VeriQR: A Robustness Verification Tool for Quantum Machine Learning Models

Yanling Lin[1,2], Ji Guan[2(✉)], Wang Fang[2], Mingsheng Ying[3], and Zhaofeng Su[1(✉)]

[1] University of Science and Technology of China, Hefei 230026, China
zfsu@ustc.edu.cn

[2] Key Laboratory of System Software (Chinese Academy of Sciences) and State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing 100190, China
guanj@ios.ac.cn

[3] Centre for Quantum Software and Information, University of Technology Sydney, NSW 2007, Australia

Abstract. Adversarial noise attacks present a significant threat to quantum machine learning (QML) models, similar to their classical counterparts. This is especially true in the current Noisy Intermediate-Scale Quantum era, where noise is unavoidable. Therefore, it is essential to ensure the robustness of QML models before their deployment. To address this challenge, we introduce *VeriQR*, the first tool designed specifically for formally verifying and improving the robustness of QML models, to the best of our knowledge. This tool mimics real-world quantum hardware's noisy impacts by incorporating random noise to formally validate a QML model's robustness. *VeriQR* supports exact (sound and complete) algorithms for both local and global robustness verification. For enhanced efficiency, it implements an under-approximate (complete) algorithm and a tensor network-based algorithm to verify local and global robustness, respectively. As a formal verification tool, *VeriQR* can detect adversarial examples and utilize them for further analysis and to enhance the local robustness through adversarial training, as demonstrated by experiments on real-world quantum machine learning models. Moreover, it permits users to incorporate customized noise. Based on this feature, we assess *VeriQR* using various real-world examples, and experimental outcomes confirm that the addition of specific quantum noise can enhance the global robustness of QML models. These processes are made accessible through a user-friendly graphical interface provided by *VeriQR*, catering to general users without requiring a deep understanding of the counter-intuitive probabilistic nature of quantum computing.

Keywords: Robustness Verification · Quantum Machine Learning · Formal Verification · Quantum Classifiers · Quantum Noise

A. Platzer et al. (Eds.): FM 2024, LNCS 14933, pp. 403–421, 2025.
https://doi.org/10.1007/978-3-031-71162-6_21

1 Introduction

Over the last decade, machine learning (ML) has driven technological advancements in various fields. The combination of machine learning with quantum computing has given rise to a new field of research known as *quantum machine learning (QML)*. In classical ML, classification models are vulnerable in adversarial scenarios [5,8]. Specifically, the addition of intentionally crafted noises to the original data can cause classifiers to make incorrect predictions with high confidence. An illustrative example is the misclassification of a panda image as a gibbon with a confidence level exceeding 99% after adding imperceptible noise [37]. Although studies have shown the potential superiority of quantum computers over classical counterparts in certain well-known ML tasks [4], the presence of noise in quantum computation is inevitable due to the limitations of quantum hardware devices in the current Noisy Intermediate-Scale Quantum (NISQ) era [34], which may cause quantum learning systems to suffer from adversarial perturbations from environmental noises. Research on the vulnerability of QML models has garnered widespread attention [15,19,20,23,30,31,41]. In particular, formal methods have been employed to verify the robustness of QML models against noises. Various algorithms have been developed to verify both local and global robustness, which have established a formal framework for verifying the robustness of QML models, allowing for detecting non-robust quantum states (also known as quantum adversarial examples) during the verification process.

Numerous tools have been developed to verify the robustness of classical ML models and improve robustness through adversarial training. Notable examples include NNV [39], Reluplex [27], DeepG [2], PRODeep [28], VerifAI [14] and AI^2 [18]. These tools have simplified the process for users to verify the robustness of their ML models. However, understanding the counter-intuitive principles of quantum mechanics, which serve as the inherent probabilistic foundation of quantum systems, poses a distinctive challenge for the average user. Therefore, there is a requirement for automated tools in the analysis of quantum systems.

In recent years, formal methods-based tools have emerged to verify the correctness of quantum systems. For instance, the development of a specification language and an automated tool called AUTOQ enables symbolic verification of quantum circuits [10]. Similarly, CoqQ, integrated into the Coq proof assistant, provides a means to reason about quantum programs [45]. A measurement-based linear-time temporal logic (MLTL) has been proposed to formally check the quantitative properties of quantum algorithms [21]. Furthermore, model checkers like QMC [33] and QPMC [16] have been proposed for verifying quantum programs and communication protocols. However, to the best of our knowledge, there are currently no dedicated tools available for verifying the robustness of QML models and then improving robustness.

Contributions. To fill the gap mentioned above, we introduce a tool named *VeriQR*. *VeriQR* is built upon the aforementioned theoretical formal verification techniques [19,20] for automatically quick robustness verification of QML models

and the improvement strategies for enhancing robustness. The architecture of *VeriQR* is shown in Fig. 1 and its main advantages are listed in the following.

1. For *universality*, *VeriQR* supports the verification of two distinct robustness properties. These are referred to as *local robustness* for QML classification models and *global robustness* for all existing QML models.
2. For *usability*, *VeriQR* offers support for QML models that are represented in the OpenQASM 2.0 format of IBM [11]. This format is widely utilized as a programming language for describing quantum circuits and algorithms.
3. For *reality*, *VeriQR* formally verifies the robustness of a QML model by adding random noise to the model. This functionality enables simulations of the noisy effects of real-world quantum hardware on the robustness verification of various QML models.
4. For *efficiency*, in addition to basic verification methods, *VeriQR* incorporates various optimization techniques. These include approximation techniques for *local robustness* and tensor network contractions for *global robustness*, which enhance the performance of the verification process.
5. For *local robustness enhancement*, *VeriQR* can utilize the identified adversarial examples from the verification process for adversarial training, akin to traditional methods. Additionally, users have the option to introduce customized noise for *improving global robustness*, as discussed in [15,20,25]. This customized noise extends beyond standard quantum noise to include user-defined quantum noise models.

In Sect. 4, we present experimental results demonstrating the versatility and practicality of *VeriQR* in verifying and improving the robustness of different QML models in real-world scenarios. The experiments cover a range of noise types and levels, showcasing the efficacy and reliability of *VeriQR*.

