Experimental Attacks on Quantum Computing and Quantum Machine Learning

Marc Maußner[1] and Volker Reers[2]
[1]Business Unit Transportation, infoteam Software AG, Bubenreuth, Germany
[2]Qseidon GmbH, Gütersloh, Germany

Marc.Maussner@infoteam.de
Volker.Reers@qseidon.de

Abstract: 2025 marks the UN International Year of Quantum Science and Technology. We expect this emerging technology to enter application in real world industry use cases in the next few years. The current focus in research and industry lies on the nominal function of quantum computing to show its "usefulness" of the technology. Hence, one should also start considering aspects of cybersecurity to prepare for currently known attacks when quantum computing is ready to be used in industry. This work provides an overview on quantum computing (QC) with the underlying used quantum mechanical phenomena. We set special highlight on quantum machine learning (QML) as it is expected to have higher expressiveness and shorter training times. The quantum machine learning lifecycle will be presented in detail with the current state of the art of theory and research of cybersecurity attacks. We aim to use this knowledge and start to construct simple experiments as proof-of-concepts for attacking assets in the CIA triad. These proof-of-concepts will set the base for attacks on more relevant use cases of QML models, leading to thoughts and experiments for countermeasures to be set up and put in place. The conclusion will consist of future work and the currently not in detail investigated sections of the lifecycle.

Keywords: Quantum computing, QML, Machine-Learning lifecycle, Countermeasures, CIA triad

1. Introduction

Quantum computing is about to revolutionize computational power, offering solutions to problems intractable for classical systems. Among its applications, Quantum Machine Learning (QML) has emerged as a promising fusion of quantum computing and machine learning, capable of enhancing data analysis, optimization, and classification tasks. As QML transitions from theory to practice, it also introduces unique vulnerabilities that demand thorough investigation to ensure secure deployment. Classical machine learning has demonstrated susceptibility to adversarial attacks, data poisoning, and model extraction (Kempka, C. D., Schaad, A. P. 2022). As QML evolves, it is imperative to examine whether such vulnerabilities persist or take new forms in the quantum realm. The quantum properties of superposition and entanglement offer computational advantages but may also create novel attack surfaces, highlighting the need for dedicated research into QML security.

This paper addresses this research gap by investigating the security of QML models. We developed QML algorithms and conducted experimental attacks to illustrate the practical feasibility of attacking QML systems and expose vulnerabilities specific to the quantum domain. Our objectives are threefold: to demonstrate practical attacks on QML models, analyse these findings, and propose robust countermeasures. With this, we aim to lay the foundation for secure and trustworthy QML systems.

The remainder of this paper is structured as follows: Section 2 reviews related work on QC and QML. Section 3 details our experimental setup, including models and attack scenarios. Section 4 presents results and analysis of the attacks, while Section 5 proposes countermeasures and their practical implications. Finally, Section 6 concludes with key findings and future research directions.

As quantum technologies advance, understanding and mitigating QML vulnerabilities are essential for secure adoption. This work contributes to the field by exposing risks and providing actionable insights for enhancing QML robustness, paving the way for secure quantum applications.

2. Quantum Computing and Quantum Machine Learning

This chapter introduces most important aspects of Quantum Computing and Quantum Machine Learning. Also the general Quantum Machine Learning Lifecycle is presented.

2.1 Quantum Computing

Quantum computing represents a paradigm shift in computation, leveraging the principles of quantum mechanics to process information in ways fundamentally different from classical computing. At its core, quantum computing exploits phenomena such as superposition, entanglement, and quantum interference to perform calculations that would be infeasible or impossible for classical systems. These quantum mechanical

properties enable quantum computers to solve certain classes of problems exponentially faster than their classical counterparts, making them particularly suited for tasks such as cryptography, optimization, and large-scale data analysis.

The fundamental unit of quantum computation is the quantum bit, or qubit, which differs significantly from the classical bit. While a classical bit represents information as either 0 or 1, a qubit can exist in a superposition of states. The superposition effectively represents both 0 and 1 simultaneously at runtime until the qubit is measured and the superposition collapses to either 0 or 1. This property, combined with entanglement – a phenomenon where the state of one qubit is correlated on another – allows quantum computers to perform parallel computations at an unprecedented scale. However, harnessing these properties requires highly controlled environments, as qubits are extremely sensitive to noise and decoherence, which can disrupt their quantum states.

