Quanvolutional Neural Networks: Powering Image Recognition with Quantum Circuits

Quantum Machine Learning Exam Project

Jay Senoner

July 18, 2025

UNIVERSITÀ DEGLI STUDI FIRENZE
Da un secolo, oltre.

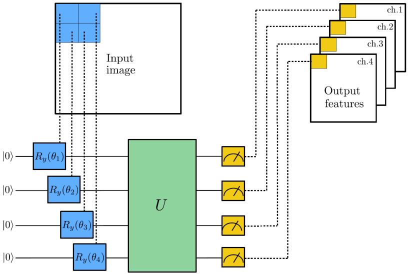
Introduction

Quanvolutional Neural Networks

Convolutional Neural Networks (CNNs) represent a core model architecture in computer vision, using learned filters to extract features from images. The paper by Henderson et al. asks a compelling question:

Central Idea

Can we replace a classical filter with a **quantum circuit** to generate more powerful, complex features that are difficult to replicate classically?



Henderson et al. Quanvolutional neural networks: Powering image Recognition with Quantum Circuits

Project Goals

Introduction

Our goal was not just to replicate the paper, but to perform a **rigorous analysis** to understand some of the practical trade-offs of the QNN approach.

- Does the QNN outperform a carefully designed classical baseline (CNN) on clean, full datasets?
- Is the QNN more **data-efficient**? Can it learn effectively from limited samples?
- Is the QNN more **robust** to perturbations of input data?
- Are the feature maps produced by quanvolutional layers actually useful without a deep neural network backend?

The Quanvolutional Layer

Implementation Details

We implemented the quantum feature extractor in PennyLane. It processes a 2×2 image patch in three stages:

1. **Encoding:** Pixel values $\phi \in [0, 1]$ are encoded into qubit states using angle encoding: $R_Y(\pi \cdot \phi)$.
2. **Transformation:** A fixed, non-trainable random quantum circuit is applied. The parameters are generated once with a fixed seed to act as a consistent filter.
3. **Measurement:** The expectation value $\langle \sigma_z \rangle$ of each qubit is measured to produce a classical feature vector.

Circuit Generation

Quanvolutional Neural Networks

```
dev = qml.device("lightning.qubit", wires=n_qubits)
rand_params = np.random.uniform(high=2 * np.pi, size
     =(n_layers, n_qubits))

@qml.qnode(dev)
def quanv_circuit(phi):
    # 1. Encoding
    for j in range(4):
        qml.RY(np.pi * phi[j], wires=j)
    # 2. Transformation (fixed params)
    RandomLayers(rand_params, wires=range(4))
    # 3. Measurement
    return [qml.expval(qml.PauliZ(j)) for j in range
         (4)]
```

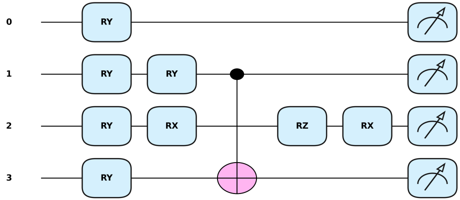


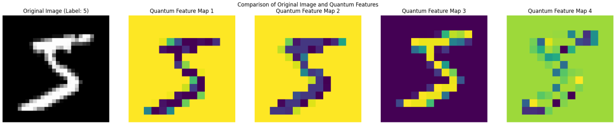
Figure: A random 1-layer quantum circuit

Patch Processing

Quanvolutional Neural Networks

```
def quanv_layer(image):
    out = torch.zeros((14, 14, 4))
    for j in range(0, 28, 2):
        for k in range(0, 28, 2):
            patch = torch.flatten(image
                 [0, 0, j : j + 2, k : k
                 + 2])
            q_results = quanv_circuit(
                 patch)
            for c in range(4):
                out[j // 2, k // 2, c] =
                     q_results[c]

    return out
```



Models

Experimental Setup

To isolate the effect of the feature extractor, we designed two models with nearly identical parameter counts.

1. **Quanvolutional CNN (QNN)**
   - **Input:** Quantum-processed feature maps $(14 \times 14 \times 4)$.
   - **Backend:** The backend is a deep classical CNN consisting of 2 convolutional blocks (Conv2D $\rightarrow$ ReLU $\rightarrow$ MaxPool2D) + 2 Linear layers with dropout
   - **Parameters:** 169,962
2. **Classical CNN Baseline**
   - **Input:** Original images $(28 \times 28 \times 1)$.
   - **First Layer:** A classical ‘Conv2D‘ layer that mimics the QNN’s transformation (2x2 kernel, stride 2). This ensures that the feature map sizes before the backend are the same as the QNN.
   - **Backend: Identical** to the QNN.
   - **Parameters:** 169,982

UNIVERSITÀ DEGLI STUDI FIRENZE
Da un secolo, oltre.

