QUANTINUUM

Fachhochschule Nordwestschweiz

3 GOOD HEALTH AND WELL-BEING

QCNNs for Small-Data Medical Diagnosis

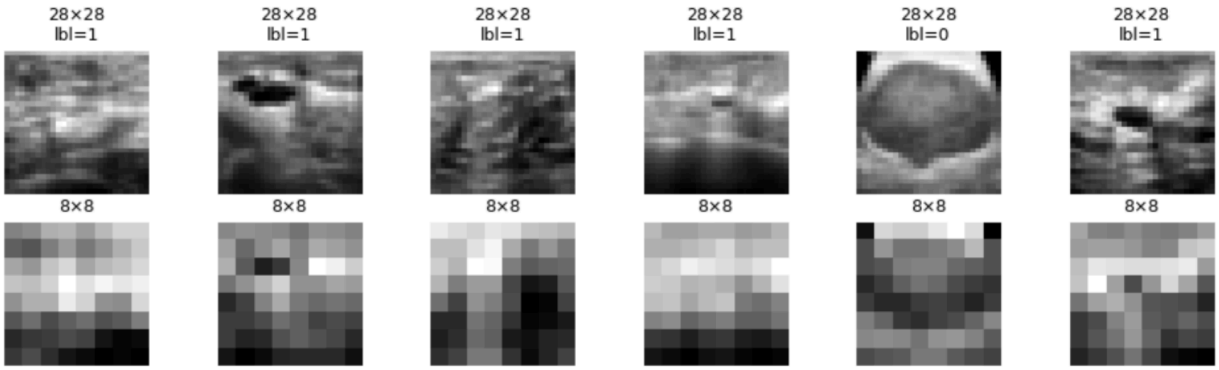
1. Context and Relevance

Image classification is a cornerstone task in medical AI, but it faces unique obstacles when applied to rare disease datasets. Unlike applications where millions of labeled images are available, clinical datasets are often small, fragmented, and collected under varying conditions. This scarcity makes it difficult to train models that can distinguish subtle patterns reliably, while also generalizing across different institutions, imaging devices, and patient populations. The challenge is not only the lack of data volume but also the variability and imbalance that make classification for rare diseases particularly demanding.

Classical approaches to image classification, such as convolutional neural networks, have achieved state-of-the-art results in many domains, but they typically require large amounts of data and computational resources to perform well. Techniques like data augmentation, transfer learning, and regularization can help alleviate data scarcity, yet their effectiveness is limited when facing highly imbalanced or rare disease datasets. Quantum convolutional neural networks (QCNNs) offer a possible alternative by encoding and processing image features more efficiently, potentially enabling better generalization from fewer samples.

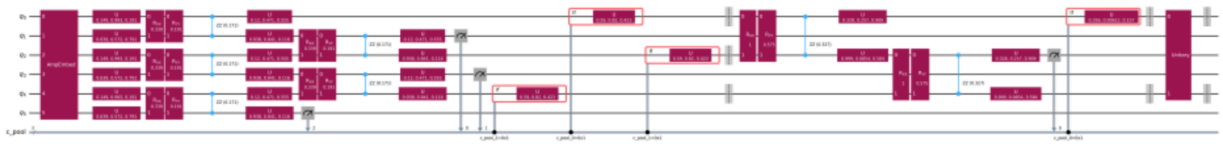
2. Challenge Overview

In this hackathon challenge, participants are tasked with investigating the question: How does the way we encode images affect quantum resource cost and generalization potential? No model training is needed, just smart circuit design. To do so, we use the publicly available BreastMNIST dataset as a proxy (find the link to the dataset in the references below). It's a binary classification task (malignant vs. benign) on grayscale images (sample shown in figure below), resized to 8× 8, 16 × 16, and 28 × 28 which is a perfect testbed for analyzing resource scaling under different quantum encodings.



Classical convolutional neural networks (CNNs) famously identify cats from dogs by scanning images pixel by pixel, layer by layer. Imagine doing something similar, but with quantum states instead of pixels (welcome to quantum convolutional neural networks, or QCNNs). Recent literature and experiments have shown QCNNs outperform classical counterparts in low-data regimes [1].

However, quantum models face their own challenges. The number of qubits is not always the main hurdle in today's noisy intermediate-scale quantum (NISQ) era. Often, it is the depth and complexity of quantum circuits that limit performance and feasibility [4]. In this hackathon challenge, we keep the QCNN architecture fixed and pre-trained, allowing you to directly experiment and witness predictions quickly. Your core task is to systematically explore how different ways of encoding images into quantum states affect the scalability and complexity of the circuit (see the figure below for the fixed QCNN architecture implemented in Qiskit for processing 8x8 grayscale images).