Quantum computing remains an area of rapid development, with continuous advancements in hardware, algorithms, and applications. In addition to topics such as quantum optimization, quantum chemistry and quantum communication, to name a few, new fields of application such as quantum machine learning are also emerging.

2.2 Quantum Machine Learning

Quantum Machine Learning (QML) represents a transformative approach to machine learning, combining the computational power of quantum computers with classical techniques for machine learning. It seeks to leverage the unique properties of quantum computing to enhance the efficiency, scalability, and accuracy of machine learning algorithms. By introducing quantum computing into the machine learning pipeline, QML opens new avenues for tackling problems that are infeasible for classical systems.

At the beginning of the QML pipeline, a quantum computer can be used to preprocess classical data to extract meaningful features or reduce dimensionality, enabling better input for classical machine learning models. Techniques such as quantum principal component analysis (qPCA) and quantum feature encoding can significantly enhance the performance of downstream tasks. Quantum algorithms can also be employed in the postprocessing phase at the end of the QML pipeline. The task here is to analyse and refine the outputs of classical machine learning models. This approach is particularly valuable for tasks requiring complex correlations or additional optimization steps.

Quanvolution, or quantum convolution, is a hybrid technique that introduces a quantum layer into classical machine learning architectures. Here, quantum circuits act as a preprocessing layer, transforming classical data into a quantum-enhanced feature space before passing it to the rest of the classical model. This approach benefits from the expressive power of quantum transformations while retaining the scalability of classical models. In comparison, Quantum Model Replacement is an approach in which the entire classical machine learning model is replaced by a quantum model. For example, a quantum neural network (QNN) operates on qubits instead of classical bits and employs quantum gates for computation. These models aim to exploit quantum parallelism to solve high-dimensional optimization problems and pattern recognition tasks more efficiently (Weigold, M. et al. 2020).

While QML is still in its infancy, ongoing research and experimentation continue to uncover its potential for practical applications (Abbas, A. et al. 2021). By enhancing traditional machine learning techniques with quantum advantages, QML holds the promise of driving innovation across fields such as natural language processing, drug discovery, and financial modelling. This means that early attention must be paid to security during development and that potential vulnerabilities and attack vectors represent an important field of research for QML.

2.3 Quantum Machine Learning Lifecycle

Like the Quantum Software Lifecycle by Weder, B. et al. (2020), the Quantum Machine Learning (QML) lifecycle encapsulates the end-to-end process of developing, deploying, and maintaining QML models. It integrates quantum-specific considerations into traditional machine learning workflows and emphasizes the unique challenges and opportunities presented by quantum systems. The lifecycle typically consists of the following stages:

- Problem Definition: Clearly define the problem and identify whether quantum advantages are applicable. This involves assessing the feasibility of using QML for the task and determining the appropriate quantum algorithms or hybrid solutions.

- Data Preparation: Quantum systems often require classical data to be encoded into quantum states. Techniques such as amplitude encoding, basis encoding, or angle encoding are used to preprocess data for quantum operations.
- Model Design and Training: Design QML models, such as quantum neural networks or hybrid quantum-classical architectures. Training involves optimizing parameters to minimize loss functions, often combining classical optimization techniques with quantum circuit executions.
- Validation and Testing: Evaluate the model's performance using metrics tailored to the problem domain. Validation includes assessing robustness against noise, errors, and adversarial attacks specific to quantum environments.
- Deployment: Deploy QML models on appropriate quantum hardware or simulators. This stage requires careful consideration of hardware constraints, such as qubit count, coherence times, and gate fidelity.
- Monitoring and Maintenance: Continuously monitor the deployed model for performance and security. Maintenance involves addressing hardware-specific issues, refining circuits for improved efficiency, and updating models to account for evolving data or new threats.

By following this lifecycle, organizations can systematically build and manage QML models, ensuring their effectiveness and resilience in real-world applications.

3. Methodology

This chapter presents the methodologies used in this study. The specific experimental setups and descriptions of attacks and assets is given in the following subchapters.

3.1 Attacks Using Crosstalk

We aim to attack quantum circuits and QML models in the Data Preparation and Model Design and Training lifecycle phases. We try to influence the outputs of target circuits by use of "Tempering Attacks" and "Fault Injection Attacks" using the "crosstalk" of the underlying qubits in hardware.