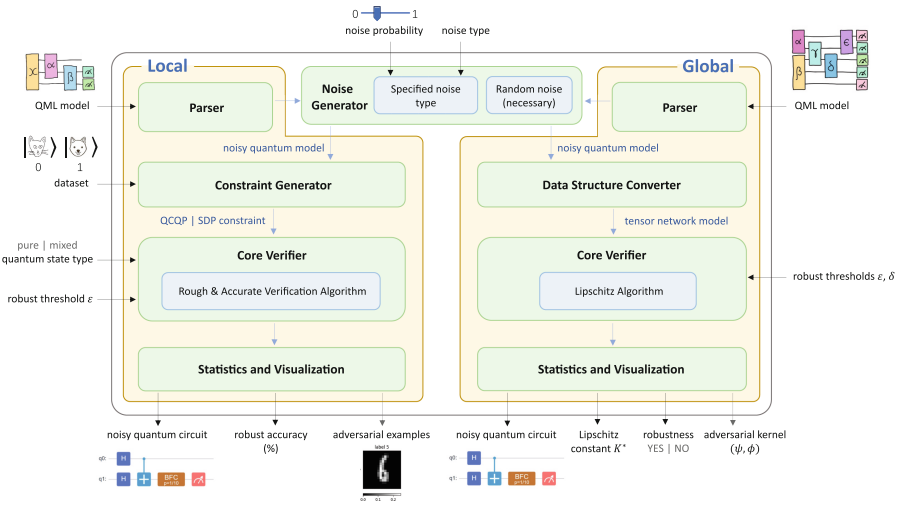

Fig. 1. An overview of the architecture of *VeriQR*.

2 Robustness for Quantum Machine Learning Models

For the convenience of the reader, we briefly introduce the concepts of quantum computing used in this paper and QML models (algorithms). We then review the local and global robustness verification problems for QML models in their most basic form, which can be handled by our tool *VeriQR*. For more details, please refer to [19,20,32].

2.1 Quantum Machine Learning Model

A QML model $\mathcal{A}$ is composed of input quantum states, a quantum circuit, and a quantum measurement.

Quantum state. The input quantum state ρ refers to the data that is processed by the quantum model. Mathematically, ρ is a positive semi-definite matrix with a size of 2^n-by-2^n, where n represents the number of quantum bits (qubits). It is important to note that the quantum state ρ can not only represent quantum information, such as the state of a physical Hamiltonian system, for physical computational tasks but also encode classical information, such as image data or financial data, for classical computational tasks.

Quantum circuit. The noisy quantum circuit $\mathcal{E}$ is used to describe the computational aspect of the QML model. A quantum circuit consists of a sequence of *quantum logic gates* and *quantum noises* (represented by yellow and brown boxes in Fig. 2, respectively).

Quantum logic gates are the building blocks of quantum circuits and can transform a quantum state into a new quantum state, like classical logic gates are for conventional digital circuits. They are described as unitary matrices relative to some orthogonal basis. Mathematically, a gate that acts on an n-qubit quantum state ρ is represented by a $2^n \times 2^n$ unitary matrix U, and its output is a evolved quantum state $\rho' = U\rho U^\dagger$, where $U^\dagger$ is the conjugate transpose of U.

Quantum noise in quantum systems can be broadly characterized as either coherent or incoherent. Coherent noise generally originates from the noisiness of the parameters in gate operations, so it is unitary evolution (represented by a unitary matrix) and easy to simulate; incoherent noise arises from the interaction between the system and the environment and thus is usually a non-unitary evolution, which transforms the state of the quantum system from a pure state ρ to a mixed state $\mathcal{E}(\rho)$ with $\mathcal{E}(\rho) = \sum_k E_k \rho E_k^\dagger$, where the matrices $\{E_k\}$ with a size of 2^n-by-2^n are called *Kraus operators*, satisfying the completeness conditions $\sum_k E_k^\dagger E_k = I$, where I is the identity operator. This transformation is also known as a *quantum channel*, it is a quantum operation characterized by a 2^n-by-2^n matrix. Mathematical representations of common 1-qubit quantum channels, including *bit flip channel (BFC)*, *phase flip channel (PFC)*, and *depolarizing channel (DC)*, are described as follows:

$$\mathcal{E}_{\mathrm{BFC}}(\rho) = (1-p)I\rho I + pX\rho X$$
$$\mathcal{E}_{\mathrm{PFC}}(\rho) = (1-p)I\rho I + pZ\rho Z$$
$$\mathcal{E}_{\mathrm{DC}}(\rho) = (1-p)I\rho I + \frac{p}{3}(X\rho X + Y\rho Y + Z\rho Z)$$

and

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}, Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}.$$

Here p represents the likelihood of the state ρ undergoing further manipulation by a quantum gate. For instance, in a bit flip channel, p signifies the chance of a bit flip operation affecting the quantum state. These three categories of quantum channels are frequently encountered noise in real-world quantum hardware. In this context, p serves as a measure of the noise level. A higher value of p corresponds to a more pronounced alteration in the initial state ρ. Therefore, the state of the quantum system after a noisy quantum circuit $\mathcal{E}$ represented by a set of matrices $\{E_k\}$ is $\mathcal{E}(\rho) = \sum_k E_k \rho E_k^\dagger$.

Quantum Measurement. At the end of each quantum circuit, a quantum measurement (represented by red boxes in Fig. 2) is performed to extract the computational outcome, which contains classical information, from $\mathcal{E}(\rho)$. This information is a probability distribution over the possible outcomes of the measurement. Mathematically, a quantum measurement is modeled by a set $\{M_c\}_{c\in\mathcal{C}}$ of positive semi-definite matrices with a size of 2^n-by-2^n. Here, $\mathcal{C}$ represents a finite set of measurement outcomes or class labels. The observation process is probabilistic: for the current state $\mathcal{E}(\rho)$, the measurement outcome $c \in \mathcal{C}$ is obtained with probability $p_c = \mathrm{tr}(M_c\mathcal{E}(\rho))$, which is the summation of diagonal entries of $M_c\mathcal{E}(\rho)$.

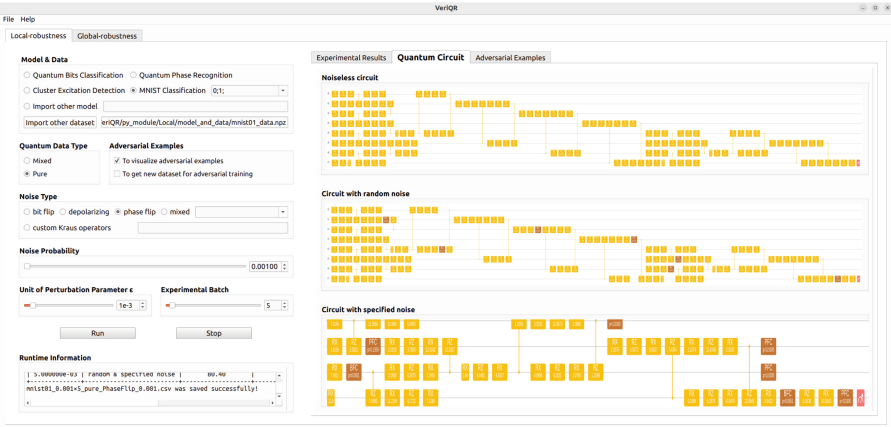

Fig. 2. GUI: the main tabs for verification task and (8-qubit) quantum circuit diagrams corresponding to the QML model to be verified. In this diagram representation, yellow boxes represent 1-qubit gates, blue ones represent controlled gates (not shown in this example), brown ones represent quantum noises, and red ones represent measurements.

