R2P
CAPSTONE PROJECT

Team 5

Quantum Neural Networks

Bilbao • May 2025

TEAM MEMBERS

Iñigo Vilaseco

Ignacio Fernández

Pedro Álvarez

Diego Mallada Conte

Adrián Gustavo del Pozo Martín

Arturo Juárez

AGENDA

1. PROBLEM STATEMENT
2. QUANTUM FEATURE MAP
3. PARAMETERIZED QUANTUM CIRCUITS
4. TRAINING
5. RESULTS
6. OTHER PROBLEM VARIANTS
7. BIBLIOGRAPHY

1. PROBLEM STATEMENT

Problem Statement: Quantum Image Classification

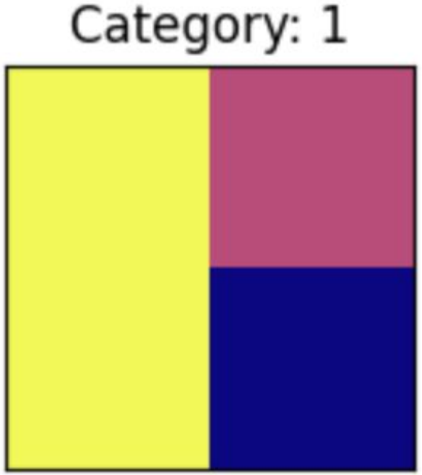
What are we doing?

We train a variational quantum circuit to distinguish three types of 2×2 “images” (lines: horizontal, vertical, diagonal).

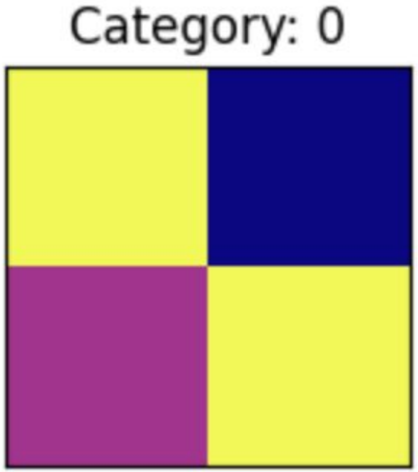
How do we encode the images?

- **Vector representation (length 4):** each of the 4 entries corresponds to one pixel/qubit.
- **Line pixels:** assigned angle value $\pi/2$
- **Background pixels:** uniform random noise in $[0, 1)$

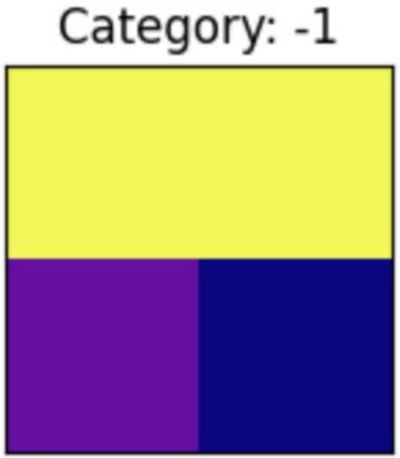
The Dataset



[π/2, 0.15, π/2, 0.54]



[0.44, π/2, π/2, 0.88]



[π/2, π/2, 0.23, 0.67]

2. QUANTUM FEATURE MAP

Quantum Feature Map: From Classical Data to Quantum States

Why a Feature Map?

- **Purpose:** Embed a real-valued vector $\mathbf{x}$ into a quantum state $|\psi(\mathbf{x})\rangle$

Our Chosen Feature Map: ZFeature Map

- **Input:** Vector $\mathbf{x} = [x_0, x_1, x_2, x_3]$
- **Construction:**
  1. Start all qubits in $|0\rangle$
  2. Apply Hadamard H on each qubit to create superposition.
  3. For each qubit i, apply $R_Z(x_i)$
  4. Apply CZ gates between every pair of qubits (full entanglement).