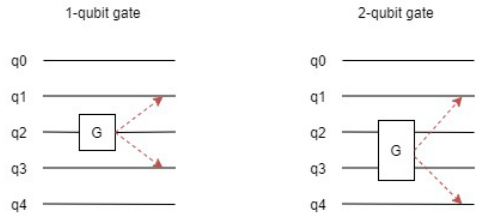

Figure 1: Crosstalk of gate G on neighbouring qubits for 1-qubit and 2-qubit gates

As gates on superconducting hardware introduce errors to neighbouring qubits (qubits connected in the coupling map) we try to use this mechanism to temper the results of specific target qubits. Figure 1 shows this behaviour for 1-qubit gates and 2-qubit gates where at least q1, q2, q3 and q4 are connected in the coupling map.

For the first experiment we follow Saki et al. (2020) and use the same coupling map for our fake device as shown in Figure 2. This means that we don't have an all-to-all connection of our 5 qubits. The circuit representing a GHZ state of 4 from the 5 qubits system is our target, see Figure 3. Normally – without attacks or noise – leading to only 2 resulting states "00000" and "01111" with equal probability of 50% of counts of the executed shots.

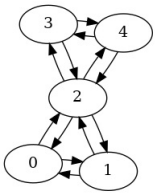

Figure 2: Coupling map of fake-device

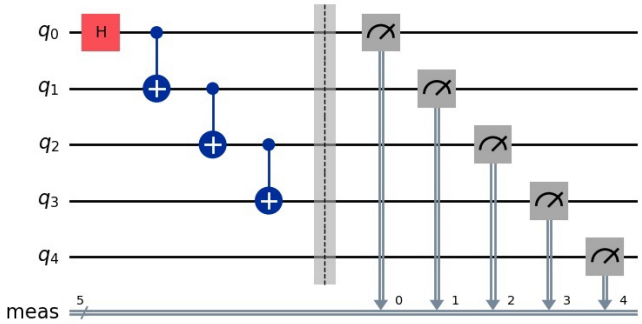

Figure 3: Target circuit - GHZ State of q0, q1, q2, q3 with q4 staying |0> (in qiskit-style)

In Saki et al. (2020) two different ways of attacks are described. Attacking by use of 1-qubit gates like X, Z, Y, H or by use of 2-qubit gates like CNOT. For a detailed overview of quantum gates see Wikipedia (2025)

At first, we try the 1-qubit gate attack with the H-gate. That means we add a new circuit to the target circuit where 10000 H-gates are to be run on qubit q4. Whereby the execution of an even count of Hadamard-gates should not change the results of the target circuit – when run by Statevector-simulators or on QPUs (Quantum Processing Units) without noise and errors.

Second, we try the 2-qubit gate attack with the CNOT-gate. That means we add a new circuit to the target circuit where 1000 CNOT-gates are to be run on qubits q3 and q4. This tempering should also not affect the result of the target circuit.

The layout of the circuits with the tempered circuits is shown in Figure 4.

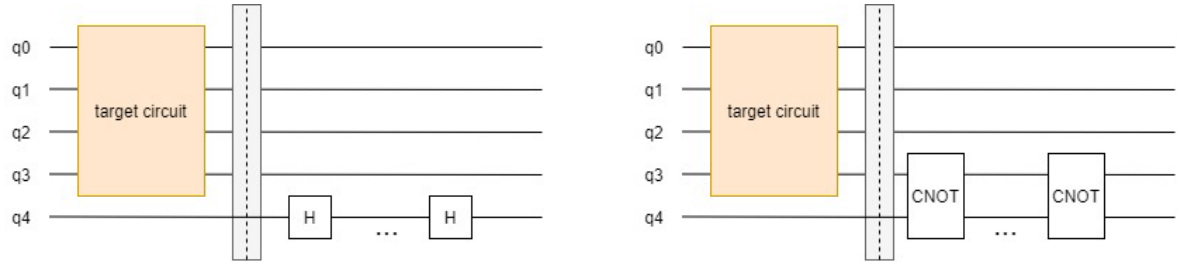

Figure 4: Layout of tempering the behaviour and injecting faults to the results with use of 1-qubit gates (left) and 2-qubit gates (right)

To ensure that the attack is not only feasible on simulators and fake-devices, but also on real QPUs we aim for running our experiments directly on IBM hardware. Therefore, we will not change the layout of the experiments but only the execution device.

To estimate how feasible our attack is and how easy or difficult it is we aim to estimate the amounts of 1-qubit and 2-qubit gates needed to see an effect in results. We will use a looped execution where we stop when we have more than 2 results with counts over 10% of the whole execution shots. The general attack layout being the same as above.