In summary, a QML model, denoted as $\mathcal{A} = (\mathcal{E}, \{M_c\}_{c\in\mathcal{C}})$, can be viewed as a randomized mapping. For any input quantum state ρ, the model outputs a probability distribution $\mathcal{A}(\rho) = \{\mathrm{tr}(M_c\mathcal{E}(\rho))\}_{c\in\mathcal{C}}$.

Pure Versus Mixed Quantum States. It is essential to note that quantum states fall into two main categories: pure and mixed states. A quantum system with a known exact state $|\psi\rangle$ containing n qubits is considered to be in a pure state, which can be represented by a column vector of size 2^n in a complex vector space. In this scenario, the density matrix (operator) representing the system is $\rho = |\psi\rangle\langle\psi|$, characterized by positive semidefinite matrices with a trace of 1. On the other hand, if the state of the quantum system is not precisely known, it is classified as a mixed state, which comprises an ensemble of pure states $(p_1, |\psi_1\rangle)$, $(p_2, |\psi_2\rangle)$, ..., $(p_n, |\psi_n\rangle)$, denoted as $\rho = \sum_i p_i|\psi_i\rangle\langle\psi_i|$. This indicates that the system is in state $|\psi_j\rangle$ with a probability of p_j. A pure state $|\psi\rangle$ can be considered a special instance of the mixed state $\rho = |\psi\rangle\langle\psi|$, implying that the collection of pure quantum states is a subset of mixed quantum states. Pure states are mainly employed to safeguard against deliberate classical attacks (by humans) embedded in input quantum states, while mixed states are utilized in a broader array of situations, including defense against quantum noise. To accommodate different application contexts, our tool *VeriQR* empowers users to choose the specific type of quantum states they wish to work with.

2.2 Robustness Verification of QML Models

Similar to their classical counterparts, QML models can be delineated into two primary categories: *regression models* and *classification models*. Consequently, two distinct forms of robustness verification are requisite.

Global Robustness for Regression Models. A *regression model* $\mathcal{A}$ uses the output distribution $\mathcal{A}(\rho)$ directly to determine the predicted value for the regression variable ρ. Naturally, a regression model that is robust to adversarial noise attacks should have the ability to maintain stable predictions with a certain degree of tolerance for small changes, which could induce incorrect predictions, in the initial data. In other words, it is necessary to treat all similar input states with minor differences similarly to ensure robustness for regression models, which is called *global robustness*.

Problem 1 (Global Robustness Formal Verification). Let $\mathcal{A} = (\mathcal{E}, \{M_c\}_{c\in\mathcal{C}})$ be a QML model, and check whether $\mathcal{A}$ is (ε, δ)-globally robust, i.e., for any pair of quantum state ρ and σ with $D(\rho, \sigma) \leq \varepsilon$, we have $d(\mathcal{A}(\rho), \mathcal{A}(\sigma)) \leq \delta$. If not, provide such a pair of quantum states violating the robustness.

Here $D(\cdot, \cdot)$ and $d(\cdot, \cdot)$ represent the trace distance of two density matrices and the total variance distance of the measurement outcome probability distributions on quantum states, respectively. These distances are used to quantify the similarities in input and output states, respectively. To solve the formal verification problem (Problem 1), the Lipschitz constant K^* of $\mathcal{A}$ is introduced with the fact that $\mathcal{A}$

is (ε, δ)-globally robust, if and only if $\delta \geq \varepsilon$ [20]. Here the Lipschitz constant K^* is the smallest K such that $d(\mathcal{A}(\rho), \mathcal{A}(\sigma)) \leq KD(\rho, \sigma)$ for all quantum states ρ and σ. So the key is to compute K^* which is done by our tool *VeriQR*.

Local Robustness for Classification Models. A *classification model (classifier)* $\mathcal{A}$ utilizes the probability distribution $\mathcal{A}(\rho)$ to assign a class label $c \in \mathcal{C}$ to the input state ρ. The most commonly used approach is to assign the label with the highest corresponding probability in the output distribution $\{\mathrm{tr}(M_c\mathcal{E}(\rho))\}_{c\in\mathcal{C}}$. Naturally, a robust classifier should be able to classify all similar input states in the same class to ensure robustness, which is referred to as *local robustness*.

Problem 2 (Local Robustness Formal Verification). Let $\mathcal{A} = (\mathcal{E}, \{M_c\}_{c\in\mathcal{C}})$ be a QML model. Given an input state ρ with label $c \in \mathcal{C}$, check whether $\mathcal{A}$ is ε-locally robust, i.e., $\mathcal{A}(\sigma) = c$ for all $\sigma \in \mathcal{N}_\varepsilon(\rho)$, the ε-neighbourhood of ρ. If not, provide an adversarial example (counter-example) $\sigma \in \mathcal{N}_\varepsilon(\rho)$ with label $l \neq c$.

Here, the ε-neighbourhood of ρ is defined as $\mathcal{N}_\varepsilon(\rho) = \{\sigma : \bar{F}(\rho, \sigma) \leq \varepsilon\}$, and $\bar{F}(\rho, \sigma)$ quantifies the similarity between states ρ and σ using fidelity [32]. To evaluate the ε-robustness of a given finite set of labeled quantum states, we assess each input example individually. Subsequently, we generate a collection of concrete adversarial examples and determine the proportion of ε-robust states in the dataset. This measure, referred to as the ε-robust accuracy of the quantum classifier $\mathcal{A}$, provides insight into its *local robustness* on the dataset.

2.3 Challenges of Implementation

When it comes to implementing a verification tool for QML models, we encounter distinct challenges compared to dealing with classical ML models.

Continuous State Space. Quantum systems that operate within a linear space of finite dimensions possess a continuous state space. This implies that QML models must account for an infinite number of quantum states when conducting global robustness verification. In contrast, classical models mainly work with discrete input datasets that have a finite number of data. This distinction renders classical robustness verification techniques [1] unsuitable for quantum systems, such as the reachability method [38] and abstract interpretation [18]. Consequently, we develop *VeriQR* as an independent tool that does not rely on any existing classical robustness tools. Instead, we implement the algorithms proposed in [19,20] to verify the robustness of QML models.

