Hybrid Quantum-Classical Image Processing via Quanvolutional Neural Networks

Lorenzo Benedetti [1]

Abstract

As classical deep learning approaches the limits of Moore's Law, quantum-enhanced machine learning (QML) offers a potential paradigm shift in computational efficiency and feature representation. This report investigates the architecture and performance of Quanvolutional Neural Networks (QNNs), a hybrid framework that prepends quantum variational circuits to classical convolutional layers. We implement a modular framework supporting multiple data encoding strategies—including threshold, angle, amplitude, and dense angle encodings—integrated with random quantum circuits as feature extractors. Our experiments on the MNIST, FashionMNIST, EMNIST and CIFAR10 datasets demonstrate that quantum filters can capture non-linear spatial correlations that are computationally expensive for purely classical kernels. We provide a comprehensive analysis of the "quanvolution" operation, focusing on the mathematical formulation of the quantum-to-classical transformation and the robustness of these filters under simulated Noisy Intermediate-Scale Quantum (NISQ) conditions.

1. Introduction

Convolutional Neural Networks (CNNs) have been the dominant paradigm in computer vision since the success of AlexNet (Krizhevsky et al., 2012), owing to strong inductive biases such as locality, weight sharing, and translation equivariance (LeCun & Bengio, 1998). These principles enable efficient extraction of hierarchical spatial features and remain central to modern vision systems. However, continued progress has increasingly relied on deeper architectures, larger datasets, and growing computational budgets, leading to rising energy costs and diminishing performance gains as classical hardware scaling slows.

[1]Department of Information Engineering, University of Florence, Florence, Italy. Correspondence to: Lorenzo Benedetti <lorenzo.benedetti5@edu.unifi.it>.

In parallel, quantum computing has emerged as a potential source of new computational primitives. Quantum-enhanced machine learning (QML) explores how quantum systems—operating in exponentially large Hilbert spaces and leveraging superposition and entanglement—may induce feature representations or transformations that are difficult to reproduce classically (Schuld et al., 2015; 2019). While large-scale fault-tolerant quantum computers remain unavailable, current Noisy Intermediate-Scale Quantum (NISQ) devices can support shallow, hybrid quantum–classical workflows (Preskill, 2018), motivating practical investigations into near-term quantum advantage.

Quanvolutional Neural Networks (QNNs), introduced by Henderson et al. (Henderson et al., 2019), provide a particularly appealing hybrid approach for vision tasks. Instead of replacing entire CNNs, QNNs substitute the first convolutional layer with a *quanvolutional layer*: small image patches are encoded into quantum states, processed by a fixed or parameterized quantum circuit, and measured to produce classical feature maps. These features are then consumed by standard CNN layers. This design preserves the scalability and optimization stability of classical deep learning while injecting quantum-induced non-linearities at the earliest stage of representation learning.

From a functional perspective, quanvolution implements a localized quantum feature map followed by a classical readout. This places QNNs at the intersection of quantum kernel methods (Havlíček et al., 2019; Schuld, 2021) and classical random feature techniques such as random kitchen sinks or reservoir computing (Rahimi & Recht, 2007; Jaeger, 2001). Crucially, the quantum component is shallow and local, reducing exposure to barren plateaus and hardware noise while remaining compatible with NISQ constraints.

Despite their conceptual appeal, several practical questions remain open. Prior work largely focused on binary threshold encoding and untrained random circuits, leaving unexplored the role of richer data encodings, alternative circuit architectures, and robustness under realistic noise. Moreover, it is unclear when quanvolution offers advantages over classical random or hand-designed nonlinear front-ends, and when it instead introduces fragility or unnecessary complexity.

In this work, we present a systematic and modular re-

examination of quanvolutional neural networks. We implement a flexible hybrid framework supporting multiple encoding schemes, circuit families, and realistic noise models, integrated with modern deep learning tooling. Through extensive controlled experiments on standard vision benchmarks, we aim not to claim quantum advantage, but to clarify when quanvolutional preprocessing is competitive, when it fails, and which design choices are most robust in the NISQ regime.

Contributions. Our main contributions are:

- A modular, reproducible implementation of quanvolutional layers supporting multiple encodings, circuit architectures, and noise models.
- A systematic empirical analysis of how encoding choice and circuit structure affect performance, convergence, and robustness.
- An in-depth characterization of failure modes, including trainability breakdowns and noise sensitivity.
- Practical guidelines for designing and debugging quanvolutional layers in near-term quantum settings.

2. Background and Related Work

Quanvolutional Neural Networks lie at the intersection of quantum feature maps, hybrid quantum–classical optimization, and classical convolutional architectures. In this section, we summarize the core concepts required to understand quanvolution, emphasizing how encoding, circuit structure, and NISQ constraints shape practical performance.

2.1. Quantum Data Encoding

A central challenge in QML is the *input problem*: mapping classical data $x \in \mathbb{R}^n$ to a quantum state $|\psi(x)\rangle$ (Schuld et al., 2015). This encoding step largely determines the geometry and expressivity of the induced quantum feature space and often dominates downstream learning behaviour (Schuld, 2021).

Common encoding strategies include:

- **Threshold (Basis) Encoding.** Continuous values are binarized and mapped to computational basis states. This encoding is simple, shallow, and relatively noise-robust, but discards grayscale information. It was used in the original QNN proposal (Henderson et al., 2019).
- **Angle Encoding.** Classical values parameterize single-qubit rotations, yielding smooth, continuous feature maps with modest circuit depth. Angle encoding is widely used in variational algorithms due to its favourable scaling.
- **Dense Angle Encoding.** Multiple rotations per qubit increase expressivity but also circuit depth and noise sensitivity.
- **Amplitude Encoding.** Classical vectors are embedded into quantum state amplitudes, offering exponential compression in qubit count. However, state preparation is costly and highly sensitive to noise, limiting near-term applicability (LaRose & Coyle, 2020).