3.2 Attacks Tampering Transpilation

We aim to attack quantum circuits and QML models in the Data Preparation and Model Design and Training lifecycle phases. We try to influence the outputs of target circuits by use of “Fault injection attack” – like the *Quantum Trojan Virus – and* “Model manipulation attack” using a modified transpiler pass.

As target circuit we take modified GHZ state of q0, q1, q2 with an inserted X-gate on q1, see Figure 5. Ideally with errorless gates and/or on simulators without noise we get there 2 resulting states “00010” and “00101”.

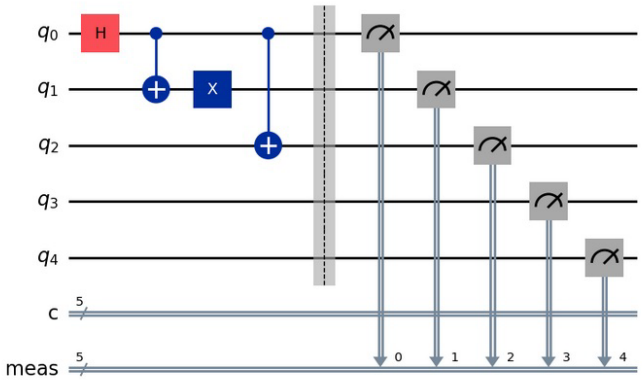

Figure 5: Target circuit for transpilation, modified GHZ state with additional X-gate (in qiskit-style)

We setup a fake-device with coupling map as shown in Figure 2. We aim to use the qiskit transpiler as described in Qiskit Development Team (2023) and transpiler-pass from Qiskit Development Team (2025) to adapt the circuit to the underlying available gates and the coupling map of the defined fake-device.

We aim to create a new transformation-pass that will insert an additional X-gate when an X-gate in the target circuit is encountered. After that we install this transformation-pass in our PassManager and transpile the target circuit. After that we will draw the transpiled circuit and run it to verify the successful attack.

3.3 Attacks Tampering Input Data

We aim to attack the output of our QML models by perturbating the input data. We aim to adapt the OnePixel attacks and PGD and FGSM attacks as already known in classical machine learning to QML with different image datasets.

At first, we aim to implement a OnePixel attack, based upon Su et al. (2019), where one pixel of an input image is manipulated to provoke a false classification. As image dataset we take the MNIST dataset from scikit-learn where each image is reflected by 8x8 pixels with values in range of [0,255]. Two examples of the dataset are shown in Figure 6.

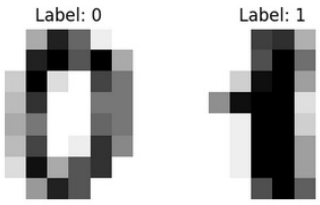

Figure 6: Example images of the MNIST dataset from scikit-learn

For this setup we will only consider the two classes with labels “0” and “1”. We will encode these 64 values into 6-qubits by Amplitude-Embedding technique and compose this with a trainable ansatz. The abstract model layout is shown in Figure 6. Afterwards we will train our model with an 80% split train- and test-set.

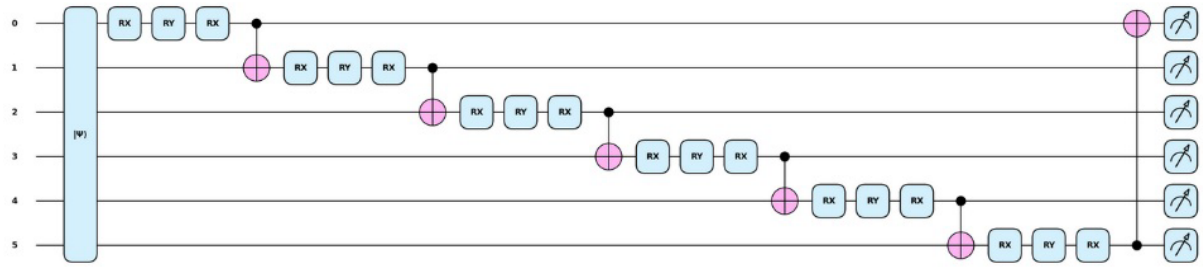

Figure 7: Target model for OnePixel attack combining Amplitude-Embedding and the trainable ansatz (in pennylane-style)

After training we randomly choose one image from our test-set and change randomly one pixel either from a darker grey colour to a lighter grey colour or vis-versa and check the prediction of the model. If we have one false classification then we stop, else we will iterate and change another random pixel of the same original image.