State Explosion. The size of QML models, which is given by the dimension 2^n, grows exponentially as the number of qubits n increases. This poses challenges in terms of memory usage and runtime when performing robustness verification on large-scale systems. To address this, we employ tensor networks as an efficient data structure for storing quantum circuits, effectively optimizing memory usage. Furthermore, we utilize Google's tensor network calculator [36] with heuristic methods as a subroutine to enhance the efficiency of verifying global robustness (Problem 1). Furthermore, we have implemented the approximate

verification algorithm [19] for local robustness verification (Problem 2). These optimization techniques allow *VeriQR* to handle robustness verification of noisy QML models with up to 20 qubits on a small service for general users (refer to Sect. 4 for experimental results). Without these optimizations, *VeriQR* is only able to handle models with up to 8 qubits.

QML Benchmarks. Currently, there are only a few benchmarks available for quantum circuits (e.g., [9]), and there is a lack of benchmarks specifically designed for QML models. To broaden the range of applicable scenarios, we have incorporated the use of OpenQASM 2.0 files as inputs. Moreover, to further enhance this capability, we have developed built-in scripts for translating QML models on several platforms (such as Huawei's MindSpore Quantum [43] and Google's Cirq [13]) into the OpenQASM 2.0 format. This enables the establishment of a unified verification benchmark framework for QML models deployed on various popular quantum platforms, including IBM's Qiskit [26], Google's TensorFlow Quantum [7], and others. In addition, we have visualized the framework by providing a graphical user interface (GUI) that converts inputted OpenQASM 2.0 code, used for describing quantum circuits, into visual representations (see the right side of Fig. 2).

3 Overview and Features of *VeriQR*

VeriQR is a graphical user interface (GUI) tool developed using C++. The decision to use C++ was influenced by the widespread use of Qt [6] in GUI programming. As shown in Fig. 1, *VeriQR* consists of two main parts: *Local robustness verification* and *Global robustness verification*.

Inputs. To utilize *VeriQR*, the user is required to import a relevant example, specifically a QML model 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)* is utilized to package a quantum circuit, quantum measurement, and training dataset. This format is particularly beneficial for individuals who are not experts in quantum computing but have proficiency in classical formal methods and machine learning. By incorporating NumPy, *VeriQR* becomes more accessible to average users without requiring extensive learning. Moreover, *VeriQR* provides four popular testing examples (see the upper left corner of Fig. 2) of quantum classifiers in .npz, catering to beginners.
2. *An OpenQASM 2.0 file (.qasm format)* expresses the quantum circuit corresponding to the QML model to be checked. OpenQASM 2.0 is an IBM-introduced format widely adopted in the quantum computing community for constructing quantum circuits [11]. *QML models trained on different quantum platforms can be converted into this format. This allows for unified and reliable verification of robustness, addressing the challenge of "QML Benchmarks" discussed in Sect. 2.3.*

It is important to mention that the verification of *global robustness* does not require the use of the original dataset as input. Therefore, users only need to import a QML model in a .qasm file for the circuit and the measurement, without the need for additional training data. Once this step is completed, users can proceed to configure parameters for the specific case of interest. These parameters consist of the following: (i) the types and levels (probabilities ranging from 0 to 1) of noise: *VeriQR* inherently provides users with the option to select three standard types of noise, namely *depolarizing*, *phase flip*, *bit flip* [32]. Furthermore, users can also customize a new noise themselves, or even choose a combination of all types of noise; (ii) the type of quantum state, which can be either mixed or pure in the local component and is set as mixed by default in the global component. The choice for the global component is predetermined as global robustness verification for mixed states can be reduced to that for pure states; and (iii) perturbation parameters, specifically two thresholds for robustness (ε, δ in Problem 1) for the verification of *global robustness* and a threshold for robustness (ε in Problem 2) for the verification of *local robustness*.

3.1 Verifying Robustness

Verification of Local Robustness. This part is comprised of five modules:

1) **Parser**: This module handles a quantum classifiers file to obtain the corresponding quantum circuit object.
2) **Noise Generator**: The input for this module is a quantum circuit object. *First*, it generates a noisy quantum circuit by adding a random noise to each qubit at random points in the circuit with a randomly determined noise probability. The purpose of this is to simulate the effect of noise to verify the robustness of the QML model on real-world quantum hardware. *In addition*, users can also use *VeriQR* to actively add noises of specific types, including commonly used standard quantum noise models and user-defined quantum noise models (using Kraus representation) with specific noise probabilities, to the noisy model. Here, user-specified noises are added at the end of the circuit, which is a common assumption. This functionality enables robustness improvements, illustrated by our experimental results in Sect. 4.
3) **Constraint Generator**: This module generates constraints based on the (noisy) quantum model and the input dataset, which are then submitted to the core verifier.
4) **Core Verifier**: This module receives the constraints, a perturbation parameter ε, and the quantum state type as inputs. Based on the state type, it chooses the appropriate constraint solver: a Semidefinite Programming solver for mixed states, and a Quadratically Constrained Quadratic Program solver for pure states [19]. It then utilizes both the under-approximation and exact algorithms to initiate the verification analysis procedure for ε-robustness. *These algorithms and solvers are specifically designed to tackle the challenge of "Continuous State Space" discussed in Sect. 2.3 when verifying the robustness of QML models. In particular, the under-approximation algorithm is implemented to address the "State Explosion" issue.*

5) **Statistics and Visualization**: This module is responsible for visualizing and displaying the results in the GUI of *VeriQR*. The computed robust accuracy of the quantum classifier, which indicates the validity of the robustness property, is outputted by *VeriQR*. Additionally, the detected adversarial examples (quantum states) are stored in a NumPy data file for further analysis and to improve robustness through adversarial training. The GUI also presents the original and noisy quantum circuit diagrams for the classifier (see Fig. 2), providing users with an intuitive way to analyze the model construction. Moreover, for the MNIST handwritten digit classification task, *VeriQR* displays pictures of the detected adversarial examples based on the digits specified by the user in the GUI. These adversarial examples are obtained by adding noisy perturbations to a set of legitimate input examples, as illustrated in Fig. 3.

Verification of Global Robustness. This part also includes five modules similar to validating *local robustness*. However, instead of being verified directly by the verifier, *the noisy model generated by the noise generator is first passed to the data structure converter and is transformed into the corresponding* tensor networks *model to improve efficiency and overcome the challenge of "State Explosion" discussed in Sect. 2.3.* The core verifier then takes the tensor network model as input and calculates the required constant K^* for model validation, following the procedure outlined in Algorithm 1 of [20]. In addition to this, the core verifier receives perturbation parameters ε and δ in Problem 1 for validation and finally determines whether the global robustness property holds by checking if $\delta \geq K^*\varepsilon$. If not, it will provide an adversarial kernel (ψ, ϕ), which is capable of generating infinitely many pairs of quantum states that violate the global robustness of the QML model.

