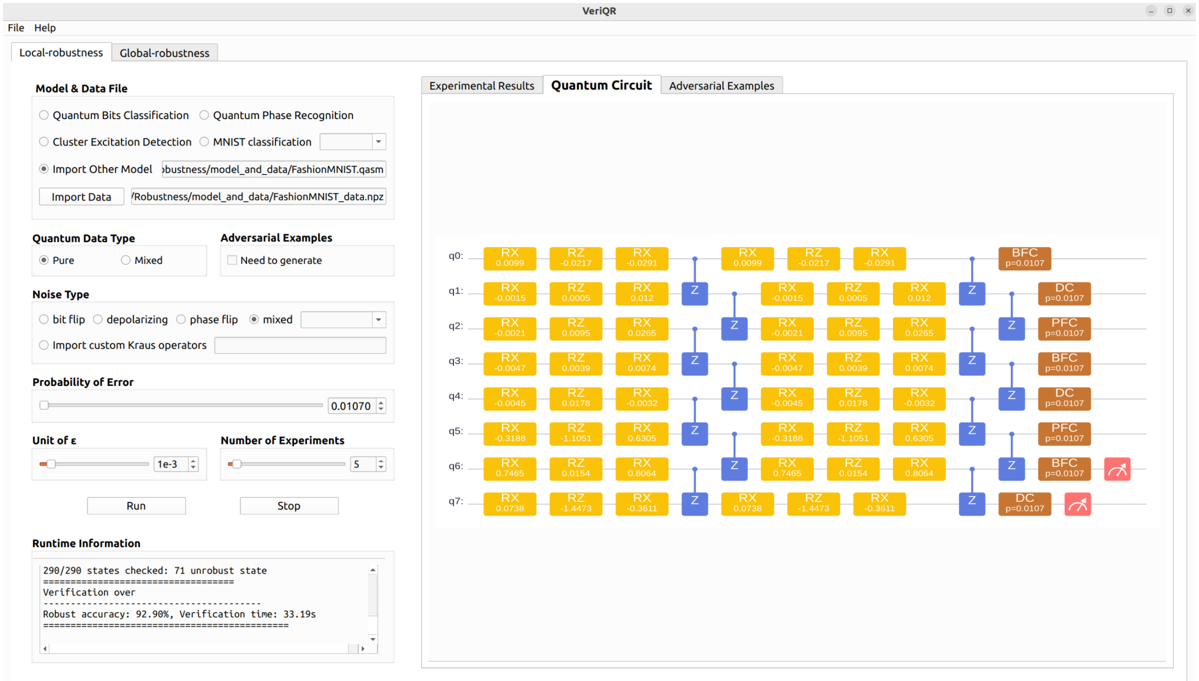
User Manual

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. It contains two parts:

Local Robustness Verification



Input

To perform local-robustness verification on *VeriQR*, the inputs required for VeriQR include:

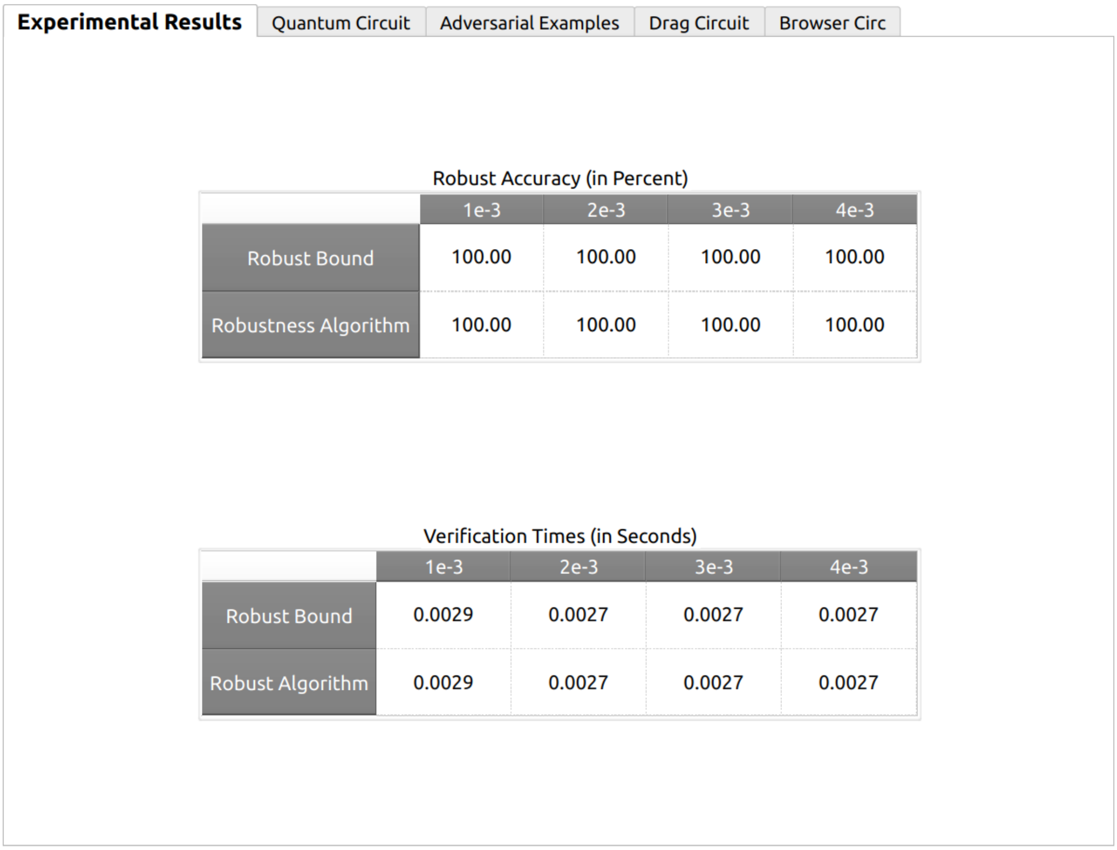
- **A well-trained quantum classifier** and **a dataset** that contains quantum states and their corresponding ground truth labels and can be sourced from either a training or testing dataset. *VeriQR* accepts a model in the following formats, each of which represents a quantum circuit with a measurement at the end of the circuit.
    1. A `NumPy data` file (`.npz` format) which a quantum circuit, quantum measurement, and training dataset are packaged together into. *VeriQR* provides four popular testing examples in the `.npz` format, including quantum bits classification, quantum phase recognition and cluster excitation detection and the classification of MNIST, catering to beginners.
    2. A `OpenQASM 2.0` file (in .qasm format) which expresses the quantum circuit corresponding to a QML model to be checked. OpenQASM 2.0 is an IBM-introduced format widely adopted in the quantum computing community for constructing quantum circuits. QML 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 a measurement operator and a dataset is also required.
- **The type of quantum state**, which can be either mixed or pure in the local component.

- **Noise settings**, which include the type and level (probability p ranging from 0 to 1) of noise. *VeriQR* provides users with the option to select three standard types of noise, namely *depolarizing*, *phase flip*, and *bit flip*. Furthermore, users are given the ability to choose a combination of these three types of noise and even customize a new noise themselves.
- **A decimal perturbation parameter** ε and **the number of experiments**. For example, for the case where ε=`1e-3` and the number of experiments is `3`, *VeriQR* will check the `1e-3`, `2e-3`,`3e-3`-robustness of the quantum classifier 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.