Participants are encouraged to work through three interactive stages:

- **Intuitive Understanding:** First, build intuition by reasoning how different encoding strategies influence key scalability metrics such as the number of trainable parameters T, parameter reuse factor M, circuit depth L, and circuit width, while keeping the QCNN architecture fixed. Sketch circuits to visualize these impacts clearly.
- **Hands-On Circuit Construction and Transpilation:** Next, implement various image encoding strategies using Qiskit, create corresponding quantum circuits, and empirically measure how circuit resources (qubits q, depth, gate count, two-qubit

gates) scale with different input resolutions. Compare your empirical findings with initial theoretical intuitions, and extrapolate these results to larger-scale realistic applications.
- **Creative Architecture Tweaking:** Finally, explore how small variations to the QCNN architecture (such as adjusting the circuit depth, width, or pooling layers (to be discussed with mentors), interact with your chosen encoding strategies. Measure how these tweaks affect resource metrics to deepen your understanding of QCNN scalability.

This structured yet playful approach allows you to explore key quantum computing ideas without the heavy computational overhead of training quantum models from scratch. By isolating encoding strategies and focusing on depth and gate complexity, you contribute directly to ongoing research efforts to make quantum circuits practical and efficient under realistic, near-term hardware constraints [4].

3. Detailed Problem Description

a. Mathematical Problem / Model Description

Quantum Convolutional Neural Networks (QCNNs) [1] have emerged as promising architectures for achieving strong generalization performance from few training samples. Their success stems from architectural features such as parameter sharing, circuit sparsity, and hierarchical pooling, which align well with theoretical guarantees in quantum learning theory.

Recent work by Caro et al. [2] formalizes these intuitions by deriving generalization bounds for variational quantum models. These bounds reveal that the generalization error depends not only on the number of trainable parameters, but also on their structural reuse and the degree of optimization during training.

Let the generalization error of a variational quantum model with parameters α be defined as:

$$\mathrm{gen}(\alpha) := R(\alpha) - \hat{R}_S(\alpha),$$

where R(α) is the expected loss over the underlying data distribution, $\hat{R}_S(\alpha)$ is the empirical loss over a training set S of size N, and α represents the vector of tunable parameters in the quantum circuit.

Theoretical bounds presented in [2] show that gen(α) can be controlled by three structural quantities of the circuit:
(i) T: the number of independently parameterized gates,
(ii) M: the maximum number of times any parameter is reused (e.g., via weight sharing),

(iii) K: the number of parameters that are significantly updated during training.

See Appendix for detailed explanation on the generalization bounds.

A crucial but often overlooked aspect in quantum model design is the data encoding step: how classical inputs $x \in R^n$ are mapped to quantum states. This map is denoted:

$$f : \mathbb{R}^n \to \mathcal{H}_{2^q}, \quad x \mapsto f(x) = |\psi_x\rangle,$$

where q is the number of qubits and H_{2q} is the Hilbert space of the quantum register. Different encoding methods, such as amplitude encoding, angle encoding, and structured image encodings like FRQI or NEQR [3], induce different circuit complexities for preparing the state $|\psi_x\rangle$.
This affects generalization in two ways:

- **Circuit size and depth:** Encodings like amplitude embedding are qubit-efficient (e.g., q = log2 n), but often require deep circuits with exponential gate count for arbitrary state preparation. This increases the effective T in the generalization bound.
- **Qubit count vs. depth trade-off:** Angle encoding uses more qubits (q = n), but prepares the state with shallow, local rotations. This can reduce T and L (depth), potentially improving generalization and hardware feasibility.

b. Proposed Quantum Approaches

For this challenge the quantum approach is limited to the (already trained) QCNN you can start with.

c. Getting Started

Participants will begin by running a pre-trained QCNN model on a small subset of the BreastMNIST dataset to verify a working setup. From there, the challenge unfolds through a structured sequence of tasks that combine hands-on experimentation with theoretical insights. The goal is to explore how image encoding impacts the scalability and generalization behavior of quantum neural networks.

1. **Run the Pretrained QCNN on BreastMNIST:** Load the provided QCNN circuit and run inference on a few BreastMNIST test samples. Verify that the predictions are stable across runs and understand how the QCNN outputs label probabilities.
   Deliverables: screenshots or logs confirming inference, brief explanation of what each step in the circuit does.