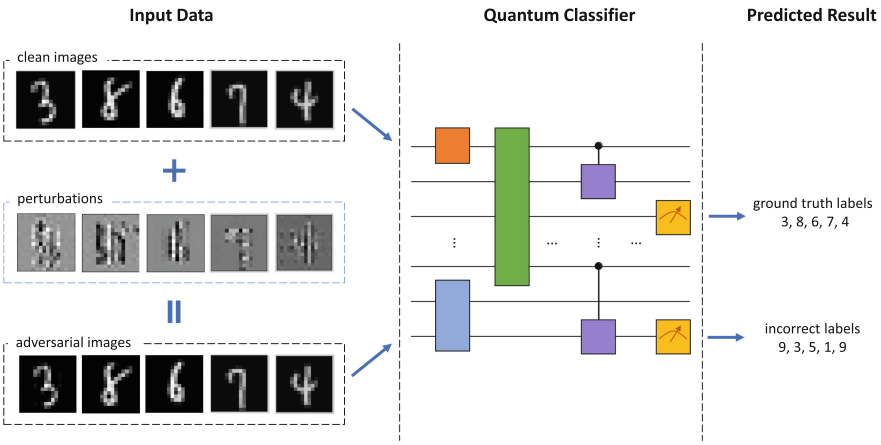

Fig. 3. The adversarial examples and the corresponding adversarial perturbations found by *VeriQR* in MINIST handwritten digit classification.

3.2 Improving Robustness

VeriQR offers adversarial training and adding specific noise to enhance the local and global robustness of QML models, respectively. The effectiveness of these strategies is validated through experiments on various QML models in Sect. 4.

Adversarial training. *VeriQR* empowers users with adversarial training capabilities, an extension of traditional machine learning. When the ε-local robustness of ρ with label l is compromised, our robustness verification algorithms embedded in *VeriQR* automatically generate an adversarial example σ. By incorporating (σ, l) into the training dataset, users can then retrain the QML model to enhance its local robustness against the adversarial examples.

Specific noise. Previous research [15,20,25] suggests that introducing specific quantum noise at strategic points in the circuits of QML models can improve global robustness. The *VeriQR* tool empowers users to apply standard or personalized noise at various locations in quantum circuits for robustness enhancement.

4 Evaluation

In this section, we evaluate the effectiveness of *VeriQR* in verifying and enhancing both the local and global robustness of various QML models. These models utilize popular parameterized quantum circuits like Quantum Neural Networks (QNN), Quantum Convolutional Neural Networks (QCNN), Quantum Approximate Optimization Algorithms (QAOA), Variational Quantum Eigensolver Algorithms (VQE) and Quantum Supremacy Algorithms. All these networks have previously demonstrated successful implementation on practical quantum hardware [44]. All the experiments are performed on a workstation with a Intel(R) Xeon(R) Gold 6254 CPU @ 3.10GHz $\times$ 72 Cores Processor and 314 GB RAM.

4.1 Local Robustness

We conducted several experiments to test the *local robustness* of various quantum classifiers with different numbers of qubits. These classifiers were trained on labeled datasets that were encoded using different quantum encoders in platforms such as Mindspore Quantum [43] and Tensorflow Quantum [7]. The classifiers examined in our study include the *qubit* classifier, which determines the qubit's position in the X-Z plane of a Bloch sphere [7]; the *iris* classifier, which categorizes irises from various subgenera [17]; the *mnist* classifier, which identifies handwritten digits, specifically 1 & 3 [12]; the *fashion* classifier, which classifies images of T-shirts and ankle boots [42]; and the *tfi* classifier, which recognizes wavefunctions at different phases in a quantum many-body system [7].

Table 1. Experimental results of the *local robustness* verification of different QML models.

Model	#Qubits	ε	Circuit	Noise Setting ($noise\_p$)	Rough Verif		Accurate Verif	
					RA (%)	VT (s)	RA (%)	VT (s)
qubit	1	0.001	c_0	noiseless	88.12	0.0038	90	2.4226
			c_1	random	88.12	0.0039	90	2.4623
			c_2	depolarizing_0.001	88.00	0.0038	90	2.4873
			c_2	depolarizing_0.005	87.62	0.0053	90	2.7140
iris	4	0.005	c_0	noiseless	98.75	0.0013	100	0.4924
			c_1	random	97.50	0.0009	100	0.8876
			c_2	mixed_0.01	97.50	0.0019	100	0.8808
			c_2	mixed_0.05	96.25	0.0021	100	3.1675
tfi	4	0.005	c_0	noiseless	86.41	0.0039	100	6.5220
			c_1	random	85.94	0.0038	100	6.6438
			c_2	mixed_0.01	85.78	0.0061	100	6.7117
			c_2	mixed_0.05	85.16	0.0063	100	7.0374
tfi	8	0.005	c_0	noiseless	98.44	0.0372	100	2.3004
			c_1	random	96.56	0.1061	100	3.9492
			c_2	bit-flip_0.01	96.56	37.0965	100	42.1246
			c_2	bit-flip_0.05	95.94	32.7195	100	38.8139
fashion	8	0.001	c_0	noiseless	90.60	0.0420	97.40	25.3777
			c_1	random	90.30	0.0934	97.30	27.4964
			c_2	bit-flip_0.01	89.90	15.6579	97.20	42.1063
			c_2	bit-flip_0.05	87.60	14.0342	96.70	48.5805
mnist (1&3)	8	0.003	c_0	noiseless	93.80	0.0543	96.00	18.5063
			c_1	random	92.60	0.0785	95.70	23.2905
			c_2	phase-flip_0.001	92.60	12.9728	95.70	36.2348
			c_2	phase-flip_0.01	92.60	11.6704	95.70	33.7894

Experiment Setting for Verification: To investigate the impact of random and specific noise on local robustness verification, we conducted experiments on four different circuits for each model as outlined in Table 1: the noiseless ideal QML model with quantum circuit c_0; circuit c_1 created by introducing random noise at various random points in circuit c_0 to simulate noise effects on NISQ devices; and circuit c_2 modified by adding specific noise with a noise level p (referred to as "noisename_p" below c_2) of four types: *depolarizing*, *phase flip*, *bit flip*, and *mixed* (a combination of the three) noise, introduced in Sect. 2.1, applied to each qubit after the random noise manipulation on circuit c_1.

Experiment Setting for Approximate Versus Exact Verification: In each robustness verification scenario, we employed two verification techniques:

a coarse method labeled "Rough Verif" and a precise method labeled "Accurate Verif". We must emphasize here the difference between accurate and rough verification methods for local robustness verification. The *rough verification* method detects non-robust states only by applying the robust bound condition from the work in [19]. However, quantum states that do not satisfy this condition may also be robust, leading to an underestimation of the robust accuracy. Therefore, the *accurate verification* method first filters out possible non-robust states using the condition, and then uses a Semidefinite Programming solver to obtain the optimal robust bound for these states, thus verifying the local robustness of each state precisely.