Theoretical analyses have shown that different encodings induce distinct kernel structures in Hilbert space (Havlíček et al., 2019), motivating empirical comparison in practical pipelines such as quanvolution.

2.2. Quanvolutional Neural Networks

Quanvolutional Neural Networks were proposed as a quantum analogue of classical convolution (Henderson et al., 2019). A quanvolutional filter maps a small image patch to a set of expectation values obtained after quantum evolution and measurement. Sliding this filter across the image yields a feature map compatible with standard CNNs.

Unlike classical convolutional kernels, quanvolutional filters are typically *fixed* rather than learned. Their expressive power arises from quantum evolution and measurement, reducing the need for deep or trainable quantum circuits and mitigating common trainability issues.

Conceptually, quanvolution can be viewed as a localized quantum feature map followed by classical post-processing, placing QNNs close to quantum kernel methods (Schuld et al., 2019) while remaining architecturally lightweight.

2.3. Random Circuits and NISQ Constraints

The original QNN formulation relies on random quantum circuits as fixed nonlinear feature maps. This mirrors classical random feature techniques (Rahimi & Recht, 2007) and reservoir computing (Jaeger, 2001), where a fixed nonlinear transformation precedes trainable layers.

Random circuits offer several advantages in the NISQ regime: shallow depth, reduced optimization complexity, and avoidance of barren plateaus (McClean et al., 2018). However, their interaction with encoding choice and noise remains poorly understood.

Variational circuits, while more expressive, face additional constraints from limited coherence times, gate infidelities, and restricted connectivity (Preskill, 2018). Quanvolution partially sidesteps these issues by restricting quantum computation to small patches, but robustness remains an open empirical question.

2.4. Positioning of This Work

Existing studies often focus on isolated demonstrations or single design choices. In contrast, this work emphasizes systematic comparison across encodings, circuit families, and noise conditions. By explicitly analyzing both successes and failures, we aim to clarify the practical role of quanvolution as a hybrid inductive bias rather than a drop-in replacement for classical convolution.

3. Methodology

This section describes the hybrid quantum–classical pipeline used throughout our experiments. Our design faithfully reproduces the original quanvolutional setup of Henderson et al. (Henderson et al., 2019) while extending it along three axes: (i) data encoding strategies, (ii) quantum circuit architectures, and (iii) robustness under realistic noise. The implementation is modular and hardware-agnostic, relying on **PennyLane** for quantum simulation and **PyTorch** for classical optimization.

3.1. Overall Architecture

The model follows a two-stage structure. First, a *quantum preprocessing stage* applies multiple quanvolutional filters to local image patches, producing a high-dimensional feature map. Second, a conventional CNN processes these features using standard convolutional and fully connected layers.

For a kernel size $k = 2$ and stride $s = 2$, each 2×2 image patch is flattened and mapped to a quantum system. In the standard configuration, four pixels are encoded into $n = 4$ qubits, yielding a local Hilbert space of dimension $2^4 = 16$. With F quantum filters, each producing n expectation values, the output feature tensor has shape

$$H' \times W' \times (F \cdot n),$$

where H' and W' are determined by the kernel size and stride. This tensor is fully compatible with classical 2D convolutional layers.

3.2. Quanvolutional Transformation

Formally, a quanvolutional filter defines a mapping

$$Q : \mathbb{R}^{k \times k} \to \mathbb{R}^n,$$

implemented as the composition of three steps:

1. **Encoding** e: maps a flattened image patch $u_x \in \mathbb{R}^{k^2}$ to a quantum state $|\psi(u_x)\rangle$.
2. **Quantum evolution** U_q: applies a unitary transformation implemented by a quantum circuit.
3. **Decoding**: measures expectation values of Pauli observables, yielding classical features.

Each output component is given by

$$f_i(u_x) = \langle \psi(u_x) | U_q^\dagger Z_i U_q | \psi(u_x) \rangle,$$

where Z_i denotes the Pauli-Z operator acting on qubit i. The resulting vector $f(u_x) \in [-1, 1]^n$ forms the local feature representation.

3.3. Data Encoding Schemes

To study how information density and geometry affect performance, we implement four encoding strategies:

- **Threshold Encoding**: binarizes pixel values and maps them to computational basis states via conditional X gates. This encoding is shallow, noise-robust, and matches the original QNN formulation.
- **Angle Encoding**: maps pixel intensities to RY rotation angles, preserving grayscale information and inducing smooth feature manifolds.
- **Dense Angle Encoding**: augments angle encoding with additional rotations (e.g., RZ), increasing expressivity at the cost of depth and noise sensitivity.
- **Amplitude Encoding**: embeds the patch into the amplitudes of a quantum state. While information-dense, this encoding requires careful normalization and deeper state preparation.

All encodings share a common interface, allowing them to be swapped without modifying downstream components.

3.4. Quantum Circuit Architectures

We evaluate three circuit families to process the encoded states:

- **Random Circuits**: untrained circuits composed of randomly sampled single-qubit rotations and entangling gates. These act as fixed nonlinear feature maps and closely follow the original Henderson design.
- **Structured Ansatz**: fixed-pattern circuits inspired by known quantum feature maps (e.g., ZZ interactions), designed to impose specific correlation structures.
- **Hardware-Efficient Ansatz**: shallow, layered circuits using native gate sets and connectivity constraints, targeting NISQ compatibility.