For the next experiment we aim to perturbate the image data not randomly but by mathematical construction with PGD (Projected Gradient Descent) as described in Kolter et al. (2025). We take the example given by Wendlinger et al. (2025). This uses the "Plus-Minus"-image dataset from pennylane, see examples in Figure 7. Our model uses 8 qubits and consists of a StronglyEntanglingLayer, see pennylane Development Team (2025), with only 4 of the 8 qubits being measured as we only expect 4 classes as the result. We train our model with a train-set of 1000 images and a test-set of 200 images. We aim to check the accuracy of our model and the predictions for 4 chosen representatives of each class. Then we attack the model by perturbating these representatives with the PGD-method.

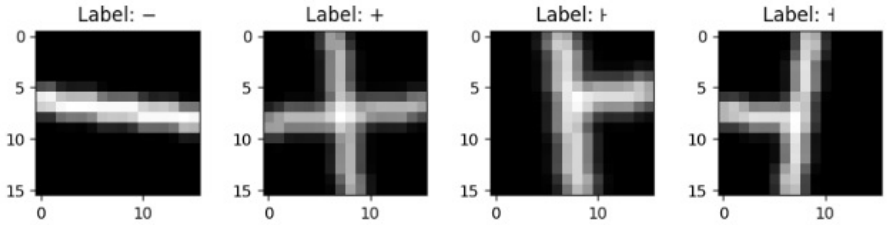

Figure 8: Example images of the Plus-Minus dataset of pennylane

We modify our experiment for PGD and adapt the perturbation rule for FGSM (Fast Gradient Sign Method) attack as described by Kolter et al. (2025). We use the same trained model as for PGD but afterwards predict results for the FGSM perturbed images.

4. Results and Analysis

In the following we will provide the results in subchapters for the experiments described in the methodology chapter.

4.1 Results of Crosstalk Attack

Figure 9 shows the results of our tempered circuits when run with qiskit's sampler device with 1000 shots. On the left side for the described 1-qubit gate attack and on the right side for the 2-qubit gate attack. We use as underlying device the fake-device.

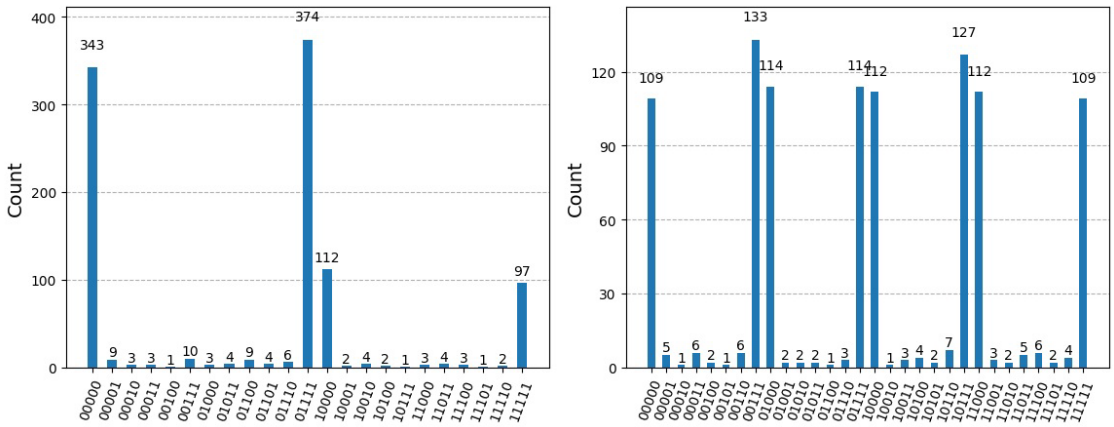

Figure 9: Results of crosstalk attack for use of 1-qubit gates (left) and 2-qubits gates (right) when executed on fake-device with 1000 shots

We see that we now not only get the expected states "00000" and "01111" but for the 1-qubit attack also states "10000" and "11111". This reflects that our introduced error of executing H-gates lead to an impact of the gate directly.

For the 2-qubit gates we see now also the states "00111", "01000", "10000", "10111", "11000" and "11111" appearing with almost equal probability to the expected states.