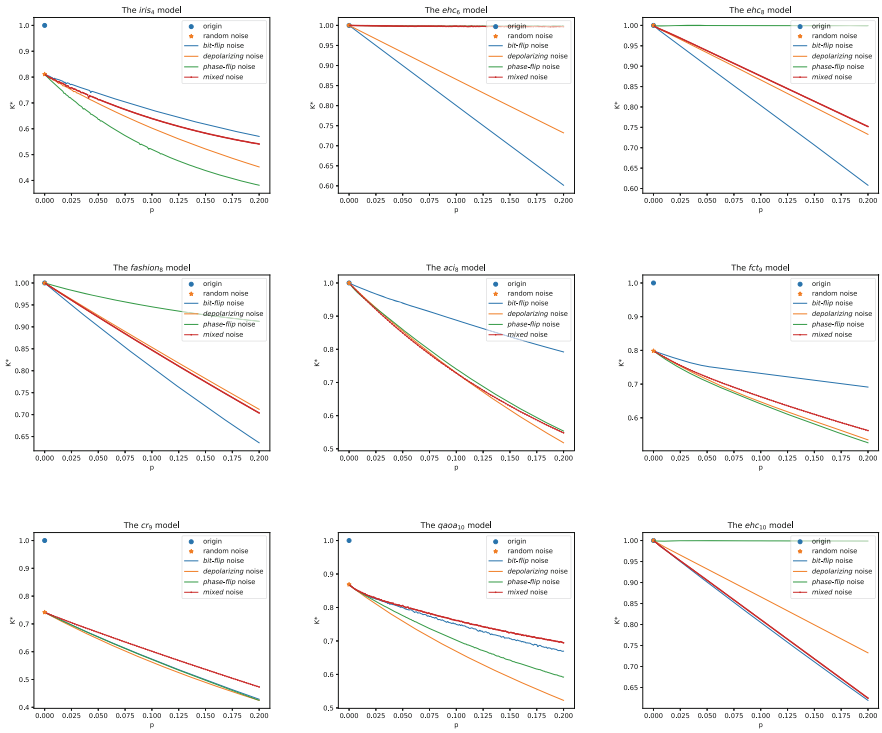

Fig. 4. Experimental results of the trade-off between the Lipschitz constant K^* (measuring *global robustness*) and noise level p in different QML models.

Table 1 presents a summary of the outcomes obtained from our experiments on *local robustness* verification. In this table, the robust accuracy of the classifiers is represented as "RA", while the verification time (in seconds) is indicated as "VT". The experimental results reveal two key aspects:

1. By examining the RA values in rows c_0, c_1, and c_2 for each QML experiment in Table 1, it becomes evident that both random noise and specific noise cannot enhance robustness, particularly in the *fashion* and *mnist* experiments.

2. When comparing the RA (VT) values between the "Rough Verif" and "Accurate Verif" columns, it is observed that the under-approximation of robust accuracy scales well in almost all cases with faster verification time, supporting the conclusions drawn in [19].

Remark 1. Our tool, *VeriQR*, serves as a formal instrument capable of identifying all non-robust quantum states (adversarial examples) during the verification process of all quantum classifiers. Analogous to classical methodologies, adversarial training can be utilized to fortify non-robust states within the retrained models. In our study, we have incorporated the adversarial training technique as outlined in Sect. 2.2 for all QML models listed in Table 1. Despite being a conventional classical practice, we have documented the outcomes on our code repository.

4.2 Global Robustness

For *global robustness*, we also incorporate various types of noise and their corresponding noise levels into the quantum models to be tested. We conducted multiple experiments on different QML models using the *VeriQR* tool. These experiments encompass a wide range of applications, including the *aci* model for adult census income prediction [3], the *fct* model for detecting fraudulent credit card transactions [40], the *cr* model for classifying individuals as good or bad credit risks based on a set of attributes [24], the *ehc* model for calculating the binding energy of hydrogen chains [35], the *qaoa* model for solving hardware grid problems [22].

Table 2. Experimental comparison of tensor network-based verification with a baseline implementation lacking tensors for assessing *global robustness*.

Model	#Qubits	Noise	p	(ε, δ)	Baseline		TN		Robust
					K^*	time (s)	K^*	time (s)	
ehc	8	bit flip	0.0001	(0.0003, 0.0075)	0.99980	0.26	0.99976	26.17	YES
		depolarizing	0.05	(0.001, 0.0075)	0.93333	0.26	0.93304	27.87	YES
		phase flip	0.025	(0.075, 0.0003)	1	0.26	0.99968	28.46	NO
		mixed	0.0005	(0.005, 0.005)	0.99938	0.24	0.99905	25.75	YES
aci	8	bit flip	0.0001	(0.003, 0.0001)	0.99985	0.18	0.99985	6.44	NO
		depolarizing	0.025	(0.03, 0.0005)	0.92640	0.25	0.92440	7.70	NO
		phase flip	0.05	(0.05, 0.001)	0.88450	0.19	0.85990	8.58	NO
		mixed	0.005	(0.005, 0.005)	0.98384	0.22	0.98326	6.06	YES
fct	9	bit flip	0.05	(0.075, 0.003)	0.99024	0.98	0.97683	13.89	NO
		depolarizing	0.05	(0.0003, 0.0001)	0.92638	0.76	0.92486	40.73	NO
		phase flip	0.01	(0.01, 0.0075)	0.98730	0.87	0.98290	10.45	NO
		mixed	0.05	(0.075, 0.0075)	0.94531	0.89	0.92949	9.06	NO