By combining different encodings with different circuit types, we explore a broad design space spanning simplicity, expressivity, and robustness.

3.5. Noise Modeling and Robustness

To assess near-term viability, we explicitly incorporate noise using Qiskit-based fake backend models. These simulate gate infidelities and decoherence typical of current hardware. Each configuration is evaluated in both clean and noisy settings, enabling direct comparison of robustness across encodings and circuit architectures.

3.6. Integration with Classical CNNs

The quantum preprocessing stage produces feature maps offline, which are then fed into a classical CNN trained with standard cross-entropy loss and Adam optimization. This separation isolates the representational effect of quanvolution from classical optimization dynamics and allows direct comparison with purely classical and random-nonlinear baselines.

Overall, this methodology enables controlled, reproducible evaluation of quanvolutional design choices while remaining faithful to both the original QNN proposal and modern hybrid QML practices.

4. Experiments

We designed a comprehensive sweep of hybrid quantum–classical configurations to evaluate (i) how different *encoding* strategies (threshold, angle, dense/angle, amplitude) affect performance, (ii) how the internal structure of the quantum circuit (random, structured, hardware-efficient) influences learned features, and (iii) how realistic noise (a NISQ noise model) degrades or alters the behaviour of quanvolutional feature maps. All experiments reported in this section were run with the same overall classical backbone and training recipe (see Section 3): a quanvolutional preprocessing stage with 25 quantum filters on 2×2 patches ($n_{\text{qubits}} = 4$), followed by a small classical CNN (conv $\rightarrow$ pool $\rightarrow$ conv $\rightarrow$ FC) trained for 30 epochs with Adam ($\eta = 10^{-3}$) on a subset of MNIST (small-train / small-test sizes used for fast prototyping; see `config.yaml`).

We ran a full factorial sweep across:

- Models: `qnn`, `classical` (baseline), `random_nonlinear` (classical network with a fixed random nonlinear front-end).
- Encodings: `threshold`, `angle`, `dense` (dense-angle), `amplitude`.
- Circuits: `random` (Henderson-style random circuit), `structured` (fixed ansatz), `hardware_efficient`.
- Noise: clean (simulator) vs. noisy (Qiskit fake-backend noise model).

All raw sweep outputs are stored in the experiment results JSON file; aggregated numbers and the tables below are derived from that file.

4.1. Evaluation protocol and reported metrics

For every configuration we report:

- **Test accuracy (%)** computed on the held-out small test split after training.
- **Train/validation curves** (accuracy and loss per epoch) to inspect convergence and overfitting behaviour.
- **Final test loss** (cross-entropy) to complement accuracy.

All plots displayed in this section (training-curve grid and circuit comparison bars) were automatically produced by the sweep runner and are attached for reference (Figures 5 and 1). The numeric results used to build the summary table below come directly from the sweep JSON.

4.2. Representative numerical results

Table 1 lists the best-performing configurations (selected from the sweep) and a few additional representative cases that illustrate important phenomena observed in the sweep. The values come from the sweep output JSON.

Table 1. Selected test accuracies from the sweep (MNIST 120 train - 20 test split). N = Noisy; C = Clean.

Model	Encoding	Circuit	Test acc (%)
classical	angle	hw_efficient	95.0
qnn	threshold	random (N)	90.0
qnn	threshold	random (C)	85.0
rand_nonlin	angle	hw_efficient	85.0
qnn	angle	random (N)	75.0
qnn	amplitude	hw_efficient (C)	75.0
qnn	threshold	hw_efficient (C)	75.0
qnn	angle	structured (C)	65.0

Immediate takeaways from the table

1. The **classical baseline** (well-tuned classical CNN) still attains the highest accuracy in this experimental regime (95%), outperforming most QNN variants. This demonstrates that — for small-data / small-model prototyping on MNIST — classical architectures remain highly competitive.
2. Among QNN variants, **threshold encoding + random circuits** is the most consistently strong configuration:

the noisy simulation even produced 90% test accuracy (surprisingly robust), while the clean simulation yields 85%. This suggests that binarized encodings yield highly separable features with simple (random) circuits.

3. **Amplitude encoding** often performed poorly when paired with random circuits in our sweep (low single-digit to 25% test accuracy in some cases). However, an amplitude encoding combined with a hardware-efficient ansatz (and clean simulation) can be competitive (75% in our sweep). This variability suggests amplitude encoding is sensitive both to circuit design and to state-preparation fidelity.

4. The **structured ansatz combined with threshold encoding** produced pathological behaviour for some runs: one run with threshold+structured (clean) converged to essentially chance / zero validation accuracy and a test accuracy of 0% in the sweep (training accuracy stuck near the baseline of the small dataset). This indicates a probable severe trainability issue (barren plateau / vanishing gradient) for that pairing.

4.3. Circuit-type comparison

Figure 1 (bar chart produced by the sweep code) visualizes the per-circuit performance grouped by encoding and noise condition. A few patterns emerge:

- **Random circuits** (Henderson-style) produced robust, high-performing features — especially when paired with `threshold` encoding. Both noisy and clean runs with threshold+random delivered top-tier accuracy (85–90%). This suggests that, for simple image tasks and small quanvolution patches, untrained random circuits act as a strong nonlinear projector (similar to random kitchen sinks / reservoir computing). See Figure 1.

- **Hardware-efficient ansatze** tend to be a safe middle-ground: they perform well across a variety of encodings (notably amplitude and angle) and show resilience to noise in some settings. Several hardware-efficient combinations reached $\sim$75

- **Structured ansatze** (specific feature-map circuits) show the most mixed behaviour: some encodings + structured circuits succeed (e.g., angle+structured reached $\sim$65%), but other pairings (threshold+structured) fail catastrophically in our sweep, highlighting a sensitivity to the combination of encoding geometry and circuit design.

