ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) | 978-1-7281-6327-7/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICASSP49357.2023.10096293

QTROJAN: A CIRCUIT BACKDOOR AGAINST QUANTUM NEURAL NETWORKS

Cheng Chu *Lei Jiang* *Martin Swany* *Fan Chen*

Department of Intelligent Systems Engineering, Indiana University, Bloomington, IN, USA

ABSTRACT

We propose a circuit-level backdoor attack, *QTrojan*, against Quantum Neural Networks (QNNs) in this paper. QTrojan is implemented by a few quantum gates inserted into the variational quantum circuit of the victim QNN. QTrojan is much stealthier than a prior Data-Poisoning-based Backdoor Attack (DPBA) since it does not embed any trigger in the inputs of the victim QNN or require access to original training datasets. Compared to a DPBA, QTrojan improves the clean data accuracy by 21% and the attack success rate by 19.9%.

***Index Terms*—** Quantum Neural Network, Variational Quantum Circuit, Quantum Backdoor, Backdoor Attack

1. INTRODUCTION

Quantum Neural Networks (QNNs) shine in solving a wide variety of problems including object recognition [1, 2], natural language processing [3, 4], and financial analysis [5]. The success of QNNs motivates adversaries to transplant malicious attacks from classical neural networks to QNNs. *Backdoor attack* is one of the most dangerous malwares abusing classical neural networks [6, 7]. In a backdoor attack, a backdoor is injected into the network model, such that the model behaves normally when the backdoor is disabled, yet induces a predefined behavior when the backdoor is activated.

Although conventional Data-Poisoning-based Backdoor Attacks (DPBAs) [6, 7] are designed for classical neural networks, it is difficult to perform a DPBA against QNNs. First, a typical DPBA [6] embeds a nontrivial-size trigger (e.g., $3\% \sim 4\%$ of the input size) into the inputs of a victim classical neural network. However, the input dimension of state-of-the-art QNNs [2, 3, 4, 5, 8] is small (e.g., 4$\sim$16 qubits). Embedding even a 1-qubit trigger into the inputs of a victim QNN makes DPBAs less stealthy. Second, a DPBA has to access the original training dataset, attach a trigger to some data samples in the dataset, and train the victim QNN to learn a predefined behavior. But the original training dataset and a long training process may not be available in real-world attacks. Third, after the backdoor of a DPBA is implanted, the DPBA cannot work if the victim QNN is retrained with the users' new clean datasets. The new clean datasets force the victim QNN to forget the predefined behavior. Fourth, a DPBA can achieve two conflicting goals, high clean data accuracy (i.e., accuracy when the backdoor is disabled) and high attack success rate (prediction ratio to the target class when the backdoor is activated) simultaneously on a classical neural network [6]. Unfortunately, we find a DPBA obtains either high clean data accuracy or high attack success rate, but not both, on a QNN, due to its shallow network architecture on a Noisy Intermediate-Scale Quantum (NISQ) computer.

To achieve high accuracy, recent work [9, 10] designs QNN circuits (aka, ansatzes) by automated searches such as deep reinforcement learning. Unfortunately, most auto-designed QNN circuits are inscrutable, since they contain sophisticated quantum circuit components which are often hard for humans to inspect. Even randomly-wired quantum gates [10] can obtain competitive accuracy on standard QNN benchmarks. This provides attackers an opportunity to insert malicious circuit-level backdoors. However, no prior work considers a circuit backdoor against QNNs.

In this paper, we propose a circuit-level backdoor attack, *QTrojan*. QTrojan adds few quantum gates as the backdoor around the encoding layer of a victim QNN. QTrojan uses several lines in a server-specific configuration file as the trigger. When QTrojan is disabled, the victim QNN achieves the same accuracy as its clean counterpart. However, after QTrojan is enabled, the victim QNN always predicts a predefined target class, regardless of the inputs. Compared to a prior DPBA, QTrojan improves the clean data accuracy by 21% and the attack success rate by 19.9%.

2. BACKGROUND

2.1. Quantum Cloud Computing

Due to the high cost of NISQ computers, average users typically run QNNs via quantum cloud services, as shown in Figure 1. A user designs a QNN circuit, trains it, compiles the trained circuit and input data into quantum analog pulses, and sends the pulse sequence to a cloud NISQ server. The server applies the pulse sequence to qubits, and returns the result to the user. A prediction result is a probability vector, where the predicted class is computed by *softmax*.