Datasets & Workflow

Experimental Setup

Datasets Analyzed:

- MNIST, FMNIST, KMNIST (grayscale, 28x28)
- CIFAR10 (converted to grayscale, center-cropped to 28x28)

Preprocessing Workflow:

- Since the quantum circuit is not trained, we apply it as a **one-time preprocessing step**.
- The resulting quantum feature maps are saved to disk as *PyTorch* tensor files (.pt)
- **Practical Challenge:** This step is computationally expensive.
  - $\sim$**8 hours** to process one training set on a modern CPU.

UNIVERSITÀ DEGLI STUDI FIRENZE
Da un secolo, oltre.

Experiment 1: Performance on Full Datasets

Dataset	Classical CNN	QNN (1-Layer)	QNN (2-Layer)
MNIST	**99.05**	98.98	97.48
FMNIST	**90.56**	90.13	88.43
KMNIST	**95.16**	94.68	94.93
CIFAR-10	61.63	**62.21**	61.12

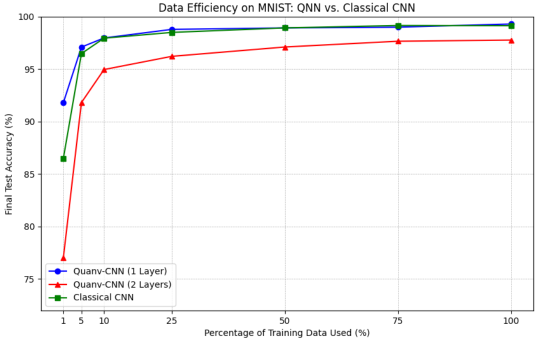
- On MNIST-like datasets, the trainable classical model achieves slightly higher accuracy.
- Considering the **8-hour quantum preprocessing time**, the classical approach is more practical for these tasks.
- The 1-layer QNN wins on the more complex CIFAR-10 dataset.
- Increasing quantum layers (2 vs 1) consistently degrades performance.

Experiment 2: Data Efficiency

Experiments

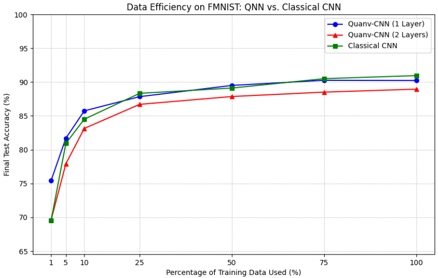
Goal: To determine whether the quantum features are ”good enough” to yield a performance advantage over classical feature maps in the low-data regime

- $\rightarrow$ Trained each model on random subsets of the training data: [1,5,10,25,50,75,100 %]
- Reported the final test accuracy for each training run

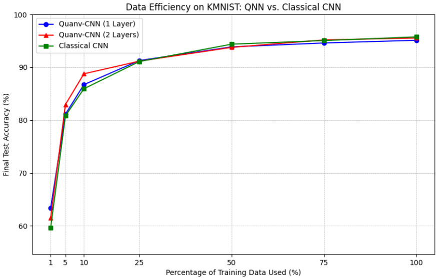


Data Efficiency on MNIST-like Datasets

Experiments



(a) Data efficiency on FMNIST

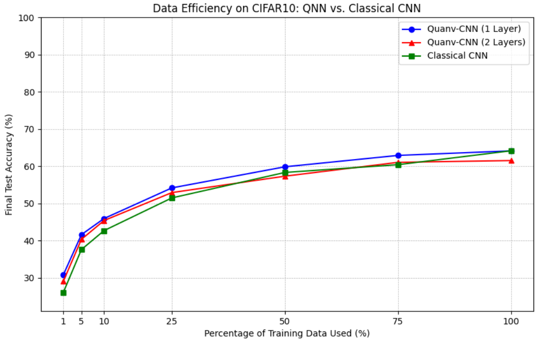


(b) Data efficiency on KMNIST

Experiment 2: Data Efficiency

Experiments

- The QNN (blue) consistently outperforms the classical CNN (green) in low-data regimes (1-25% of data) across all datasets.
- On CIFAR-10, this advantage is sustained across the **entire** data range.



Experiment 3: Robustness to Input Noise

Goal:

- To determine whether the quantum features confer any degree of resilience to input noise compared to the classical baseline.

Method:

- Evaluate pre-trained models on test sets corrupted with two distinct types of noise at varying levels of intensity.

Gaussian Noise (Continuous Perturbation)

Perturbs **every** pixel I_p by adding random noise n, where:

$$n \sim \mathcal{N}(0, \sigma^2)$$

The standard deviation σ controls the noise intensity.

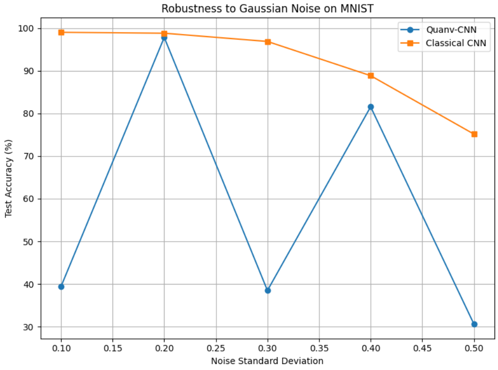
Salt & Pepper Noise (Sparse Errors)

Corrupts a random **fraction** p of pixels, setting them to the extreme values of 0 (pepper) or 1 (salt).