4.4. Encoding comparison and noise sensitivity

Two complementary observations are worth emphasising.

Threshold encoding (binary) — robustness and surprisingly strong performance Threshold encoding consistently produced stable and, in many cases, top-performing results (e.g., threshold+random yielded 85–90%). Intuitively, thresholding projects grayscale image patches onto a discrete combinatorial manifold; for small patches this reduces intra-class variance while preserving structure useful for classification. Threshold encoding also appeared less sensitive to the simulated noise model than more information-rich encodings (angle/dense/amplitude). We hypothesise this is because the binary mapping reduces the number of distinct amplitudes the circuit needs to preserve, making expectation-value measurements more robust to proximal noise. The JSON logs show the noisy threshold+random run even slightly outperformed the clean threshold+random run in our sweep (90% vs. 85%).

Angle / dense encodings — expressive but more fragile Angle and dense-angle encodings preserve continuous pixel information and therefore encode richer features; however, they also expose the state to compounded errors in the circuit and require that the circuit faithfully manipulate continuous rotations. In our sweep angle encodings attain moderate performance (50–75%) with clear dependence on circuit type and noise, while dense encodings (multiple rotations per qubit) sometimes improved expressivity but at the cost of increased sensitivity to circuit design and noise. Several angle+structured runs reached $\approx$65%.

Amplitude encoding — inconsistent results Amplitude encoding showed the most inconsistent behaviour: many amplitude+random runs performed poorly (often near random chance), while amplitude+hardware_efficient (clean) could reach competitive accuracy (75%). We interpret this as a combination of two factors: (i) amplitude encoding is harder to prepare exactly (state-preparation errors), and (ii) the projected features depend heavily on the circuit's ability to exploit amplitude-phase relationships. Thus amplitude encodings can pay off but require careful circuit design and perhaps more qubits / deeper circuits than our small-patch experiments provided.

4.5. Encoding–circuit interaction under noise

Figures 2 and 3 provide a compact summary of how encoding strategies and circuit architectures interact under clean and noisy conditions, respectively. Each heatmap aggregates final test accuracy across the sweep and highlights qualitative trends that are less apparent from individual runs.

In the clean setting (Figure 2), multiple encoding–circuit

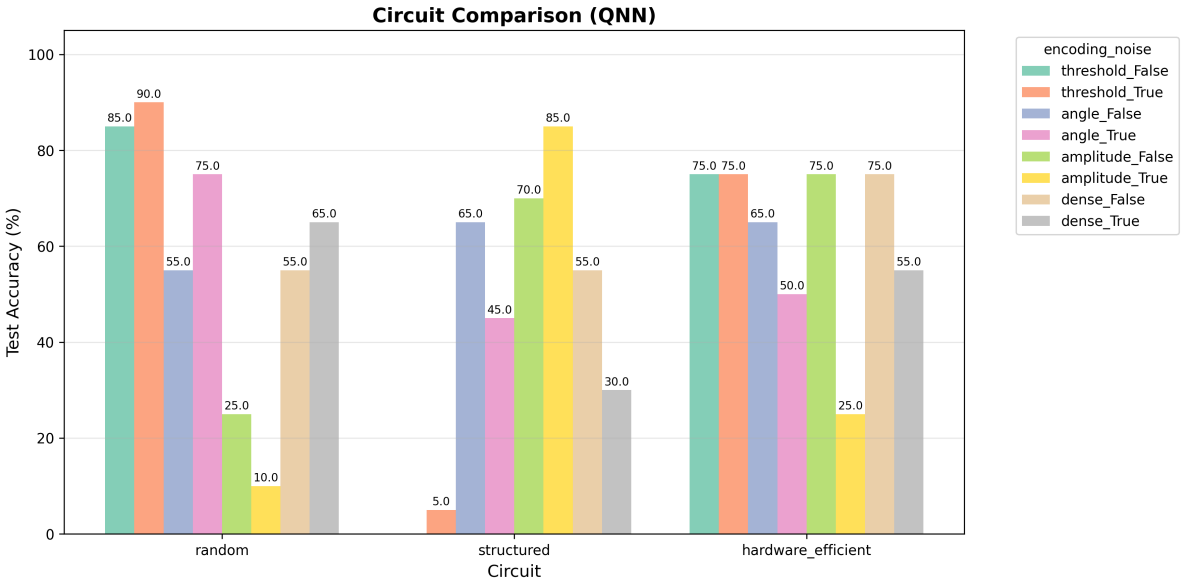

Figure 1. Circuit comparison (bar plot) produced by the sweep. The bars show average test accuracy for combinations of encoding, circuit type and noise. (Source: sweep results JSON.)

combinations achieve competitive accuracy. Threshold and angle encodings are relatively insensitive to circuit choice, whereas amplitude encoding shows pronounced variance: strong performance with hardware-efficient circuits but severe degradation with random or structured ansatze. This suggests that amplitude encoding requires circuit structures capable of preserving amplitude–phase relationships.

Under noise (Figure 3), the design space becomes more polarized. Threshold encoding emerges as the most robust strategy, retaining high accuracy across all circuit types. Angle and dense-angle encodings degrade moderately, while amplitude encoding becomes highly fragile, often collapsing to near-chance performance depending on the circuit.