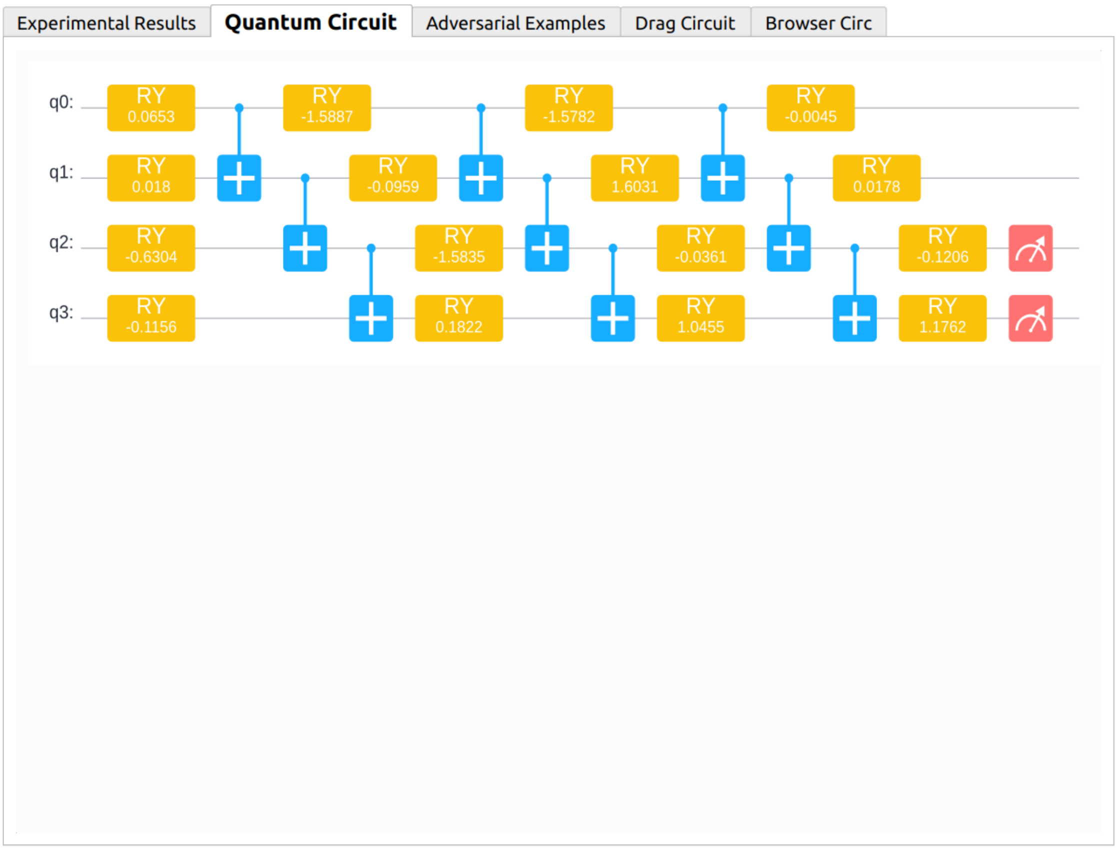
Robust Accuracy (in Percent)

	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

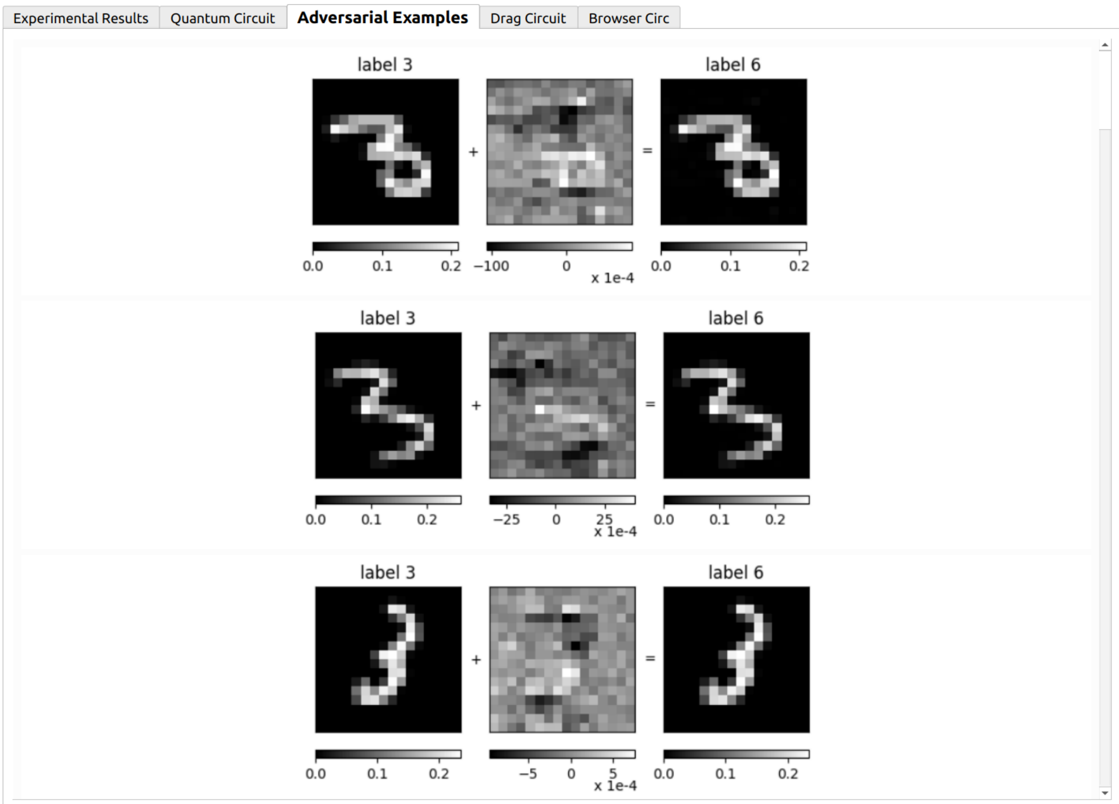
Verification Times (in Seconds)