2. **Sketch the QCNN Architecture with an arbitrary encoding:** Recreate a sketch of the provided QCNN circuit, highlighting the convolution and pooling layers. How does

the size of the encoded input depend on the chosen image resolution? How does the number of layers depend on the size of the encoded input?
Deliverables: Draw 3 circuits with different chosen input resolutions, i.e. number of qubits (*"Pen and paper will solve almost anything." - Nicholas Bate ;)*)

3. **Implement and Compare Image Encodings:**
   Please note: from this step onwards, participants are free to explore image datasets of their choice. In the first point you used BreastMNIST since the provided model was trained for that dataset.
   Select at least two encoding methods (e.g., amplitude, angle, FRQI, NEQR) and use them to map BreastMNIST images into quantum circuits. Begin with 8 × 8 images, resizing or preprocessing as needed.
   Deliverables: encoding functions, preprocessing steps, and a short note explaining the trade-offs of each encoding.

4. **Reconstruct Images and Evaluate Fidelity:** For each encoding, reconstruct the input images from measurement outcomes using different shot counts. Evaluate reconstruction quality using fidelity metrics such as structural similarity index measure (SSIM) and mean squared error (MSE).
   Deliverables: Plot SSIM and MSE as a function of number of shots. Identify the smallest number of shots that yields visually or numerically acceptable reconstructions.

5. **Investigate Circuit Scaling for Two Encodings:** Pick two encodings from Task 3 and scale the input size from 8 × 8 to larger sizes such as 16 × 16, and beyond, up to the maximum that successfully transpiles on your setup. For each case, build the full pipeline (encoding, QCNN, measurement) and measure circuit resource usage.
   Deliverables: Record the number of qubits, total gates, number of two-qubit gates, and total circuit depth. Create log–log plots and estimate scaling trends.

6. **Estimate Generalization-Relevant Metrics (T and M):** For the same two encodings and input sizes:

(a) Compute T: the number of unique trainable parameters in the QCNN.
(b) Compute M: the maximum number of times a parameter is reused, due to convolutional or pooling operations.

7. **Explore 2-qubit and k-qubit Dense Layers:** Dive into the internals of the QCNN by focusing on the use of SU(k) unitary blocks:
   (a) Implement a function to map 15 real parameters to a valid SU(4) matrix.
   (b) Apply this unitary as a two-qubit block using either Qiskit's UnitaryGate.
   (c) How many independent parameters are needed for SU(k)? Are they real or complex?
   (d) Generalize to SU(k) for more qubits. What is k for an n-qubit gate? Implement the corresponding circuit.

4. Expected Deliverables

- **Working Prototype:** A functional model that takes data as input and produces a solution to the posed challenge. We recommend a step-by-step approach, beginning with a proof-of-concept for a very simple toy model before expanding to a full model, e.g. tackling larger data sets or problem instances.
- **Technical Explanation:** Participants must be able to explain the quantum algorithms used, the data model, and the rationale for choosing the specific approach. The focus should be on the clear and deliberate use of methodology and the benchmarking process, rather than solely on high-performing results. Participants should benchmark their quantum model against standard classical solutions. If more than one model was developed during the hackathon the team may also benchmark against those.
- **Classical Bottleneck Analysis and Justification for Quantum Computing Approach:** An analysis of standard approaches and their bottlenecks. This is the basis for arguing why a quantum computing approach may be beneficial for the problem at hand.
- **SDG Impact Assessment:** An assessment of the impact of your solution on the main SDG tackled by this challenge. Your arguments should be backed by references to publications and publicly available data.
  Also: What other SDGs may be impacted by your solution (both to the positive or to the negative)? List them and argue for why your solution is relevant also to those.
- **Business Case:** A concise plan outlining the commercial viability, target users, and market strategy. Think about how your solution could be commercialized, how you would approach commercialization, what would the product be, who would be the customers, how you would make money with it.
  Also: In what other fields / problems could your solution be useful?

- **Pitches & Technical Deep Dive:** At the end of the hackathon you will have the chance to present your solution in a final presentation as a 3 min pitch with slides. Please explain the solution's potential and its future roadmap, including an honest discussion of the challenges that need to be addressed before it can be a fully realized product, all based on the above deliverables. The pitch will be followed by a 2 min Q&A with the jury for the respective challenge, which consists of 50% technical experts and 50% business experts.
  In addition, you will have the chance to present your solution in a 10 min technical deep dive to the technical experts in the jury. For this, no slides are required, but a prepared code review is envisioned.
  The overall 10 best teams across all challenges get to show the same 3 min pitch in front of all participants and all jury members again, this time without Q&A.