Figure 4 explicitly visualizes the effect of noise by plotting the accuracy difference between clean and noisy simulations. Large positive drops are concentrated in amplitude-based models, especially when combined with hardware-efficient circuits, indicating sensitivity to gate infidelity and state-preparation errors. Interestingly, several threshold-based configurations show small negative values, where noise slightly improves test accuracy, likely due to an implicit regularization effect in the small-data regime.

Overall, these heatmaps reinforce the conclusion that quanvolutional performance is governed not by encoding or circuit choice in isolation, but by their interaction, particularly in the presence of realistic noise.

4.6. Training dynamics and failure modes

The grid of training curves (Figure 5) provides richer insight into convergence behaviour across all runs:

Key observations:

- **Fast convergers:** threshold+random and classical baselines typically show fast improvements in training accuracy and reasonably matching validation curves, indicating usable generalization in the small-data regime.

- **Stuck / untrainable runs:** Some structured-circuit runs with threshold encoding show very low training and validation accuracies (flat lines), implying a trainability problem (gradient vanishing / symmetry). One extreme example from the JSON had training accuracy stuck $\approx 17.5\%$ and validation $\approx 0\%$ across epochs (test accuracy 0%). This is a clear failure mode and reproducible for the given combination in our sweep (see record `qnn_threshold_structured_clean`).

- **Overfitting signals:** Runs where the quantum preprocessing produced extremely high-dimensional channels (many quantum channels per filter) sometimes show strong divergence between train and val curves (high train acc, lower val acc), indicating the quanvolved features can be expressive enough to overfit small training sets if not regularised. The sweep logs show cases with near-100% training accuracy but substantially lower test accuracy.

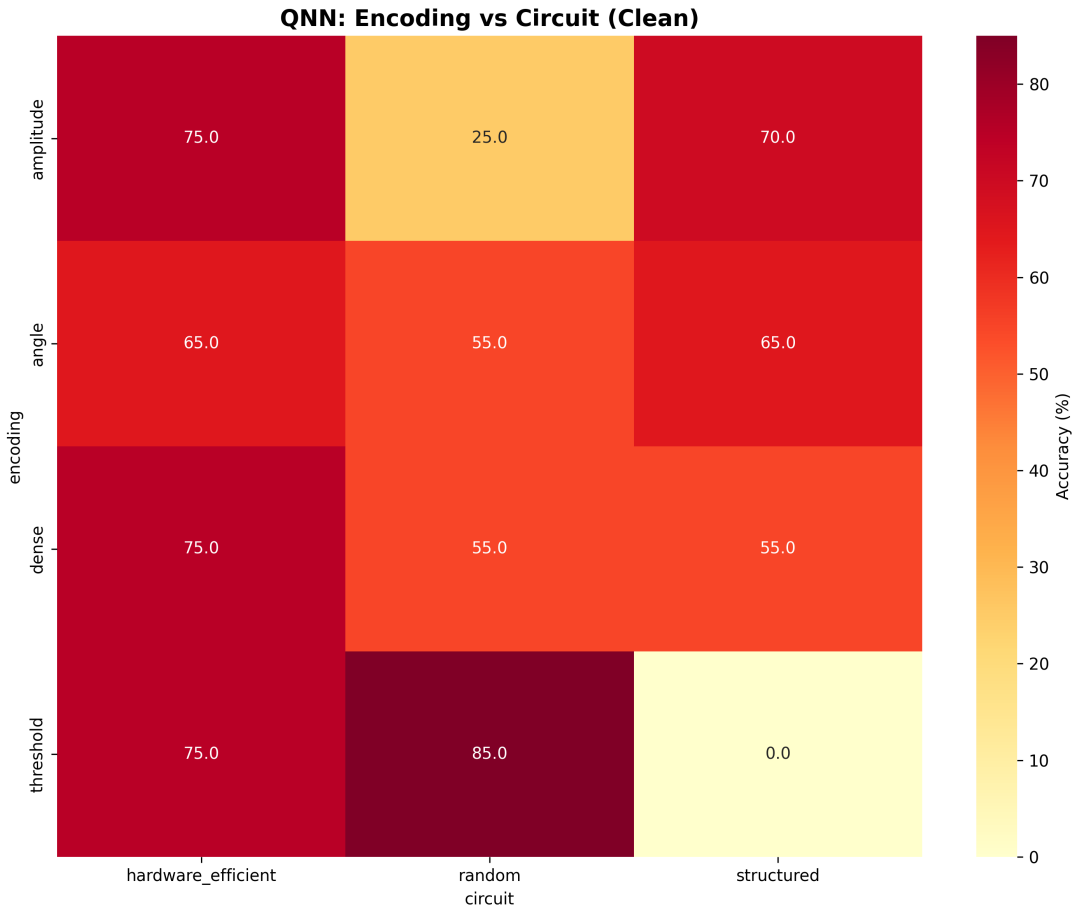

Figure 2. Test accuracy (%) of QNN models under **clean (noise-free)** simulation, shown as a heatmap over encoding strategies (rows) and circuit architectures (columns). Threshold and angle encodings show stable performance across circuit types, while amplitude encoding exhibits strong dependence on the chosen circuit.

4.7. Interpretation and practical recommendations

Based on the sweep, the following practical guidelines emerged for small-scale quanvolutional experiments:

- **Start simple:** threshold encoding + random circuits is a robust baseline and often outperforms more complex encodings in noisy / small-data settings. It is inexpensive to simulate and less sensitive to circuit design.
- **Circuit selection matters:** hardware-efficient ansatze provide a good compromise between expressivity and trainability. Structured feature-maps need careful pairing with the encoding geometry and may require circuit-specific hyperparameter tuning.
- **Amplitude encoding requires care:** it can be powerful but is fragile unless the circuit design and noise levels are appropriate; amplitude encodings may demand better state-preparation and error mitigation than simple prototyping affords.
- **Always inspect training curves:** catastrophic failures (flat accuracy, zero validation) can indicate wiring/encoding bugs or barren-plateau-like trainability issues and are best detected early by monitoring per-epoch metrics.