3. PARAMETERIZED QUANTUM CIRCUITS

Parameterized Quantum Circuits: Theory Overview

A Parameterized Quantum Circuit (PQC) is a sequence of quantum gates that depend on continuous parameters $\boldsymbol{\theta}$

It is usually expressed as:

$$U(\boldsymbol{\theta}) = \prod_k U_k(\theta_k)$$

Where each $U_k(\theta_k)$ is a parameterized single-qubit rotation or controlled gate.

Training: the parameters $\boldsymbol{\theta}$ are optimized by minimizing a cost function $C(\boldsymbol{\theta})$.

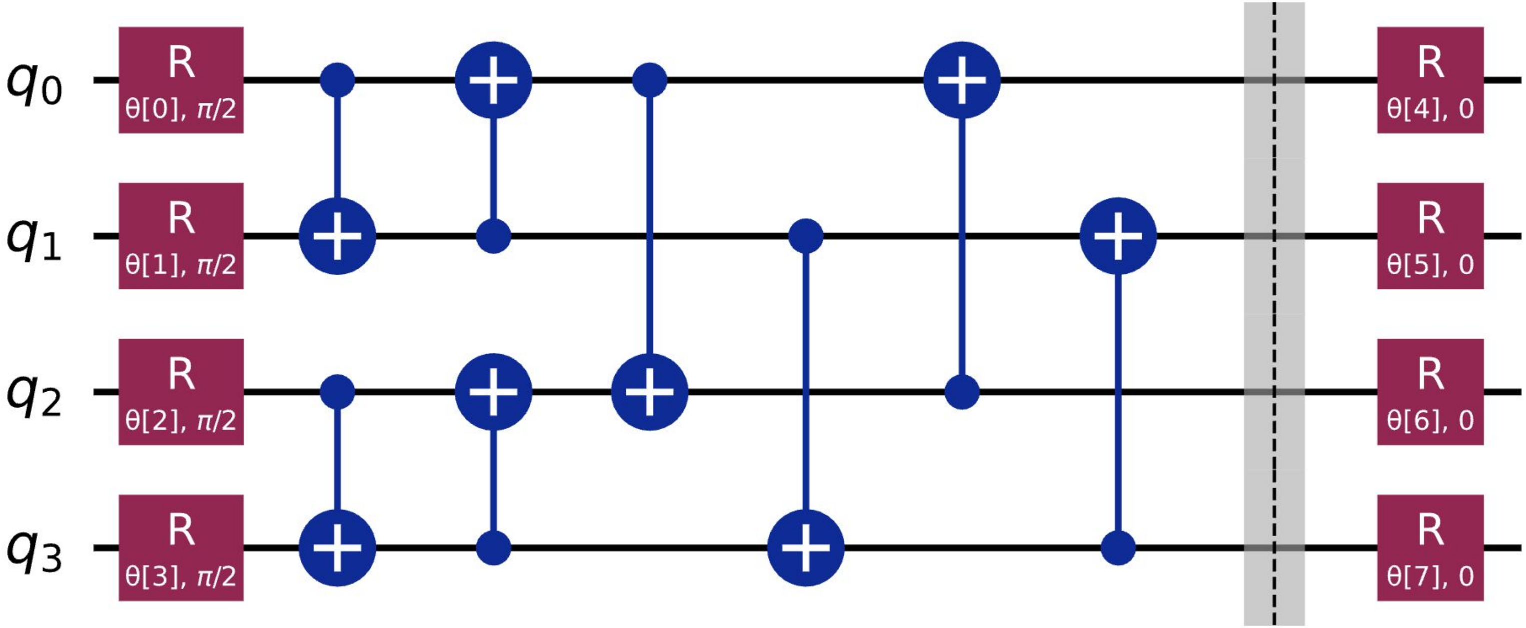
Our Chosen Ansatz Architecture

- **Parameter vector:**

  $\boldsymbol{\theta} = [\theta_0, \ldots, \theta_7]$ (two angles per qubit, here 2 x 4 = 8).