Figure 10 shows the results of the crosstalk attack when executed on a real QPU from IBM (here: ibm_sherbrooke). We see that for the 1-qubit attack we even get more representative resulting states “01000” and “10111”. This is because of the different error rates of the QPU for measurement errors and fidelity T1 and T2 in the underlying gates.

The histograms for the 2-qubit gate attack look similar for execution on HW and on simulation using fake-device.

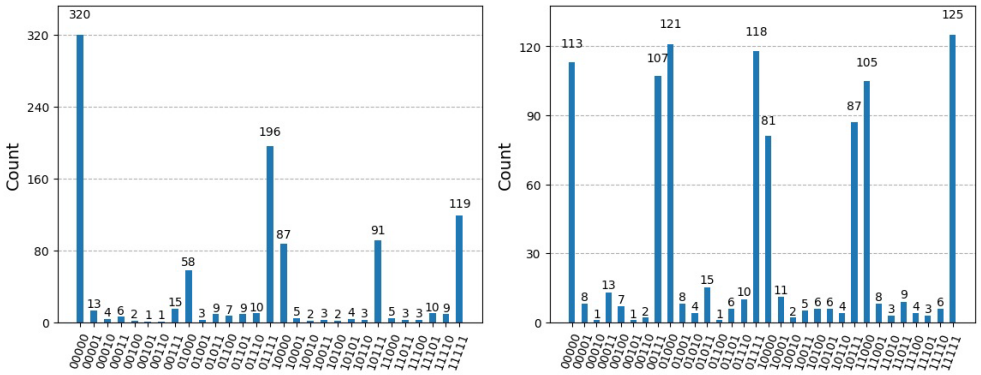

Figure 10: Results of crosstalk attack for use of 1-qubit gates (left) and 2-qubit gates (right) when executed on IBM QPU - ibm_sherbrooke with 1000 shots

We also explored the number of gates needed to introduce an error in results reflecting the number of counts of erroneous states being higher than 10% of the number of shots. We ran our experiment on the fake-device simulating ibm_sherbrooke. We examined that for the 1-qubit attack we need around 2300 to 2400 H-gates to provoke the erroneous state. For the 2-qubit attack we only needed around 140 to 160 gates. As the gate realization of 2-qubit gates is more complex and attractable to errors as for the 1-qubit gates this result seems to fit and reflect the problems in real QPUs.

4.2 Results of Transpilation Attack

We managed to implement an own transformation-pass for qiskit that as easiest demonstration just doubled the X-gates of a circuit. After that we integrated this transformation-pass into our PassManager used for transpilation. As result we got the tempered circuit with additional X-gate for qubit q1 as shown in Figure 11.

The execution of this circuit lead to the tempered and expected values “00000” and “001111” as shown in histogram in Figure 12.

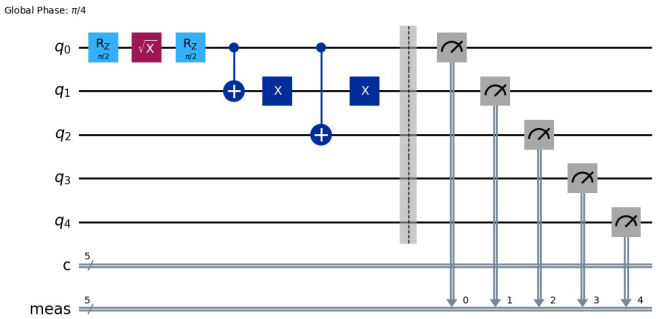

Figure 11: Results of transpilation attack; Tempered circuit layout after transpilation to HW gates is shown on the left. The additional inserted X-gate can be seen for qubit q1

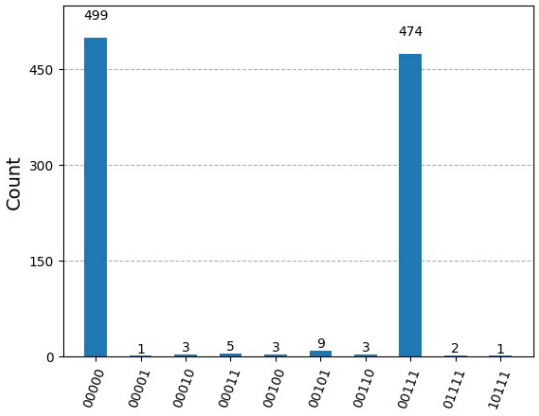

Figure 12: When tempered circuit is run on simulator, we get the expected histogram of results – almost equal probability of states "00000" and "00111" - shown on the right

When using qiskit it is quite easy to integrate an extra transformation-pass that manipulates the output of the circuit as one can directly use the underlying DAGCircuit from Qiskit Development Team (2025) with nodes and edges that represent operations/gates on the qubits of the circuit.