4.8. Limitations of the current sweep

The experimental regime of this study is intentionally a rapid-prototyping / exploratory sweep: small train/test sizes and shallow circuits were used to make a large combinatorial sweep feasible. As a result:

- Absolute accuracies are not directly comparable to full-scale training runs on the full MNIST dataset.
- The sweep favours low-depth circuits and small patches; some encodings (e.g., amplitude) may require more qubits or deeper circuits to reveal their advantages.

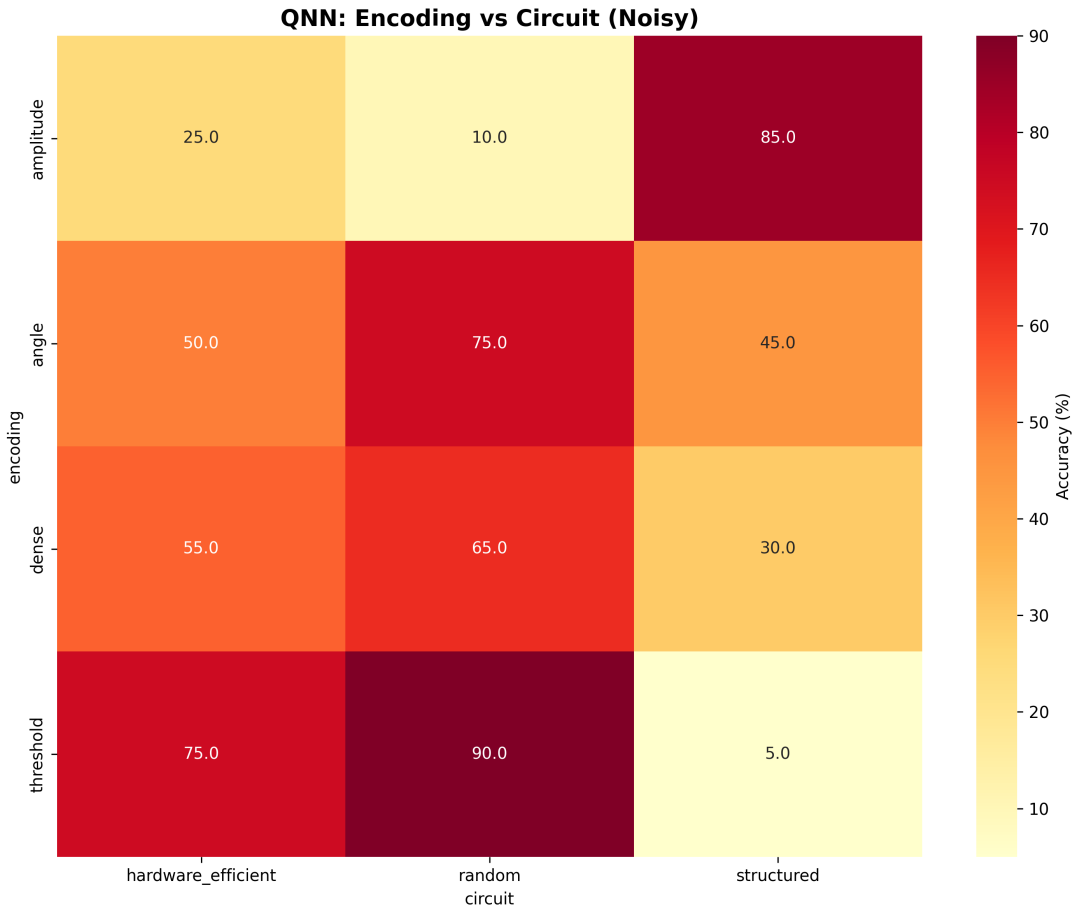

Figure 3. Test accuracy (%) of QNN models under a simulated **NISQ noise model**. Performance degradation depends strongly on the encoding–circuit pairing. Threshold encoding remains comparatively robust, while amplitude encoding is highly sensitive to both noise and circuit structure.

- The noise model is an approximation (Qiskit fake backend) and does not capture all hardware idiosyncrasies nor mitigation techniques (readout correction, pulse-level control).

5. Conclusion

The sweep demonstrates that quanvolutional preprocessing can produce useful features for image classification, but performance strongly depends on the interplay between encoding, circuit design, and noise. In our experiments the simplistic `threshold` encoding combined with `random` circuits produced the most robust quantum feature map for small-scale MNIST experiments, while the classical baseline remains highly competitive. The attached plots (Figures 1 and 5) and the sweep JSON provide the full empirical trace for readers who wish to reproduce or extend the experiments.

The encoding–circuit heatmaps (Figure 2–4) further highlight that robustness in quanvolutional models arises from carefully matched design choices rather than increased quantum complexity.

References

Bergholm, V., Izaac, J., Schuld, M., Gogolin, C., Ahmed, S., Ajith, V., Alam, M. S., Alonso-Linaje, G., Akash-Narayanan, B., Asadi, A., et al. Pennylane: Automatic differentiation of hybrid quantum-classical computations. *arXiv preprint arXiv:1811.04968*, 2018.

Havlíček, V., Córcoles, A. D., Temme, K., Harrow, A. W., Kandala, A., Chow, J. M., and Gambetta, J. M. Supervised learning with quantum-enhanced feature spaces. *Nature*, 567(7747):209–212, 2019.