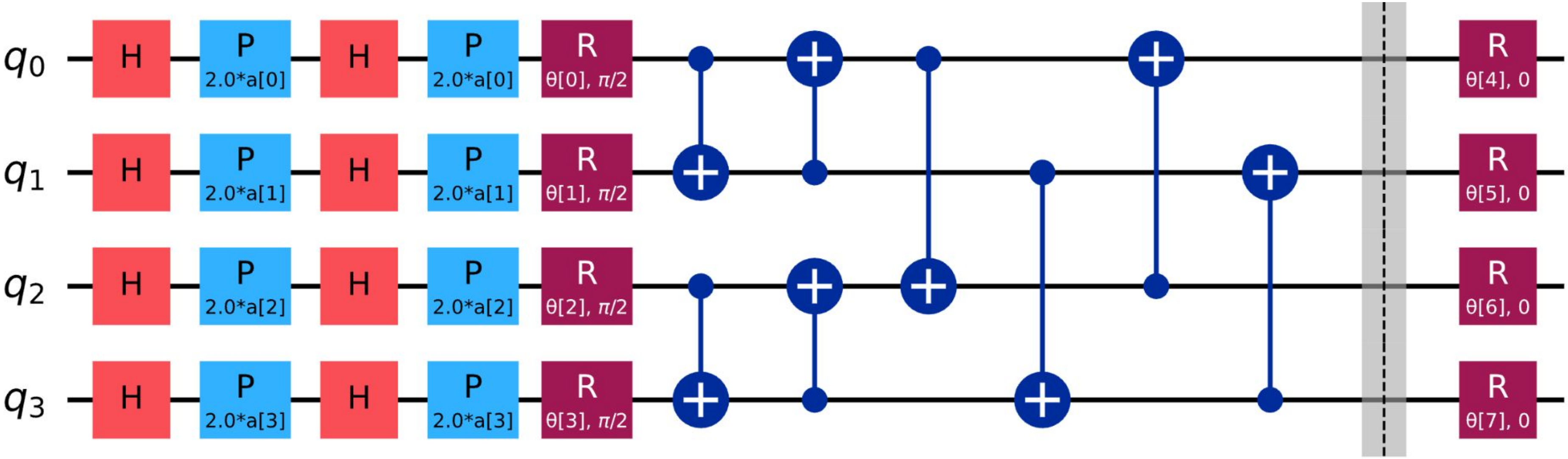
- **Layer 1:**
  - Single-qubit rotations $R_Y(\theta_i)$ on each qubit i
- **Entanglement:**
  - **CNOTs connecting neighboring qubits horizontally and vertically** (in the 2×2 grid).
- **Layer 2:**
  - Single-qubit rotations $R_X(\theta_{4+i})$ on each qubit.

Our Chosen Ansatz Architecture



Merging both ansatz and feature map

We compose the feature map and ansatz into a single circuit, then define the measurement operator $O = Z^{\otimes n}$ (using SparsePauliOp.from_list([("Z"*n,1)])) whose expectation value $\langle O \rangle$ serves as the classifier output.



4. TRAINING

Quantum Forward Pass

What is the Forward Pass?

- **Definition:** The computation that maps input data and circuit parameters to measurement outcomes.
- **Purpose:** Produces predictions $\hat{\boldsymbol{y}}$ by executing the quantum circuit and measuring the chosen observable.

$$(\mathbf{x}, \boldsymbol{\theta}) \to \langle Z^{\otimes n} \rangle$$

- **Analogy:** Like the “forward propagation” in a classical neural net, but replacing matrix multiplications with quantum operations.

Loss functions

The **loss function** $C(\boldsymbol{\theta})$ quantifies the discrepancy between the measured expectation values and the true labels.

$$C(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N}\left(\langle Z^{\otimes n}\rangle_i - y_i\right)^2$$

where:

$$\langle Z^{\otimes n}\rangle_i = \langle \psi(\mathbf{x}_i;\boldsymbol{\theta}) \,|\, Z^{\otimes n} \,|\, \psi(\mathbf{x}_i;\boldsymbol{\theta})\rangle$$

$$y_i \in \{-1, 0, +1\}$$

.

It drives the classical optimizer to update $\boldsymbol{\theta}$ so that the quantum circuit's outputs match the target labels.