2.2. Variational Quantum Circuit

In a classical neural network [6], the first multiple layers generate an embedding for an input, e.g., a sentence or an image, while the last layer maps the embedding to a probability vector. On the contrary, in a QNN [3, 4, 8], these functions

THIS WORK WAS SUPPORTED IN PART BY NSF CCF-1908992, CCF1909509, CCF-2105972, AND NSF CAREER AWARD CNS-2143120.

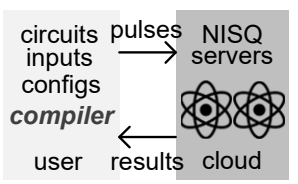


Fig. 1: QNNs in cloud.

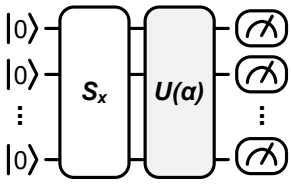


Fig. 2: A VQC example.

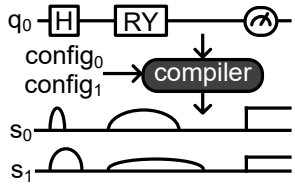


Fig. 3: Quantum compilation.

Schemes	DPBA	**QTrojan**
No Trigger in Inputs	✗	✓
No Training Data	✗	✓
No Training Process	✗	✓
Works after Retraining	✗	✓

Table 1: DPBA vs QTrojan.

are implemented by a variational quantum circuit (VQC) [9] composed of an encoding layer S_x, a variational circuit block $U(\alpha)$, and a measuring layer, as shown in Figure 2. The quantum state ρ_x is prepared to represent the classical input data x by S_x. ρ_x is entangled and rotated to generate the processed state $\tilde{\rho}_x$ by $U(\alpha)$. The probability vector $\hat{y}[\tilde{\rho}_x]$ is generated by measuring $\tilde{\rho}_x$ for multiple times. S_x has its fixed function and thus is not trainable. The VQC training is to find the quantum gate rotation angles in $U(\alpha)$ that minimize a cost function between predictions and ground truth labels.

2.3. Quantum Compiler

To run a QNN on a cloud-based NISQ server, as Figure 3 exhibits, the user has to first locally compile the QNN VQC and its input data into a sequence of analog pulses [11, 12] with a server-specific configuration file. The sequence of pulses manipulates qubits to implement QNN inferences on cloud-based NISQ computers. A pulse [12] can be specified by an integer duration, a complex amplitude, and the standard deviation. Different servers support different pulse durations, maximum pulse amplitudes, and pulse channel numbers. Even the same server requires different values for its pulse error calibration at different times. A configuration file [11, 12] describing the latest information of a NISQ server enables the compiler to generate a high-quality pulse sequence for a QNN and its input data. When the same QNN circuit has a new piece of input data, the compiler has to re-compile the circuit with the new input. To minimize noises and errors on a NISQ server, it is important for the quantum compiler to download and use its latest configuration file before each compilation.

2.4. Threat Model

For QTrojan, we assume the victim users receive a QNN circuit from the attacker and train the variational block of the circuit with their own datasets. This case frequently happens, since most average users without domain knowledge tend to download a circuit architecture designed by domain experts from the internet, and train it with their own datasets. Both the quantum compiler and NISQ servers are trustworthy in our threat model. However, we assume the attacker can insert triggers into a configuration file and the victim user needs to download the configuration file to minimize noises and errors before each compilation. With a benign configuration file, the QNN works normally for all inputs. On the contrary, the QNN using a configuration file with a trigger classifies all inputs into a predefined target class. Unlike the white-box threat model used by prior DPBAs [6, 7], we assume a more conservative threat model, where the attacker does not require the original training dataset, training details including training method and hyper-parameters, long retraining process, or any meaningful test dataset. Furthermore, QTrojan still works even after the victim QNN is retrained with the victim users' new clean datasets.