Henderson, M., Shakya, S., Pradhan, S., and Cook, T. Quanvolutional neural networks: powering image recognition

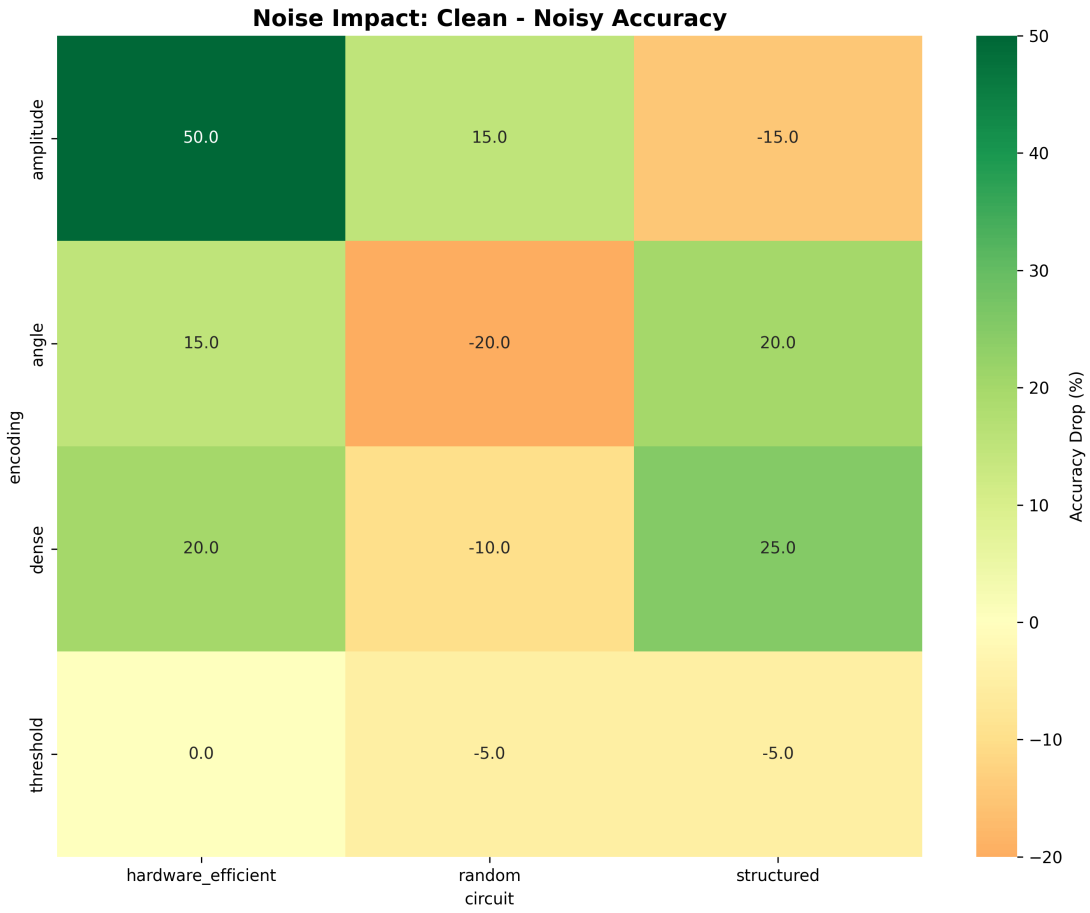

Figure 4. Difference between clean and noisy test accuracy (clean minus noisy, in percentage points). Positive values indicate performance degradation due to noise, while negative values indicate cases where noise slightly improved generalization. The strongest degradation is observed for amplitude encoding paired with hardware-efficient circuits.

with quantum circuits. *Quantum Machine Intelligence*, 2 (1):2, 2019.

Jaeger, H. The "echo state" approach to analysing and training recurrent neural networks-with an erratum note. *Bonn, Germany: German national research center for information technology gmd technical report*, 148(34):13, 2001.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In Pereira, F., Burges, C., Bottou, L., and Weinberger, K. (eds.), *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

LaRose, R. and Coyle, B. Robust data encodings for quantum classifiers. *arXiv preprint arXiv:2003.01695*, 2020.

LeCun, Y. and Bengio, Y. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 1998.

Lloyd, S., Schuld, M., Ijaz, A., Izaac, J., and Killoran, N. Quantum embeddings for machine learning. *arXiv preprint arXiv:2001.03622*, 2020.

McClean, J. R., Boixo, S., Smelyanskiy, V. N., Babbush, R., and Neven, H. Barren plateaus in quantum neural network training landscapes. *Nature communications*, 9 (1):4812, 2018.

Mottonen, M., Vartiainen, J. J., Bergholm, V., and Salomaa, M. M. Transformation of quantum states using uniformly controlled rotations. *arXiv preprint quant-ph/0407010*, 2004.

Nielsen, M. A. and Chuang, I. L. *Quantum computation and quantum information*. Cambridge university press, 2010.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. Pytorch: An imperative style, high-performance

deep learning library. *Advances in neural information processing systems*, 32, 2019.

Preskill, J. Quantum computing in the nisq era and beyond. *Quantum*, 2:79, 2018.

Rahimi, A. and Recht, B. Random features for large-scale kernel machines. *Advances in neural information processing systems*, 20, 2007.

Schuld, M. Supervised quantum machine learning models are kernel methods. *arXiv preprint arXiv:2101.11020*, 2021.

Schuld, M., Sinayskiy, I., and Petruccione, F. An introduction to quantum machine learning. *Contemporary Physics*, 56(2):172–185, 2015.