Importantly: A deeper investigation into one quantum computing approach is preferred over a superficial overview of many. The goal is not to find a single "best" model, but to demonstrate a clear understanding of the chosen approach. The solution should focus on demonstrating how the quantum approach can be applied to the problem, specifically addressing the challenge specified above. The goal is to articulate a compelling case for a future computational speed-up or greater accuracy, rather than proving an immediate advantage.

Target audience: The project should be compelling to technical professionals who are interested in exploring how quantum computing can be applied to their domain.

5. Resources

[Context] SDG 3: https://sdgs.un.org/goals/goal3

[Intro, SDK] Qiskit's Quantum Learning Module:
https://quantum.cloud.ibm.com/learning/en

[1] Quantum convolutional neural networks: https://arxiv.org/abs/1810.03787

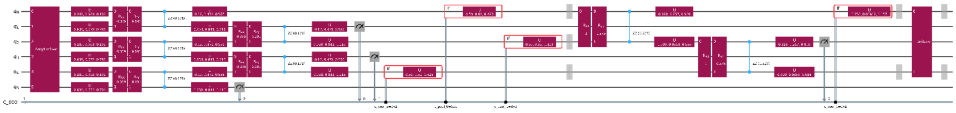
[2] Generalization in quantum machine learning from few training data:
https://arxiv.org/abs/2111.05292

[3] Representation of Classical Data on Quantum Computers: https://arxiv.org/abs/2410.00742

[4] Quantum-parallel vectorized data encodings and computations on trapped-ions and transmons QPUs: https://arxiv.org/abs/2301.07841

Reference material provided

[Resource] Reference QCNN Architecture:



A code snippet for building the QCNN architecture shown above will be provided on site for participants to get started.

[Resource] Encoding Strategies:

Method	**Definition**	**Qubits Required (Example: 64x64 Image)**
FRQI(Flexible Representation of Quantum Images)	Encodes pixel positions using $\log_2(N^2)$ qubits; pixel intensity is encoded using amplitude (1 additional qubit).	$\log_2(64\times64) = 12 + 1 = 13$
NEQR(Novel Enhanced Quantum Representation)	Same position encoding as FRQI; pixel intensity stored using 8 qubits (binary format for 8-bit grayscale)	$\log_2(64\times64) + 8 = 12 + 8 = 20$
GNEQR(Generalized NEQR)	Extends NEQR to multi-channel images (e.g., RGB); for grayscale,	Same as NEQR = 20

	same as NEQR	
QPIE(Quantum Probability Image Encoding)	Encodes a $2^n \times 2^n$ image into 2n qubits by normalizing pixel intensities and mapping them to amplitude probabilities. Pixel positions are represented by computational basis $\|k\rangle$, eliminating the need for additional qubits for pixel values	$\log_2(64) + \log_2(64) = 6 + 6 = 12$

* This is not an exhaustive list and participants are encouraged to explore beyond.

[Resource] Starter Code Snippets: An example code snippet for amplitude encoding will be provided on site.

[Data] Open-Source BreastMNIST Dataset: https://medmnist.com/

6. Appendix

Theorem 1 (Basic Bound): If the quantum model has T independent parameterized k-local channels (no reuse), then with high probability:

$$\mathrm{gen}(\alpha) \in \mathcal{O}\left(\sqrt{\frac{T \log T}{N}}\right)$$

This is the worst-case baseline bound, and emphasizes the importance of minimizing the number of free parameters T to reduce overfitting in low-data regimes.

Theorem 2 (Gate-Sharing Bound): If each parameterized gate is reused up to M times (as in QCNNs), then:

$$\mathrm{gen}(\alpha) \in \mathcal{O}\left(\sqrt{\frac{T \log(MT)}{N}}\right)$$

This shows that gate-sharing introduces only a logarithmic penalty in M, encouraging circuit designs with parameter reuse such as convolutional layers.

Theorem 3 (Optimization-Aware Bound): Let the displacements of the parameters during training be denoted $\Delta_1 \geq \Delta_2 \geq \cdots \geq \Delta_T$. Then:

$$\mathrm{gen}(\alpha) \in \mathcal{O}\left(\min_{0 \leq K \leq T}\left\{\sqrt{\frac{K \log(MT)}{N}} + M \sum_{k=K+1}^{T} \Delta_k\right\}\right)$$

This bound incorporates training dynamics. If only $K \ll T$ parameters undergo large updates (i.e., optimization is sparse), the generalization behavior improves significantly.