Training Process: Mini-Batch Optimization

1. **Initialization**

$$\boldsymbol{\theta}^{(0)} \sim \mathcal{U}(0, 2\pi)^{2n}$$

   Random two-angle-per-qubit vector for the ansatz.

2. **Batch Loop** (for epoch $e = 0, \ldots, E - 1$)
   - Partition $\{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$ into batches of size b.
   - For each batch $\mathcal{B}$:
     - **Define cost** $C_{\mathcal{B}}(\boldsymbol{\theta}) = \dfrac{1}{b} \sum_{i \in \mathcal{B}} \left(\langle Z^{\otimes n} \rangle_i(\boldsymbol{\theta}) - y_i \right)^2$
     - **Update** $\boldsymbol{\theta} = \underset{\theta}{\operatorname{argmin}}\ C_{\mathcal{B}}(\boldsymbol{\theta})$

       using the derivative-free COBYLA optimizer (maxiter = 100)

Training Process: Mini-Batch Optimization

3. **Output:**
    - Trained parameters $\boldsymbol{\theta}^*$
    - Recorded loss history $\{C_{\mathcal{B}}(\boldsymbol{\theta})\}$

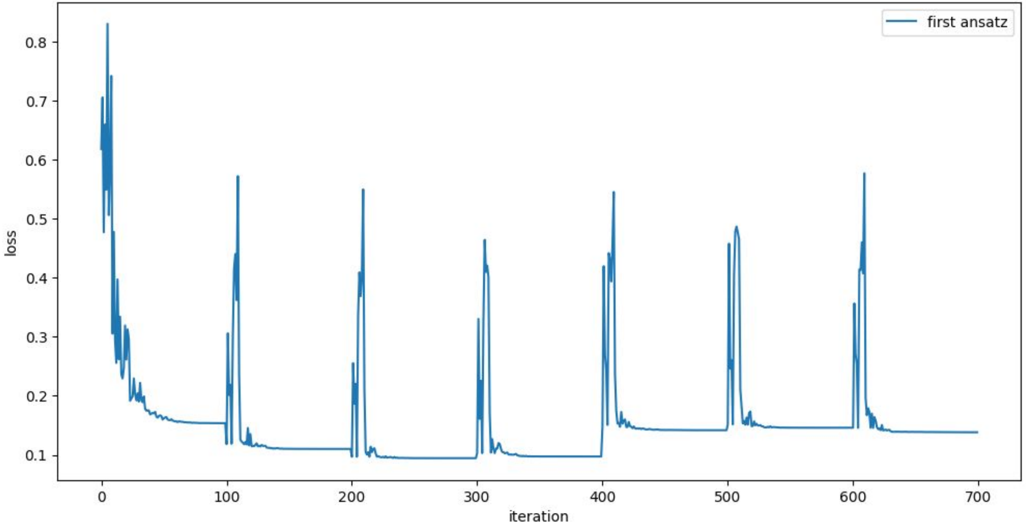
5. RESULTS

Results & Performance

Train accuracy: 84.14%

Test accuracy: 82.67%

Modest train–test gap (~1.5 pp) indicates good generalization.

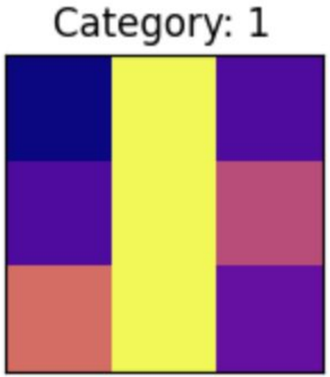


- **Spikes at ~0, 100, 200, …:** the very first evaluation in each new mini-batch—random weights on fresh data → high initial loss.
- **Rapid decay after each spike:** COBYLA quickly lowers the MSE within that batch.