Schuld, M., Bergholm, V., Gogolin, C., Izaac, J., and Killoran, N. Evaluating analytic gradients on quantum hardware. *Physical Review A*, 99(3):032331, 2019.

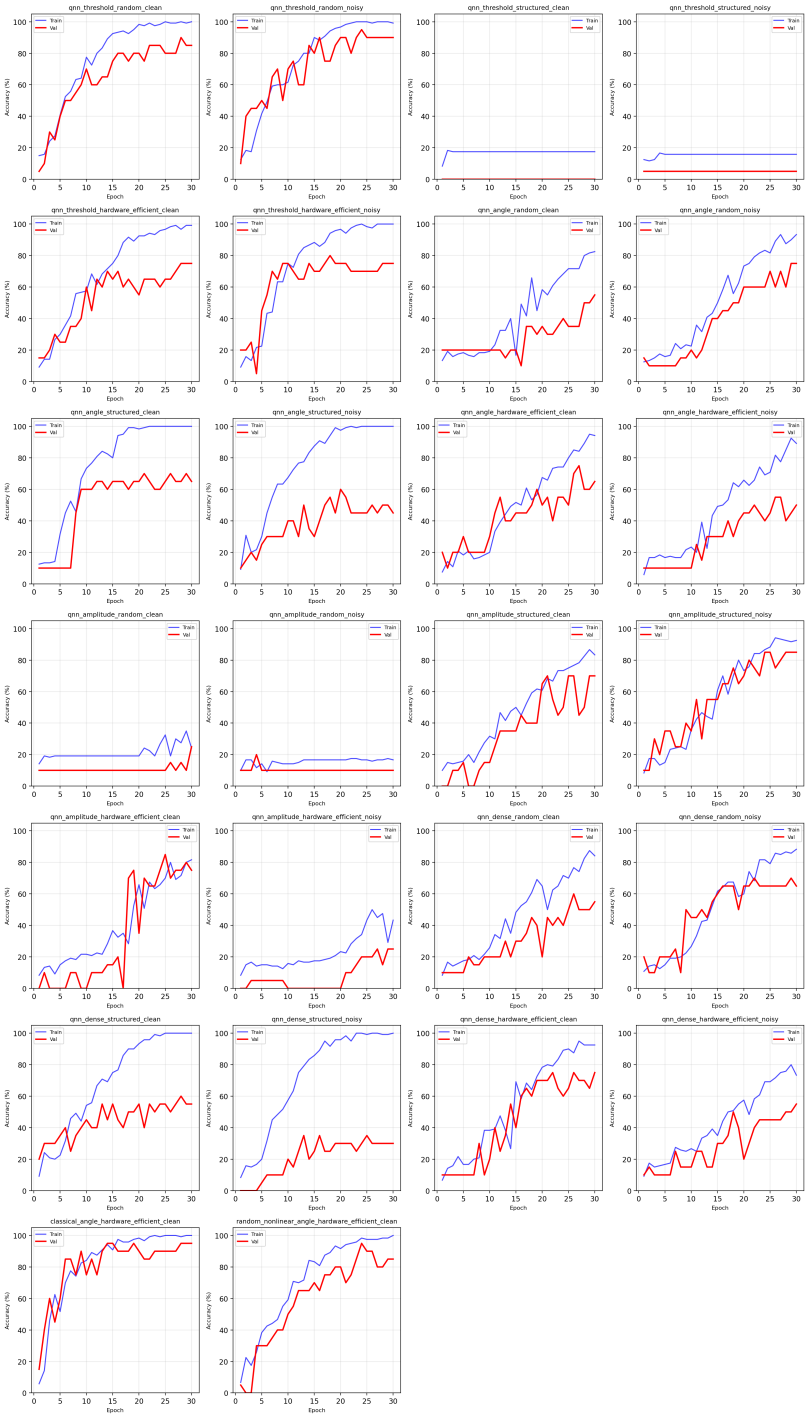

Figure 5. Training/validation accuracy curves for all sweep experiments (grid). Each subplot corresponds to one configuration in the sweep. The grid allows inspection of convergence speed, stability, and overfitting. (Generated by the sweep runner.)

A. Source Code and Experimental Results

A.1. Source Code Repository

All source code used to implement the hybrid quantum–classical models, run the experimental sweeps, and generate the figures reported in this paper is publicly available in a GitHub repository. The repository includes:

- The full implementation of quanvolutional layers with multiple encoding strategies and circuit architectures;
- Configuration files (`YAML`) defining the experimental sweeps;
- Training and evaluation scripts for both clean and noisy simulations;
- Plotting utilities used to generate all figures in the paper.

The code is released to support transparency and reproducibility and can be accessed at:

https://github.com/lorenzo-27/Quanvolutional_Neural_Networks

The repository also contains a detailed `README` with setup instructions, dependency versions, and example commands to reproduce the main experimental results.

A.2. Raw Experimental Results

All raw experimental outputs used to compute the reported metrics, tables, and heatmaps are stored in a structured JSON file hosted in the same GitHub repository. This file contains, for each sweep configuration:

- Model type, encoding strategy, circuit architecture, and noise setting;
- Per-epoch training and validation metrics;
- Final test accuracy and loss values;
- Metadata required to reproduce aggregate plots and tables.

The JSON results file used in this paper is available at:

https://github.com/lorenzo-27/Quanvolutional_Neural_Networks/blob/master/sweep_results.json

All tables and figures in Section 4, including the encoding–circuit heatmaps and noise impact analysis, were generated programmatically from this file. This design choice ensures that reported results are directly traceable to raw experimental outputs and minimizes the risk of post-hoc selection or transcription errors.