4.3 Results of Tampering Input Data

For the OnePixel attack we randomly choose one image of the test-set. Then we iteratively took one random pixel and turned it to a value corresponding to black colour if the target pixel had a lighter grey colour. If the target pixel was in a darker grey colour, we turned it to a value corresponding to white colour. We predicted the label of the tempered image with our model and looked if we successfully achieved a misprediction. We iteratively choose randomly another pixel if we didn't achieve a misprediction. We just inspected if the attack was successful. We stopped here and will have to quantify these results in an additional study in the future.

For the PGD attack we successfully managed mispredictions of 3 out of 4 test images, as shown in Figure 13. We used a QML-classifier consisting of 32 StronglyEntanglingLayers, see pennylane Development Team (2025), with 8 qubits integrated as layer in PyTorch. Together with a CrossEntropyLoss-function and the Adam-optimizer with a learning rate of 0.1.

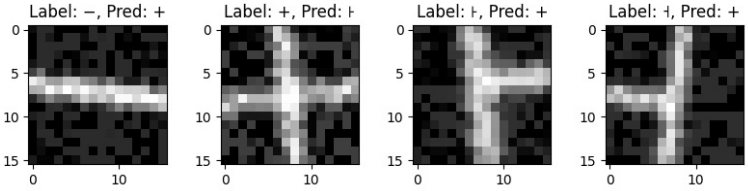

Figure 13: Images and resulting prediction after the PGD attack. We see that 3 out of 4 images get misclassified

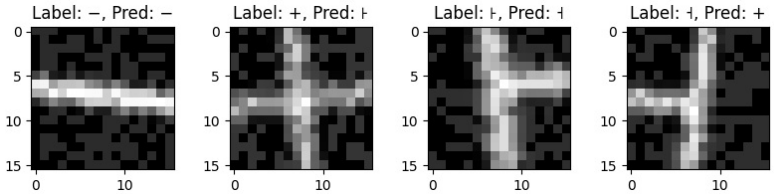

Figure 14: Images and resulting prediction after the FGSM attack. We see that 3 out of 4 images get misclassified

Figure 14 presents that we successfully managed mispredictions of 3 out of 4 test images with FGSM attack. We used the same QML model, same loss function and same optimizer for this experiment.

5. Countermeasures

As we have shown, Quantum Machine Learning (QML) introduces unique security challenges, as the quantum systems underpinning these models are vulnerable to attack vectors not present in classical environments.

Addressing these vulnerabilities is crucial for ensuring the robustness and trustworthiness of QML applications. This section outlines specific countermeasures for the attack vectors shown above.

Crosstalk occurs when concurrent quantum operations interfere with each other, leading to unintended state changes and errors in QML models. Effective countermeasures include:

- On-Premise Deployment: Running QML workloads on dedicated, on-premise quantum devices can mitigate risks by eliminating interference from other users and applications.
- Isolation Mechanisms: Implementing barriers or scheduling techniques to isolate QML algorithms from concurrent processes on shared quantum hardware can minimize crosstalk.
- Error Rate Improvements: Leveraging ongoing advancements by quantum hardware vendors, such as error rate reduction and improved qubit coherence, can further reduce susceptibility to crosstalk.

Transpilation, the process of converting high-level quantum circuits into executable forms optimized for specific hardware, poses risks if malicious or untrusted transpilation providers are used. Countermeasures include:

- Use of Trusted Transpilation Providers: Employ only well-vetted, secure providers to minimize the risk of introducing vulnerabilities during transpilation.
- Equivalence Checks: Validate the integrity of the transpiled circuit by
- *Inverse Execution: Apply the inverse of the original circuit to the transpiled version and verify that the output matches the initial state.*
- *Benchmarking: Use benchmarking tools like the MQT Bench from the Munich Quantum Toolkit (2025) to assess the fidelity and correctness of the transpiled circuit.*

Tampering attacks, such as adversarial perturbations during the training phase, can degrade QML model performance. Mitigation strategies include:

- Training with Perturbations: Regularly introduce controlled perturbations during training to make models more robust against adversarial inputs.
- Robust Optimization Techniques: Employ optimization algorithms that are resistant to minor perturbations in training data.