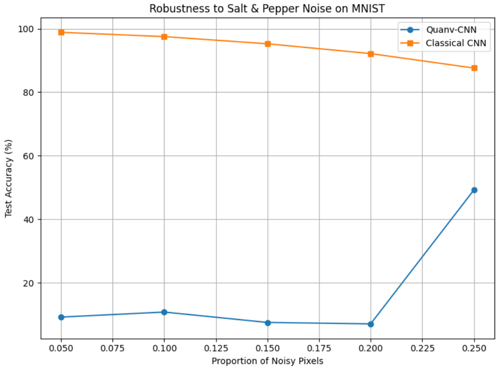
Da un secolo, oltre.

Experiment 3: Robustness to Input Noise (MNIST)

Experiments



(a) Gaussian Noise on MNIST



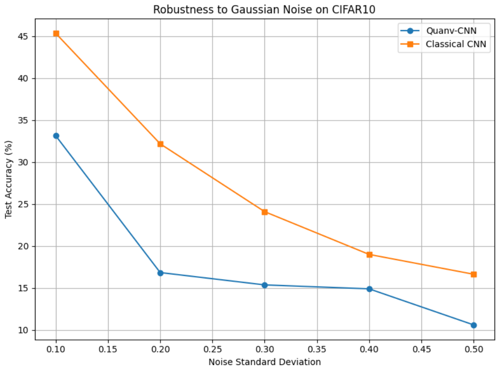
(b) Salt & Pepper Noise on MNIST

The classical model (orange) shows a graceful decline, while the QNN (blue) is highly erratic and brittle. This trend was consistent across all datasets.

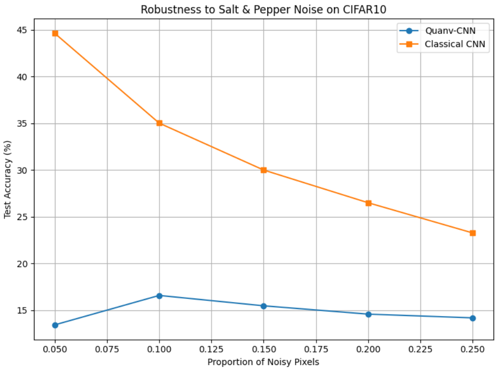
UNIVERSITÀ DEGLI STUDI FIRENZE
Da un secolo, oltre.

Experiment 3: Robustness to Input Noise (CIFAR10)

Experiments



(a) Gaussian Noise on CIFAR10



(b) Salt & Pepper Noise on CIFAR10

Experiment 4: Architectural Ablation

Goal:

- To isolate the "raw power" of the features themselves, without the influence of a deep CNN backend.

Method:

- We designed a **Minimal Classifier**: a *Flatten* layer followed by a single *Linear* layer.
- This minimal model was trained directly on the feature maps produced by the quantum and the classical feature extractors

Table: Test Accuracy (%) with a Minimal Classifier.

Dataset	Classical Features	Quantum Features
MNIST	92.57	**93.62**
FMNIST	84.29	**85.08**
KMNIST	70.30	**73.60**
CIFAR-10	29.64	**33.19**

Key Finding

The quantum features are consistently and significantly better for direct, shallow classification across all tested datasets.

UNIVERSITÀ DEGLI STUDI FIRENZE
Da un secolo, oltre.

QNN vs CNN

Conclusions

Quanvolutional CNN (QNN)

✓ **Strengths:**

- Performs well in the low data regime
- Powerful features for shallow classification

✗ **Weaknesses:**

- Sensitive to input noise
- High computational cost

Classical CNN (Baseline)

✓ **Strengths:**

- More robust to input noise
- Lower computational cost
- Performs better on most full, clean datasets

✗ **Weaknesses:**

- Lower performances in the low data regime
- Weaker features for shallow classification

UNIVERSITÀ DEGLI STUDI FIRENZE
Da un secolo, oltre.

Future Work

Conclusions

- **Trainable Quantum Kernels:** The most promising direction is to explore **variational quantum circuits**. A learnable kernel could potentially adapt to the data statistics to become robust to noise while retaining its powerful feature extraction capabilities.

- **Robust Encoding Schemes:** Investigating alternative, more resilient methods for encoding classical data into quantum states is crucial to mitigate the input sensitivity we observed.

- **Real Hardware Analysis:** Testing this architecture on NISQ-era devices to verify the paper's hypothesis that a random kernel may be inherently resilient to *quantum hardware noise*.

Thank You
Conclusions

Questions?

The complete project, including all code and experimental logs, is available on GitHub (code) and Weights and Biases (logs & plots):

github.com/jaysenoner99/quanv_nn

wandb.ai/jaysenoner/quanvolutional-nn-mnist?nw=nwuserjaysenoner1999