2.5. Backdoor Attacks in Classical Neural Networks

Attackers inject backdoors [6, 7] into a classical neural network during a time-consuming training process, so that the victim network behaves normally on benign samples whereas its predictions are consistently changed to a predefined target class if the backdoor is activated by a nontrivial-size trigger. A typical way to inject the backdoor is poisoning the original training dataset [7], i.e., some training samples are modified by adding the trigger and paired with the predefined target label. However, it is difficult to access the original training dataset or use a long training process to attack the victim network in both classical and quantum domains. Almost all data-poisoning-based backdoors [6, 7] can be eliminated if the users retrain the victim model with their new clean datasets. Moreover, due to the limited input dimension of state-of-the-art QNNs, embedding a nontrivial-size trigger makes the backdoor attack less stealthy. Although backdoor attacks against classical neural networks achieve both high clean data accuracy and high attack success rate, a QNN may not be able to learn both the clean data task and the trigger-embedded data task well, due to its shallow network architecture on a NISQ computer. In this paper, we propose circuit-level QTrojan to perform backdoor attacks against QNNs. As Table 1 shows, QTrojan does not need to access the original dataset, use a long training process, or attach a trigger to input data. QTrojan can still work even after the user retrains the victim QNN with their new clean datasets.

2.6. Other Quantum-related Backdoor Attacks

For other quantum-related backdoor attacks, prior work [13] creates backdoors in quantum communication systems for key distribution and coin-tossing via laser damage. To the best of our knowledge, QTrojan is the first circuit-level backdoor attack against QNNs.

3. QTROJAN

3.1. Overview

We propose QTrojan to mask the original input of a victim QNN and force its encoding layer to output fixed quantum states belonging to the predefined target class by integrating a

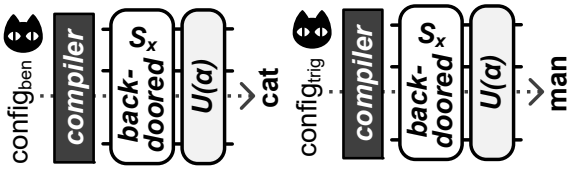


Fig. 4: The overview of QTrojan.

few quantum gates into the victim QNN VQC. More specifically, we add two extra layers around the encoding layer of the victim QNN. Unlike conventional DPBAs, which embed a trigger into input data, these two additional QTrojan layers can be disabled or activated by a configuration file via a trustworthy quantum compiler. As Figure 4 shows, using a benign configuration file, the victim QNN classifies an image (cat) normally to its class ("cat"). However, a configuration file with a trigger causes the victim QNN to maliciously classify the image (cat) into a predefined target class ("man").

3.2. A Backdoored Data Encoding Layer

In this section, we describe how to backdoor the data encoding layer of a victim QNN by QTrojan.

Angle Encoding. The first step in a QNN is to convert classical input data $\mathcal{X}$ to n-qubit quantum states D_n by its data encoding layer S_x. The most widely adopted data encoding methods in state-of-the-art QNNs are *amplitude* encoding and *angle* encoding [14]. Although amplitude encoding represents N features by $n = \log_2(N)$ qubits, its preparation requires a $\mathcal{O}(2^n)$ circuit depth, making a QNN more error-prone [15]. In contrast, angle encoding requires N qubits with a constant depth (i.e., less than three layers) quantum circuit to represent N features, and is thus more suitable for NISQ devices due to its noise immunity and simplicity of implementation [1, 4]. In this work, we adopt the angle encoding defined in [14] as follows:

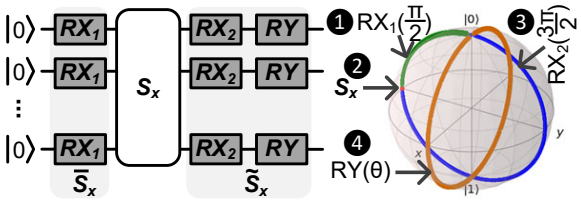
$$|\mathbf{x}\rangle = \bigotimes_{i=1}^{N} \cos(x_i)|0\rangle + \sin(x_i)|1\rangle \quad (1)$$

where $\mathbf{x} = [x_1, \ldots, x_N]$ is a N-feature vector, and $\otimes$ is tensor product operator. To represent more features by the same number of qubits, there is a denser version of angle encoding [14] defined as

$$|\mathbf{x}\rangle = \bigotimes_{i=1}^{\lceil N/2 \rceil} \cos(\pi x_{2i-1})|0\rangle + e^{2\pi i x_{2i}} \sin(\pi x_{2i-1})|1\rangle \quad (2)$$

where $2N$ features are represented by N qubits. In this paper, we focus on designing backdoors for angle encoding and dense angle encoding.