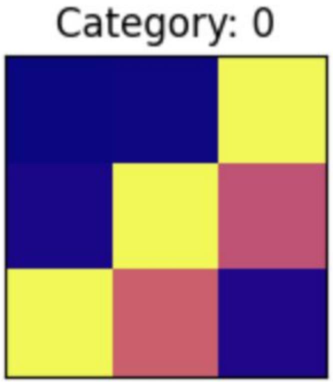
6. Increasing the data size: 3x3 images

The Dataset

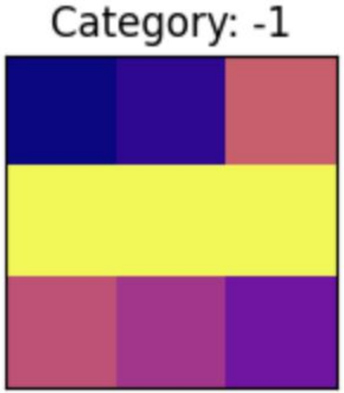
- **Input dimension:** n = 9 qubits (3×3 image)
- **Line length:** 3 pixels



$$\begin{bmatrix} n_{00} & \frac{\pi}{2} & n_{02} \\ n_{10} & \frac{\pi}{2} & n_{12} \\ n_{20} & \frac{\pi}{2} & n_{22} \end{bmatrix}$$

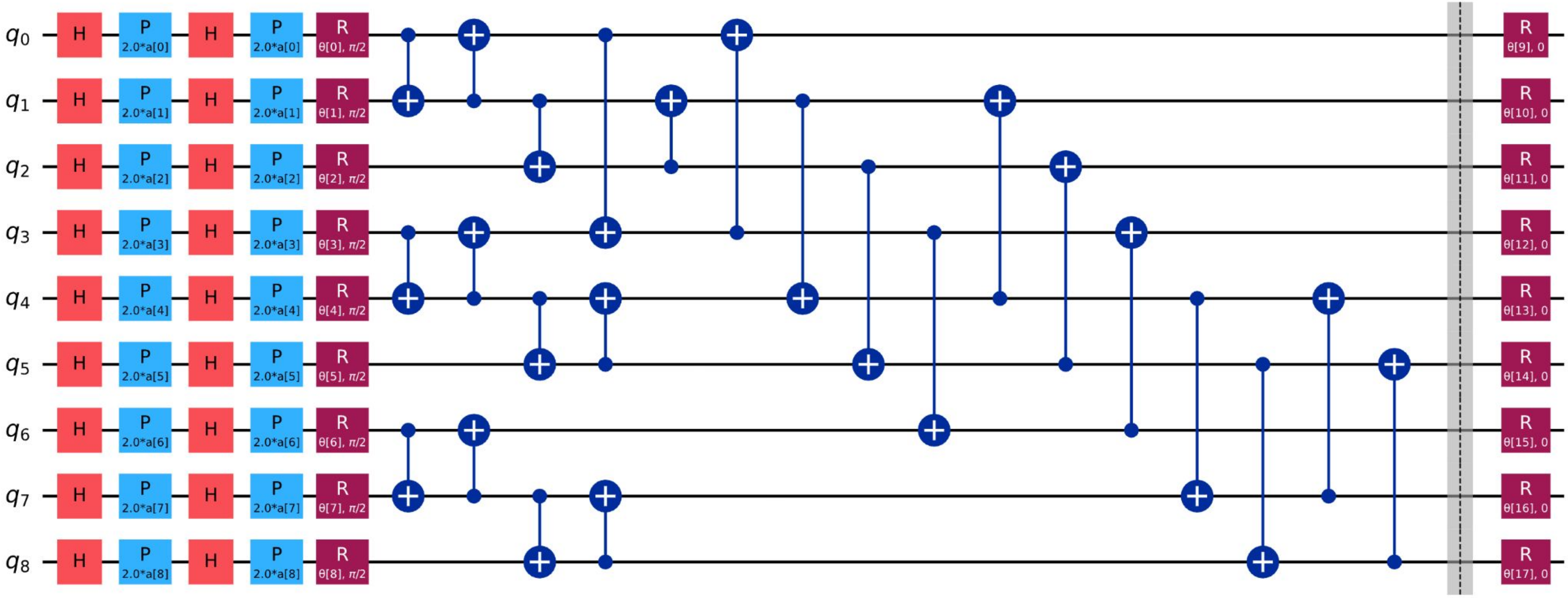


$$\begin{bmatrix} n_{00} & n_{01} & \frac{\pi}{2} \\ n_{10} & \frac{\pi}{2} & n_{12} \\ \frac{\pi}{2} & n_{21} & n_{22} \end{bmatrix}$$



$$\begin{bmatrix} n_{00} & n_{01} & n_{02} \\ \frac{\pi}{2} & \frac{\pi}{2} & \frac{\pi}{2} \\ n_{20} & n_{21} & n_{22} \end{bmatrix}$$