continued

Table 2. continued

Model	#Qubits	Noise	p	(ε, δ)	Baseline		TN		Robust
					K^*	time (s)	K^*	time (s)	
cr	9	bit flip	0.025	(0.01, 0.0005)	0.93964	0.65	0.93819	14.44	NO
		depolarizing	0.005	(0.075, 0.005)	0.98637	1.21	0.98515	6.49	NO
		phase flip	0.025	(0.0003, 0.0001)	0.94753	0.97	0.93772	9.63	NO
		mixed	0.025	(0.0001, 0.0001)	0.95579	0.93	0.94980	12.15	YES
qaoa	10	bit flip	0.005	(0.05, 0.0005)	0.99843	5.23	0.98507	16.98	NO
		depolarizing	0.0001	(0.01, 0.003)	0.99983	6.15	0.99965	16.10	NO
		phase flip	0.005	(0.075, 0.0075)	0.99224	5.14	0.98516	17.95	NO
		mixed	0.001	(0.03, 0.0075)	0.99923	4.98	0.99657	16.16	NO
ehc	10	bit flip	0.075	(0.05, 0.0003)	0.85409	3.37	0.85262	82.25	NO
		depolarizing	0.0005	(0.03, 0.001)	0.99933	5.69	0.99924	40.33	NO
		phase flip	0.01	(0.0003, 0.0075)	1	4.36	0.99857	66.67	YES
		mixed	0.0001	(0.005, 0.001)	0.99981	5.26	0.99977	38.13	NO
ehc	12	bit flip	0.005	(0.0005, 0.0003)	0.99001	169.42	0.98965	76.77	NO
		depolarizing	0.0005	(0.0001, 0.005)	0.99933	253.11	0.99926	189.35	YES
		phase flip	0.075	(0.001, 0.0075)	1	163.61	0.99880	675.50	YES
		mixed	0.001	(0.01, 0.0001)	0.99997	195.48	0.99984	64.50	NO
inst	16	bit flip	0.005	(0.0005, 0.0003)	-	TO	0.98009	1052.73	NO
		depolarizing	0.0005	(0.0003, 0.005)	-	TO	0.99833	33.99	YES
		phase flip	0.05	(0.001, 0.0075)	-	TO	0.95131	381.15	YES
		mixed	0.001	(0.005, 0.0003)	-	TO	0.99899	123.25	NO
qaoa	20	bit flip	0.05	(0.005, 0.001)	-	TO	0.91194	2402.32	NO
		depolarizing	0.075	(0.005, 0.003)	-	TO	0.83488	433.05	NO
		phase flip	0.0005	(0.0001, 0.0001)	-	TO	0.99868	70.00	YES
		mixed	0.05	(0.075, 0.0003)	-	TO	0.89682	4635.55	NO

Noise improving global robustness. Fig. 4 depicts the scaling of the Lipschitz constant K^* (which quantifies global robustness as discussed in Sect. 2.2) across various models at different noise levels p for four distinct noise types. The figure also showcases the experimental outcomes of the original model alongside a model derived from the original version with random noise. These results indicate that *the global robustness of all models improves due to quantum noise*, as evidenced by the reduced K^* value in the models. This outcome validates earlier theoretical findings suggesting that specific quantum noise can boost global robustness [15,20,25]. The presence of "_n" in each model name in the figure signifies the model's utilization of n qubits.

High efficiency of tensor network. Importantly, *VeriQR* transformed quantum models into tensor network models and applied a tensor network-driven algorithm (referred to as "TN" in Table 2) for global robustness assessment. Table 2 provides an experimental comparison with a baseline implementation

(labeled as "Baseline") that does not incorporate tensors in global robustness evaluation. In this evaluation, a timeout threshold ("TO" entries) of 7,200 s was imposed. The results demonstrate that *the tensor network approach significantly enhances verification speed for a large number of qubits (more than 12)*, thereby improving the scalability compared to the precise *local robustness* verification outlined in Table 1.

Remark 2. To further verify the robustness of *VeriQR* both locally and globally, we have conducted additional experiments. These experiments involved testing the QML models presented in Tables 1 and 2 but with varying numbers of qubits and different types and levels of noise. Furthermore, the experimental QML models encompass 45 MNIST classifiers that have been designed to classify all possible combinations of handwritten digits $\{0, 1, 2, \ldots, 9\}$. All of these experiment results, along with the corresponding artifact for this paper, can be accessed in our code repository.

5 Conclusion

This paper presented *VeriQR*, a graphical user interface (GUI) tool developed to verify the robustness of QML models in the current NISQ era, where noise is unavoidable. *VeriQR* offers exact, under-approximate, and tensor network-based algorithms for local and global robustness verification of real-world QML models in the presence of quantum noise. Throughout the verification process, *VeriQR* can identify quantum adversarial examples (states) and utilize them for adversarial training to improve the local robustness as the same as classical machine learning. Additionally, *VeriQR* applies specific quantum noise to enhance the global robustness. Furthermore, *VeriQR* is capable of accommodating any quantum model in the OpenQASM 2.0 format and can convert QML models into this format to establish a unified benchmark framework for robustness verification.

Acknowledgments. We would like to thank Runhong He for his valuable discussion. This work was partly supported by the Youth Innovation Promotion Association, Chinese Academy of Sciences (Grant No. 2023116), the Australian Research Council (Grant No. DP220102059), National Natural Science Foundation of China (Grants No. 62002333) and Innovation Program for Quantum Science and Technology (Grants No. 2021ZD0302901). This work was done when Yanling Lin was a remote research intern supervised by A/Prof. Ji Guan at the Institute of Software, Chinese Academy of Sciences.

Data Availability Statement. The raw (classical) data underlying this article are available in the article, and the corresponding quantum version data can be found in the online supplement material - the github code repository https://github.com/VeriQ/VeriQR or the artifacts at [29].

References

1. Albarghouthi, A., et al.: Introduction to neural network verification. Found. Trends® Programm. Lang. **7**(1–2), 1–157 (2021)
2. Balunovic, M., Baader, M., Singh, G., Gehr, T., Vechev, M.: Certifying geometric robustness of neural networks. In: Advances in Neural Information Processing Systems, vol. 32 (2019)
3. Becker, B., Kohavi, R.: Adult. UCI Machine Learning Repository (1996)
4. Biamonte, J., Wittek, P., Pancotti, N., Rebentrost, P., Wiebe, N., Lloyd, S.: Quantum machine learning. Nature **549**(7671), 195–202 (2017)
5. Biggio, B., Roli, F.: Wild Patterns: ten years after the rise of adversarial machine learning. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, pp. 2154–2156 (2018)
6. Blanchette, J., Summerfield, M.: C++ GUI programming with Qt 4. Prentice Hall Professional (2006)
7. Broughton, M., et al.: TensorFlow Quantum: a software framework for quantum machine learning. arXiv preprint arXiv:2003.02989 (2020)
8. Chakraborty, A., Alam, M., Dey, V., Chattopadhyay, A., Mukhopadhyay, D.: A survey on adversarial attacks and defences. CAAI Trans. Intell. Technol. **6**(1), 25–45 (2021)
9. Chen, K., et al.: VeriQBench: a benchmark for multiple types of quantum circuits. arXiv preprint arXiv:2206.10880 (2022)
10. Chen, Y.-F., Chung, K.-M., Lengál, O., Lin, J.-A., Tsai, W.-L.: AutoQ: An Automata-Based Quantum Circuit Verifier. In: Enea, C., Lal, A. (eds.) Computer Aided Verification: 35th International Conference, CAV 2023, Paris, France, July 17–22, 2023, Proceedings, Part III, pp. 139–153. Springer Nature Switzerland, Cham (2023). https://doi.org/10.1007/978-3-031-37709-9_7
11. Cross, A.W., Bishop, L.S., Smolin, J.A., Gambetta, J.M.: Open quantum assembly language. arXiv preprint arXiv:1707.03429 (2017)
12. Deng, L.: The mnist database of handwritten digit images for machine learning research. IEEE Signal Process. Mag. **29**(6), 141–142 (2012)
13. Developers, C.: Cirq. https://quantumai.google/cirq
14. Dreossi, T., et al.: VERIFAI: a toolkit for the formal design and analysis of artificial intelligence-based systems. In: International Conference on Computer Aided Verification, pp. 432–442. Springer (2019). https://doi.org/10.1007/978-3-030-25540-4_25
15. Du, Y., Hsieh, M.H., Liu, T., Tao, D., Liu, N.: Quantum noise protects quantum classifiers against adversaries. Phys. Rev. Res. **3**(2), 023153 (2021)
16. Feng, Y., Hahn, E.M., Turrini, A., Zhang, L.: QPMC: a model checker for quantum programs and protocols. In: Bjørner, N., de Boer, F. (eds.) FM 2015. LNCS, vol. 9109, pp. 265–272. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-19249-9_17
17. Fisher, R.A.: Iris. UCI Machine Learning Repository (1988)
18. Gehr, T., Mirman, M., Drachsler-Cohen, D., Tsankov, P., Chaudhuri, S., Vechev, M.: AI2: safety and robustness certification of neural networks with abstract interpretation. In: 2018 IEEE Symposium on Security and Privacy (SP), pp. 3–18. IEEE (2018)
19. Guan, J., Fang, W., Ying, M.: Robustness verification of quantum classifiers. In: Silva, A., Leino, K.R.M. (eds.) Computer Aided Verification: 33rd International Conference, CAV 2021, Virtual Event, July 20–23, 2021, Proceedings, Part I,