A Backdoor Against Angle Encoding. We present a circuit backdoor for QTrojan to hijack the data encoding layer S_x of a victim QNN. As Figure 5(a) shows, S_x is sandwiched by the backdoor consisting of a pre-encoding layer $\bar{S}_x$ and a post-encoding layer $\tilde{S}_x$. $\bar{S}_x$ is composed of N RX gates, each of which works on a qubit. And $\tilde{S}_x$ has N pairs of RX-RY gates, each of which manipulates one qubit. By rotating the qubit from $|0\rangle$ by an angle, the original angle encoding layer



(a) Backdoored Encoding Layer (b) Angle Adjustment

Fig. 5: The backdoored encoding layer of QTrojan.

S_x encodes the qubit as a data point on the circle of $y = 0$ in a Bloch sphere shown in Figure 5(b). In our circuit backdoor, ❶ the pre-encoding layer $\bar{S}_x$ moves the qubit to the leftmost point of the Bloch sphere. ❷ Although S_x is normally applied on the qubit, the qubit is still constrained in the leftmost point of the Bloch sphere. ❸ The RX_2 gate of $\tilde{S}_x$ rotates the qubit by $3/2\pi$ and moves it back to $|0\rangle$. ❸ Finally, the RY gate of $\tilde{S}_x$ rotates the qubit by θ predefined by the attacker.

3.3. A Trigger in Configuration Files

To generate the pulses of a QNN that can accurately and efficiently run on a NISQ server, a quantum compiler acquires the server details from a configuration file [12], which also defines pulse values for error calibration. When there is a new input, based on the latest configuration file, the quantum compiler converts the QNN circuit and new input into a sequence of pulses. The attacker uses a few lines as a trigger in the configuration file to respectively set $\pi/2$, $3\pi/2$, and θ to the RX_1, RX_2, and RY gates of the backdoored encoding layer, so that QTrojan can be activated. The trigger can be disguised as pulse error calibration for the data encoding layer.

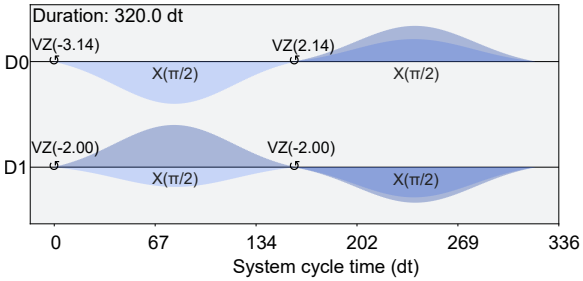


Fig. 6: The pulse-level overhead of QTrojan.

3.4. Pulse-level Overhead

We compiled the original angle encoding layer S_x and the backdoored encoding layer ($\bar{S}_x$+ S_x + $\tilde{S}_x$) of QTrojan into pulse sequences D_0 and D_1 respectively, as shown in Figure 6. Although two data encoding layers have slightly different pulse amplitudes, QTrojan does not add the circuit depth at all and can be completed by a sequence of pulses having the same duration as S_x.

3.5. Potential Defenses

CMOS-based circuit-level backdoor tests and detections [16] may be helpful to prevent QTrojan.

Schemes	QNN (%)	DPBA (%)		QTrojan (%)	
	accuracy	CDA	ASR	CDA	ASR
MNIST-2	98.25	91.56	99.5	98.25	100
MNIST-4	58.6	43	68.75	58.6	100

Table 2: The comparison between DBPA and QTrojan (MNIST-X: X-group classification on MNIST; CDA: clean data accuracy; ASR: attack success rate).

4. EXPERIMENTAL METHODOLOGY

Datasets. We adopted MNIST [17] to evaluate QTrojan. Since NISQ computers support only a limited number of qubits, we down-sampled the 28×28 images in MNIST to 4×4 through principal component analysis, similar to prior work [18, 19]. We studied only 2-group (0,1) and 4-group (0-3) classifications on MNIST. We also built a classical and quantum hybrid LSTM (QLSTM) [4] to learn the sequential dependency in periodic `sin` functions.

Circuit. For MNIST, we designed a 16-qubit QNN circuit composed of an angle encoding layer, 2 parameterized blocks, and a measurement layer. Each parameterized block has a ROT layer and a ring-connected CRX layer. To learn the temporal `sin` curve, we built a 4-qubit QLSTM circuit consisting of a dense angle encoding layer, 2 parameterized blocks, and a measurement layer. Each parameterized block has a ROT layer and a ring-connected CNOT layer.