Merging both ansatz and feature map



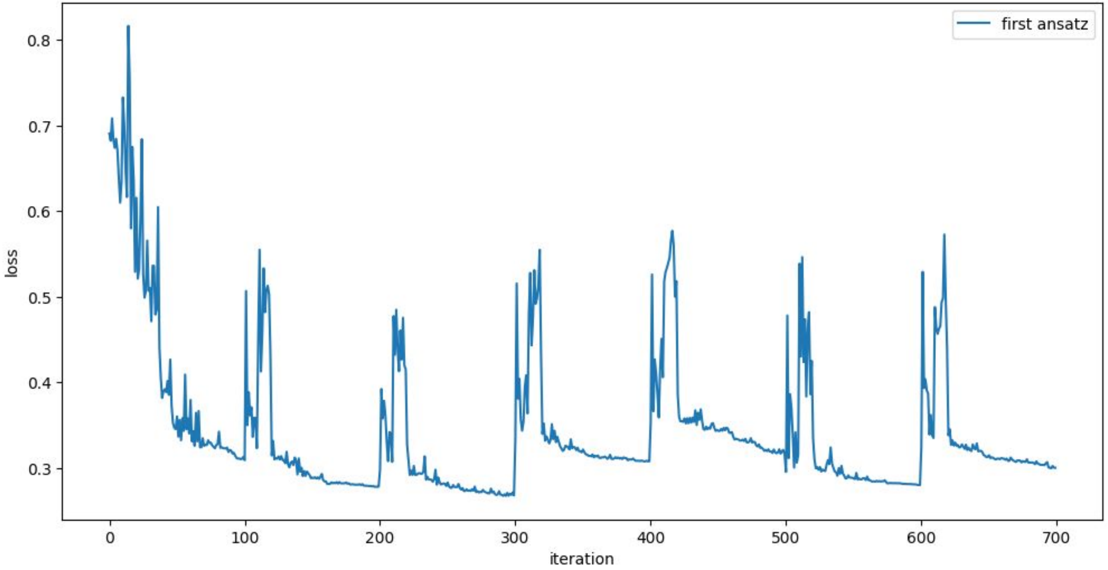
Results & Performance

Train accuracy: 72.57 %

Test accuracy: 73.33%

Train–test gap (~ -0.76 pp) indicates no overfitting.

Lower overall accuracy than the 2×2 case (~82.7% test)