	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

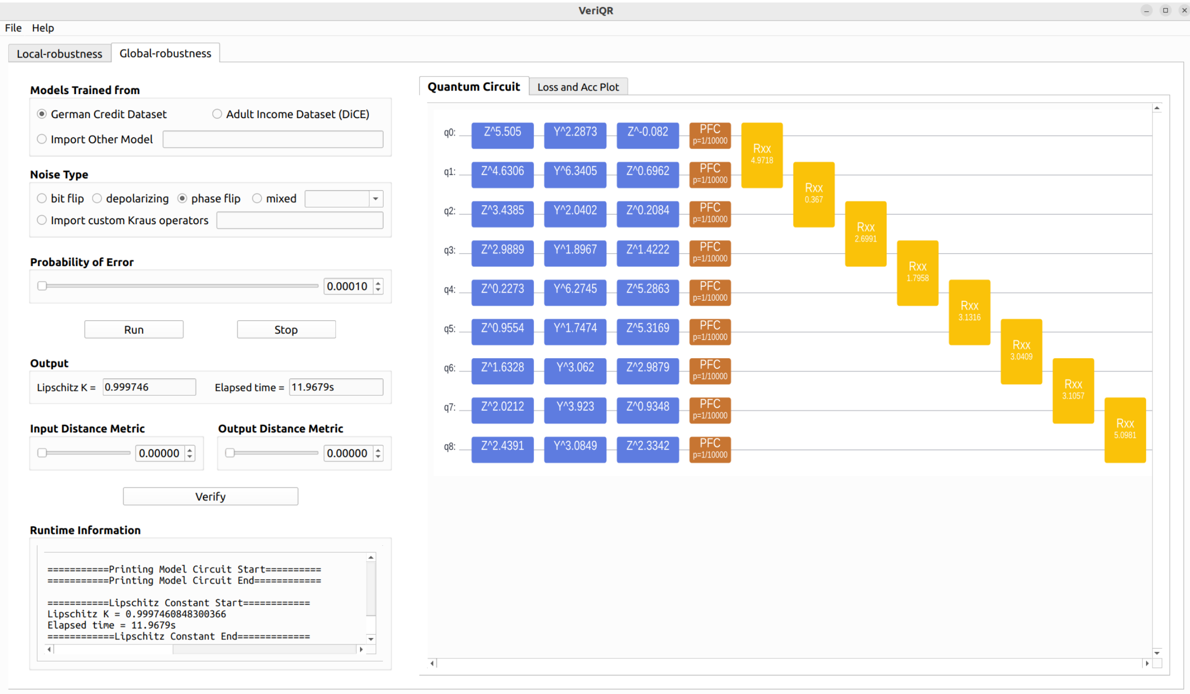
- Moreover, it depicts the quantum circuit corresponding to each quantum classifier in a diagram. (You can use the mouse wheel to zoom in or out of the picture.)



- Remarkably, *VeriQR* generates adversarial examples of the MNIST classifier and displays them in the graphical user interface. Here you can choose any combinations of handwritten digits $\{0, 1, 2, \ldots, 9\}$ to generate adversarial examples.



Global Robustness Verification



Input

- **A well-trained QML model**. *VeriQR* only accepts a QML model in the `.qasm` format in the global component. It is important to mention that the verification of *global robustness* does not require the original dataset as input. Therefore, users only need to import a QML model in a `.qasm` file as mentioned above, without the need for additional dataset. *VeriQR* inherently provides several examples in the `.qasm` format as introduced in the [experimental results] section.
- **Noise settings**, same as the local component.

The above input is used to calculate the Lipschitz constant K^*, which can be started by clicking the "**Run**" button. After the calculation is completed, *VeriQR* accepts the following parameters for verifying the *global robustness*:

- **Two decimal perturbation parameter** ε and δ.

Output

In this section, *VeriQR* also depicts the quantum circuit diagram corresponding to each QML model.