Simulation. We built QNNs and QTrojan using Qiskit [12]. We considered the *FakeAlmaden* as our backend and noise model in Qiskit. We used an ADAM optimizer, a learning rate of 1e-3, and a weight decay value of 1e-4 as default hyperparameters. The learning rate of QLSTM is 1e-2.

Metrics. We define clean data accuracy (CDA) and attack success rate (ASR) to study QTrojan. CDA means the percentage of input images classified into their corresponding correct classes with a benign configuration file. With a higher CDA, it is more difficult to identify a backdoored QNN. ASR indicates the percentage of input images with a triggered configuration file classified into the predefined target class. The higher ASR QTrojan can achieve, the more effective it is.

5. RESULTS

DPBA against QNN. We performed DPBA on 2/4-group MNIST classification (MNIST-2/4). As Table 2 shows, compared to the clean QNN, the CDA of DPBA degrades by 6.8% on MNIST-2 and 26.6% on MNIST-4, although its ASR is higher than the clean QNN accuracy. This is because the learning capability of state-of-the-art QNN circuits is limited by their shallow architectures on NISQ computers. The QNN simply cannot learn both the MNSIT classification task and the backdoored task well simultaneously. Besides the low CDA, the stealthiness of DPBA on a QNN is still damaged by its 1-qubit trigger (6.25% of a 16-qubit input) and its dependence on the original training data. Moreover, a few-epoch retraining of the DPBA-backdoored QNN with new training datasets can greatly reduce the ASR of DPBA.

Schemes	ASR of QTrojan (%)			
	1 qubit	2 qubits	3 qubits	4 qubits
MNIST-2	100	100	100	100
MNIST-4	61.18	72.92	81.4	100

Table 3: The ASR of QTrojan with only $\bar{S}_x$ on few qubits.

QTrojan against QNN. We also implanted QTrojan in MNIST-2/4. As Table 2 highlights, the CDA of QTrojan is exactly the same as the accuracy of the clean QNN, when its pre-encoding layer $\bar{S}_x$ and post-encoding layer $\tilde{S}_x$ are disabled. In contrast, when $\bar{S}_x$ and $\tilde{S}_x$ are activated by a triggered configuration file, the ASR of QTrojan is 100% in both tasks. Compared to DPBA, QTrojan improves the CDA by 21% and the ASR by 19.9% on average. Due to the 100% ASR, QTrojan may not require all its components to attack the victim QNN. Next, we show adding only a pre-encoding layer $\bar{S}_x$, but no post-encoding layer, to some qubits, still makes QTrojan strong enough to achieve a high ASR.

Partial QTrojan against QNN. We added $\bar{S}_x$ (RX gates) on $1 \sim 4$ qubits of the victim QNN. When QTrojan is disabled, the CDA of the backdoored QNN is still the same as its clean counterpart. As Table 3 shows, for the simple task MNIST-2, adding an RX gate on one qubit of the victim QNN yields a 100% ASR. On the contrary, for the complex task MNIST-4, adding one RX gate on one qubit results in only a 68.75% ASR, which is lower than that of DPBA. We then added more RX gates on more qubits. When two qubits have RX gates, QTrojan improves the ASR by 6% over DPBA. When we added 4 RX gates on 4 qubits, the victim QNN ASR achieves 100%. Therefore, we can see that only a pre-encoding layer $\bar{S}_x$ on some qubits is powerful enough to abuse the victim 16-qubit QNN.

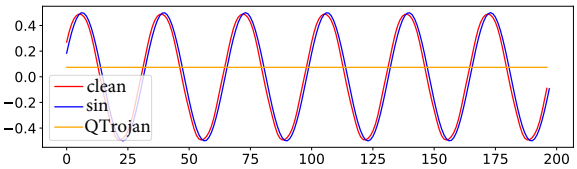


Fig. 7: QTrojan against QLSTM.

QTrojan against QLSTM. We also implanted QTrojan to attack a QLSTM model. As Figure 7 shows, QLSTM (clean) is trained to learn a `sin` curve (sin). Through two RX gates and one RY gate on each qubit, QTrojan forces the prediction result of QLSTM to a horizontal line. It is difficult for DPBA to insert a trigger on the consecutive inputs of QLSTM. On the contrary, QTrojan can easily activate RX gates to attack QLSTM by a server-specific configuration file.

6. CONCLUSION

In this paper, we propose a circuit-level backdoor attack, *QTrojan*, against quantum machine learning. QTrojan can be implemented by few quantum gates attached to victim QNN circuits. Compared to DPBA, QTrojan improves the CDA by 21% and the ASR by 19.9% on average.