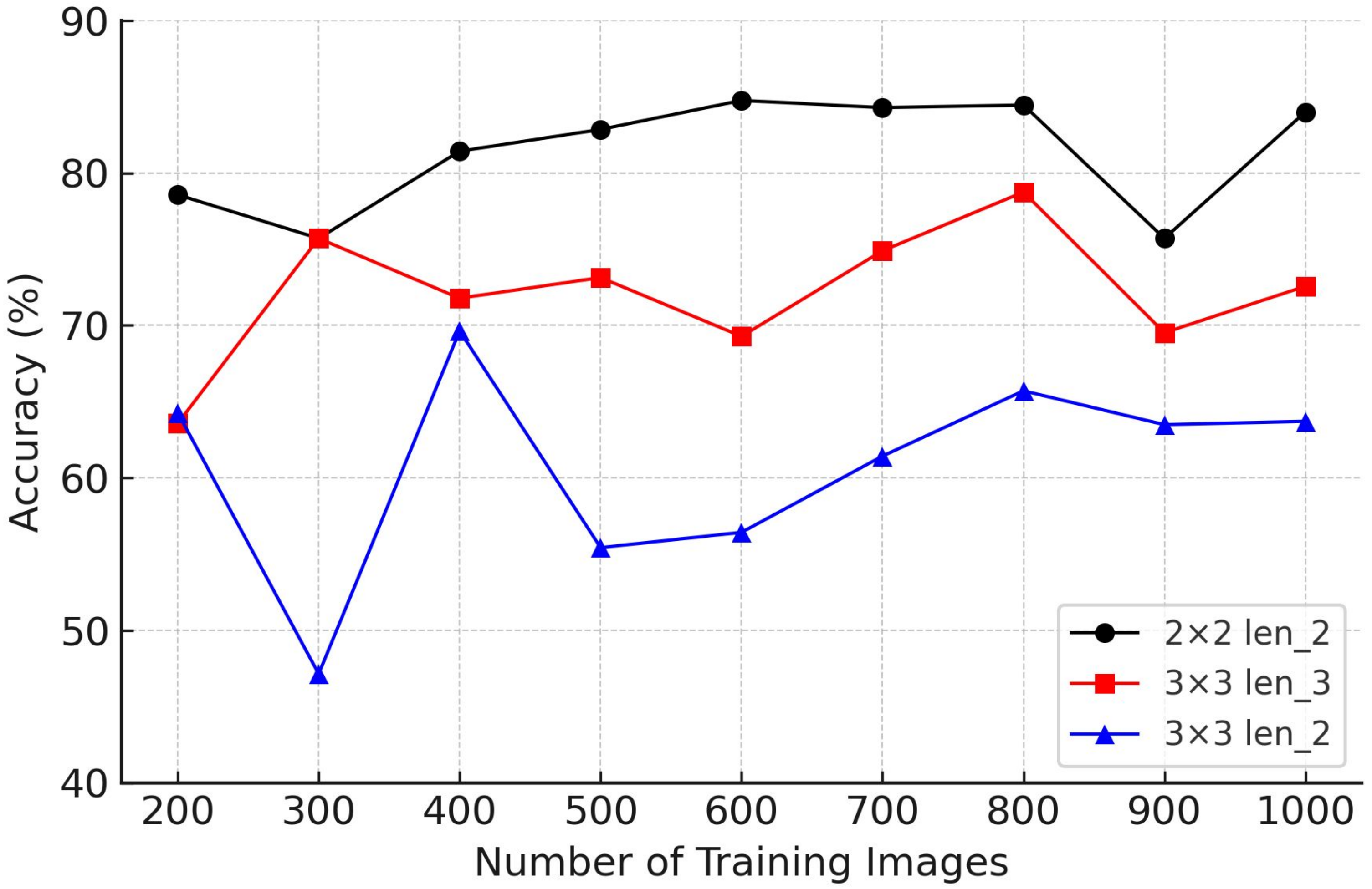


- **Spikes at ~0, 100, 200, …:** the very first evaluation in each new mini-batch—random weights on fresh data → high initial loss.
- **3×3 variant plateau** around **0.30 – 0.32**
- **2×2 baseline plateau** was around **0.14**

7. Testing the program

Prediction Accuracy

Training Images	2x2 images (len 2)	3x3 images (len 3)	3x3 images (len 2)
200	78%	64%	63%
300	75%	47%	75%
400	81%	69%	71%
500	82%	55%	73%
600	84%	56%	69%
700	84%	61%	74%
800	84%	65%	78%
900	75%	63%	69%
1000	84%	63%	72%



90
80
70
60
50
40
Accuracy (%)
200
300
400
500
600
700
800
900
1000
Number of Training Images
2×2 len_2
3×3 len_3
3×3 len_2

8. BIBLIOGRAPHY

Bibliography

- A review of Quantum Neural Networks: Methods, Models, Dilemma
  arXiv:2109.01840v1
- Training Quantum Embedding Kernels on Near-Term Quantum Computers
  arXiv:2105.02276
- Variational Quantum Classifier, Elies M. Gil Fuster & J.I. Latorre
- Quantum neural networks and variational circuits course (link)

[GitHub Repository]