By implementing these countermeasures, organizations can enhance the security and reliability of QML systems. Addressing vulnerabilities specific to quantum environments ensures that QML can be adopted with confidence across critical applications.

6. Conclusion

First experiments - as proof-of-concepts - based on the knowledge gained in Reers et al. (2024) showed that QML is also exposed to cybersecurity assets regarding the CIA-triad. We found that it is quite easy to use crosstalk to temper output of quantum circuits and models in QML. Nevertheless, the countermeasures like putting boundaries between parallel execution of circuits or recalibration before executing new experiments that can be done are important for QPU providers to ensure that such attacks cannot happen on their cloud service platforms.

We have shown that it is easy – as this is intended by qiskit platform – to add additional transformation-passes to the transpiler. In future work, we will aim to gain more insight on how we can bypass this mechanism and introduce our transformation-pass for more realistic scenarios. We will also try to modify the implementation from inserting additional X-gates to a more realistic and intelligent modification also leading to mispredictions in QML models.

Based upon the OnePixel, PGD and FGSM attacks, we will summarize the theory behind and try to create new attack scenarios. We will quantify our results for the random OnePixel attack and will also consider getting the model more robust against these attacks by training with perturbed samples. And we will extend our experiments to be run on real QPUs to see what happens when our models face noisy results and which role current error mitigation techniques play with respect to robustness against these attacks.

References

Abbas, A., Sutter, D., Zoufal, C., Lucchi, A., Figalli, A., & Woerner, S. (2021). The power of quantum neural networks. *Nature Computational Science*, 403-409.

Arute, F., Arya, K., Babbush, R. et al. (2019). Quantum supremacy using a programmable superconducting processor. Nature 574, 505–510.

Kempka, C. D., & Schaad, A. P. (2022). Securing the ML Lifecycle. *Industry IoT Consortium.*

Kolter, Z., & Madry, A. (2025, 01 09). *Adversarial Robustness - Theory and Practice*. Retrieved from https://adversarial-ml-tutorial.org/adversarial_examples/

Munich Quantum Toolkit. (2025, 01, 21). TUM Chair for Design Automation. Retrieved from https://www.cda.cit.tum.de/research/quantum/mqt

pennylane Development Team. (2025, 01 17). *qml.StronglyEntanglingLayers*. Retrieved from https://docs.pennylane.ai/en/stable/code/api/pennylane.StronglyEntanglingLayers.html

Qiskit Development Team. (2023, 06 07). *Transpiler (qiskit.transpiler)*. Retrieved from Transpiler (qiskit.transpiler): https://qiskit.org/documentation/apidoc/transpiler.html

Qiskit Development Team. (2025, 01 17). *DAGCircuit*. Retrieved from https://docs.quantum.ibm.com/api/qiskit/qiskit.dagcircuit.DAGCircuit

Qiskit Development Team. (2025, 01 17). *Write a custom transpiler pass*. Retrieved from https://docs.quantum.ibm.com/guides/custom-transpiler-pass

Reers, V., & Maußner, M. (2024). Comparative Analysis of Vulnerabilities in Classical and Quantum Machine Learning. *INFORMATIK 2024*, 555-571.

Saki, A. A., Alam, M., & Ghosh, S. (2020, 08). Analysis of crosstalk in NISQ devices and security implications in multi-programming regime. 25-30.

Su, J., Vargas, D. V., & Sakurai, K. (2019). One Pixel Attack for Fooling Deep Neural Networks. *arXiv:1710.08864*.

Weder, B., Barzen, J., Leymann, F., Salm, M., & Vietz, D. (2020). The Quantum software lifecycle. *APEQS 2020: Proceedings of the 1st ACM SIGSOFT International Workshop on Architectures and Paradigms for Engineering Quantum Software*, 2-9.

Weigold, M., Barzen, J., Leymann, F., & Salm, M. (2020). Data encoding patterns for quantum computing. *PLoP '20: Proceedings of the 27th Conference on Pattern Languages of Programs*, 1-11.

Wendlinger, M., & Tscharke, K. (2025, 01 09). *Adversarial attacks and robustness for quantum machine learning*. Retrieved from https://pennylane.ai/qml/demos/tutorial_adversarial_attacks_QML

Wikipedia. (2025, 01 14). *Quantum logic gate*. Retrieved from https://en.wikipedia.org/wiki/Quantum_logic_gate