7. REFERENCES

[1] Jindi Wu, Zeyi Tao, and Qun Li, "Scalable quantum neural networks for classification," *arXiv preprint arXiv:2208.07719*, 2022.

[2] Jan-Nico Zaech, Alexander Liniger, Martin Danelljan, Dengxin Dai, and Luc Van Gool, "Adiabatic quantum computing for multi object tracking," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, June 2022, pp. 8811–8822.

[3] Riccardo Di Sipio, Jia-Hong Huang, Samuel Yen-Chi Chen, Stefano Mangini, and Marcel Worring, "The dawn of quantum natural language processing," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2022, pp. 8612–8616.

[4] Samuel Yen-Chi Chen, Shinjae Yoo, and Yao-Lung L. Fang, "Quantum long short-term memory," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2022, pp. 8622–8626.

[5] Daniel J. Egger, Claudio Gambella, Jakub Marecek, Scott McFaddin, Martin Mevissen, Rudy Raymond, Andrea Simonetto, Stefan Woerner, and Elena Yndurain, "Quantum computing for finance: State-of-the-art and future prospects," *IEEE Transactions on Quantum Engineering*, vol. 1, pp. 1–24, 2020.

[6] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg, "Badnets: Identifying vulnerabilities in the machine learning model supply chain," *Machine Learning and Computer Security Workshop*, 2017.

[7] Yingqi Liu, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Weihang Wang, and Xiangyu Zhang, "Trojaning attack on neural networks," in *Annual Network and Distributed System Security Symposium*. 2018, The Internet Society.

[8] Jacob Biamonte, Peter Wittek, Nicola Pancotti, Patrick Rebentrost, Nathan Wiebe, and Seth Lloyd, "Quantum machine learning," *Nature*, vol. 549, no. 7671, pp. 195–202, 2017.

[9] Yuxuan Du, Tao Huang, Shan You, Min-Hsiu Hsieh, and Dacheng Tao, "Quantum circuit architecture search for variational quantum algorithms," *npj Quantum Information*, vol. 8, no. 1, pp. 1–8, 2022.

[10] Hanrui Wang, Yongshan Ding, Jiaqi Gu, Yujun Lin, David Z Pan, Frederic T Chong, and Song Han, "Quantumnas: oise-adaptive search for robust quantum circuits," in *IEEE International Symposium on High-Performance Computer Architecture*, 2022, pp. 692–708.

[11] David C McKay et al., "Qiskit backend specifications for openqasm and openpulse experiments," *arXiv preprint arXiv:1809.03452*, 2018.

[12] Thomas Alexander, Naoki Kanazawa, Daniel J Egger, Lauren Capelluto, Christopher J Wood, Ali Javadi-Abhari, and David C McKay, "Qiskit pulse: Programming quantum computers through the cloud with pulses," *Quantum Science and Technology*, vol. 5, no. 4, pp. 044006, 2020.

[13] Vadim Makarov, Jean-Philippe Bourgoin, Poompong Chaiwongkhot, Mathieu Gagné, Thomas Jennewein, Sarah Kaiser, Raman Kashyap, Matthieu Legré, Carter Minshull, and Shihan Sajeed, "Creation of backdoors in quantum communications via laser damage," *Physical Review A*, vol. 94, no. 3, pp. 030302, 2016.

[14] Ryan LaRose and Brian Coyle, "Robust data encodings for quantum classifiers," *Physical Review A*, vol. 102, pp. 032420, Sep 2020.

[15] Cheng Chu, Nai-Hui Chia, Lei Jiang, and Fan Chen, "Qmlp: An error-tolerant nonlinear quantum mlp architecture using parameterized two-qubit gates," in *ACM/IEEE International Symposium on Low Power Electronics and Design*, 2022.

[16] He Li, Qiang Liu, and Jiliang Zhang, "A survey of hardware trojan threat and defense," *Integration*, vol. 55, pp. 426–437, 2016.

[17] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[18] Maria Schuld, Alex Bocharov, Krysta M. Svore, and Nathan Wiebe, "Circuit-centric quantum classifiers," *Physical Review A*, vol. 101, pp. 032308, Mar 2020.

[19] Edward Farhi and Hartmut Neven, "Classification with quantum neural networks on near term processors," *arXiv preprint arXiv:1802.06002*, 2018.