pp. 151–174. Springer International Publishing, Cham (2021). https://doi.org/10.1007/978-3-030-81685-8_7
20. Guan, J., Fang, W., Ying, M.: Verifying fairness in quantum machine learning. In: Shoham, S., Vizel, Y. (eds.) Computer Aided Verification: 34th International Conference, CAV 2022, Haifa, Israel, August 7–10, 2022, Proceedings, Part II, pp. 408–429. Springer International Publishing, Cham (2022). https://doi.org/10.1007/978-3-031-13188-2_20
21. Guan, J., Feng, Y., Turrini, A., Ying, M.: Measurement-based verification of quantum Markov chains. In: Gurfinkel, A., Ganesh, V. (eds.) Computer Aided Verification: 36th International Conference, CAV 2024, Montreal, QC, Canada, July 24–27, 2024, Proceedings, Part III, pp. 533–554. Springer Nature Switzerland, Cham (2024). https://doi.org/10.1007/978-3-031-65633-0_24
22. Harrigan, M.P., et al.: Quantum approximate optimization of non-planar graph problems on a planar superconducting processor. Nat. Phys. **17**(3), 332–336 (2021). https://doi.org/10.1038/s41567-020-01105-y
23. Helstrom, C.W.: Detection theory and quantum mechanics. Inf. Control **10**(3), 254–291 (1967)
24. Hofmann, H.: Statlog (German Credit Data). UCI Machine Learning Repository (1994)
25. Huang, J.C., et al.: Certified robustness of quantum classifiers against adversarial examples through quantum noise. In: ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 1–5. IEEE (2023)
26. IBM: Learn quantum computation using Qiskit. https://qiskit.org/textbook/preface.html (Accessed 2021)
27. Katz, G., Barrett, C., Dill, D.L., Julian, K., Kochenderfer, M.J.: Reluplex: an efficient SMT solver for verifying deep neural networks. In: Majumdar, R., Kunčak, V. (eds.) CAV 2017. LNCS, vol. 10426, pp. 97–117. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63387-9_5
28. Li, R., et al.: PRODeep: a platform for robustness verification of deep neural networks. In: Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, pp. 1630–1634 (2020)
29. Lin, Y., Guan, J., Fang, W., Ying, M., Su, Z.: Artifact for veriQR (2024). https://doi.org/10.5281/zenodo.12526235
30. Liu, N., Wittek, P.: Vulnerability of quantum classification to adversarial perturbations. Phys. Rev. A **101**(6), 062331 (2020)
31. Lu, S., Duan, L.M., Deng, D.L.: Quantum adversarial machine learning. Phys. Rev. Res. **2**(3), 033212 (2020)
32. Nielsen, M.A., Chuang, I.L.: Quantum computation and quantum information. Phys. Today **54**(2), 60 (2001)
33. Gay, S.J., Nagarajan, R., Papanikolaou, N.: QMC: a model checker for quantum systems. In: Gupta, A., Malik, S. (eds.) CAV 2008. LNCS, vol. 5123, pp. 543–547. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-70545-1_51
34. Preskill, J.: Quantum computing in the NISQ era and beyond. Quantum **2**, 79 (2018)
35. Quantum, G.A., et al.: Hartree-Fock on a superconducting qubit quantum computer. Science **369**(6507), 1084–1089 (2020)
36. Roberts, C., et al.: TensorNetwork: a library for physics and machine learning (2019). https://tensornetwork.readthedocs.io/en/latest/index.html

37. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572 (2014)
38. Tran, H.-D., et al.: Robustness verification of semantic segmentation neural networks using relaxed reachability. In: Silva, A., Leino, K.R.M. (eds.) Computer Aided Verification: 33rd International Conference, CAV 2021, Virtual Event, July 20–23, 2021, Proceedings, Part I, pp. 263–286. Springer International Publishing, Cham (2021). https://doi.org/10.1007/978-3-030-81685-8_12
39. Tran, H.-D., et al.: NNV: the neural network verification tool for deep neural networks and learning-enabled cyber-physical systems. In: Lahiri, S.K., Wang, C. (eds.) Computer Aided Verification: 32nd International Conference, CAV 2020, Los Angeles, CA, USA, July 21–24, 2020, Proceedings, Part I, pp. 3–17. Springer International Publishing, Cham (2020). https://doi.org/10.1007/978-3-030-53288-8_1
40. ULB, M.L.G.: Credit card fraud detection. https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud
41. Weber, M., Liu, N., Li, B., Zhang, C., Zhao, Z.: Optimal provable robustness of quantum classification via quantum hypothesis testing. NPJ Quant. Inf. **7**(1), 76 (2021)
42. Xiao, H., Rasul, K., Vollgraf, R.: Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. arXiv preprint arXiv:1708.07747 (2017)
43. Xu, X., et al.: MindSpore Quantum: a user-friendly, high-performance, and AI-compatible quantum computing framework. arXiv preprint arXiv:2406.17248 (2024)
44. Zeguendry, A., Jarir, Z., Quafafou, M.: Quantum machine learning: a review and case studies. Entropy **25**(2), 287 (2023)
45. Zhou, L., Barthe, G., Strub, P.Y., Liu, J., Ying, M.: CoqQ: foundational verification of quantum programs. In: Proceedings of the ACM on Programming Languages, vol. 7(POPL), pp. 833–865 (2023)