness of the quantum classifier will be checked in turn.
- For the robustness verification of the MNIST classifier, VeriQR supports the generation of adversarial examples. Users can click the `Need to generate` checkbox to make a choice.

Output

For local-robustness verification:

- VeriQR will output whether the robustness property holds, which is reflected by the calculated robust accuracy of the quantum classifier.



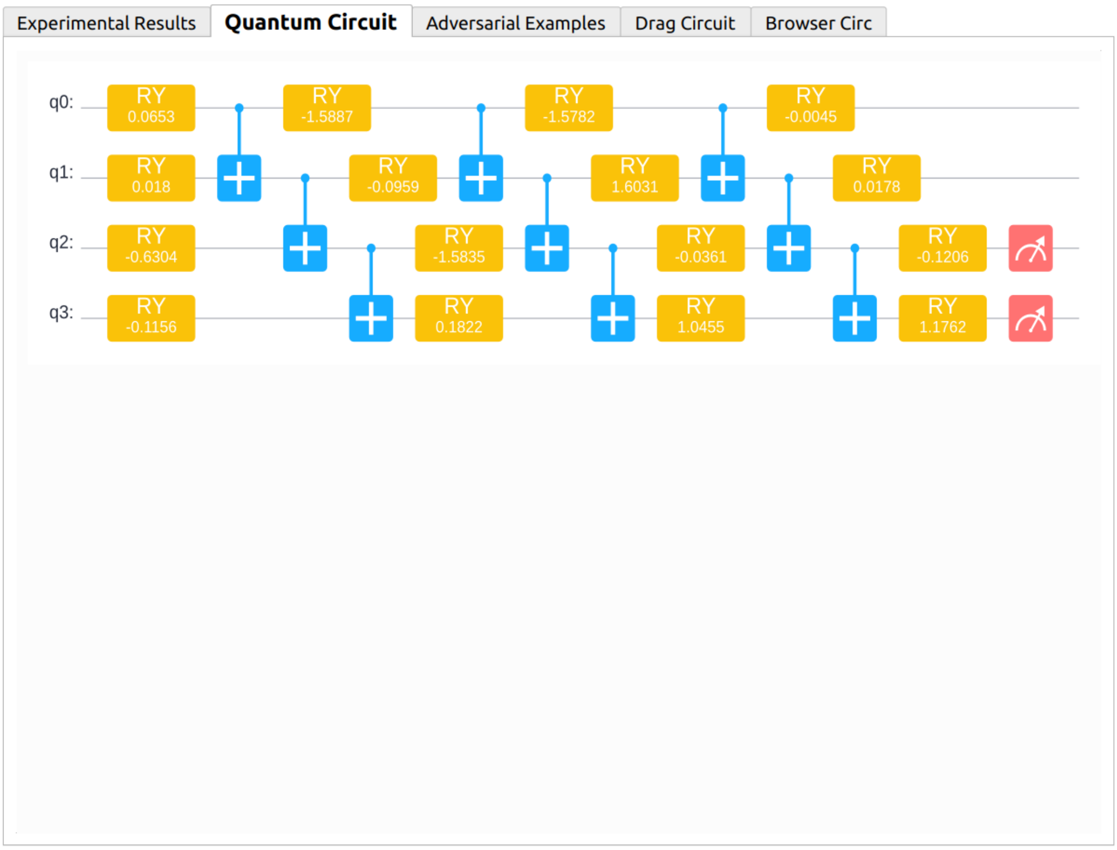
Robust Accuracy (in Percent)

	1e-3	2e-3	3e-3	4e-3
Robust Bound	100.00	100.00	100.00	100.00
Robustness Algorithm	100.00	100.00	100.00	100.00

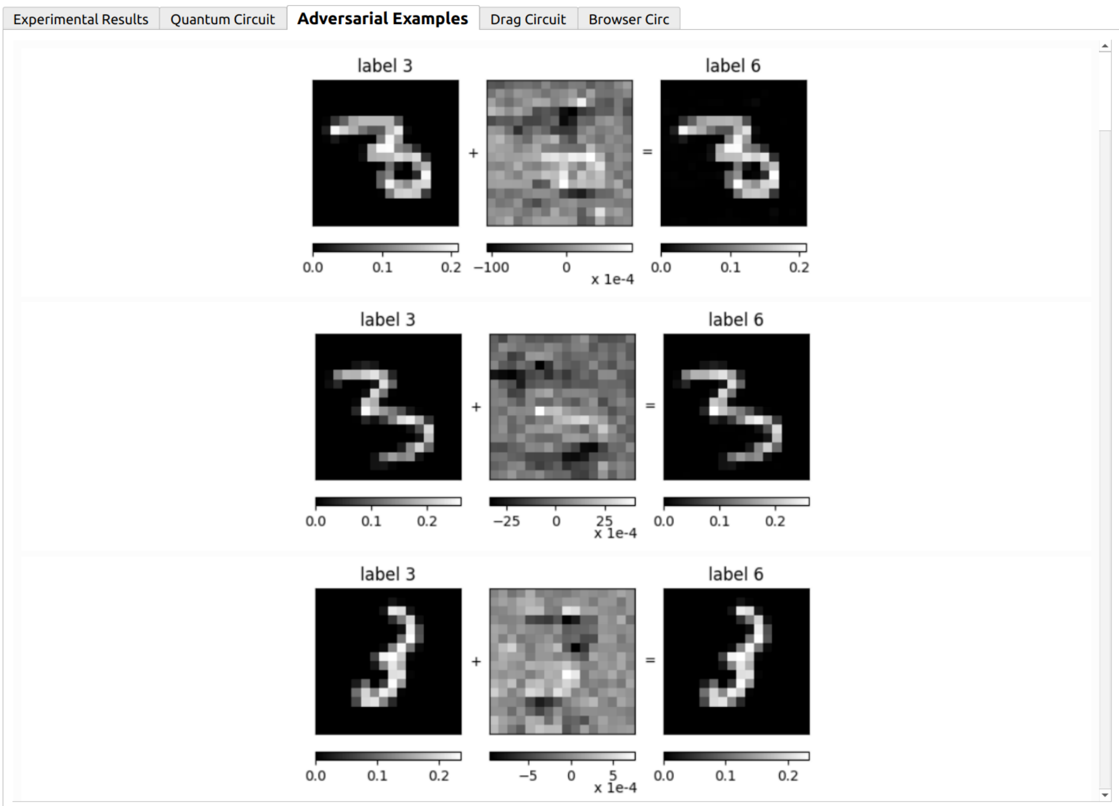
Verification Times (in Seconds)

	1e-3	2e-3	3e-3	4e-3
Robust Bound	0.0029	0.0027	0.0027	0.0027
Robust Algorithm	0.0029	0.0027	0.0027	0.0027

- Moreover, it depicts the quantum circuit corresponding to each quantum classifier in a diagram. (You can use the mouse wheel to zoom in or out of the picture.)



- Remarkably, PRODeep generates adversarial examples of the MNIST classifier and displays them in `.png` images. Here you can choose any number between 0 and 9 to generate adversarial examples.



Global-robustness Verification

////// TODO

Conduct Experiments

////// TODO

In the GUI,

- you can click the "**Import file**" button to select a NumPy data file (with the .npz suffix) that consists of a (well-trained) quantum classifier and corresponding training dataset to verify. And after setting all experiment parameters (unit of robust accuracy, number of experiments, quantum data type), click the "**run**" button to check the robustness of this classifier.
- you can also open a saved runtime information file (with the .txt suffix) by selecting "**Open a result data file**" from the File menu so that you don't need to run the program again.