Quanvolutional Neural Networks: Powering Image Recognition with Quantum Circuits

Jay Senoner
Quantum Machine Learning Exam Report
jay.senoner@edu.unifi.it

Abstract

This report details the implementation and rigorous evaluation of a Quanvolutional Neural Network (QNN), a hybrid quantum-classical model for image recognition proposed by Henderson et al. We implement a non-trainable quanvolutional layer using a random quantum circuit, serving as a feature extractor for a subsequent classical convolutional neural network. To assess its efficacy, we conduct a multifaceted comparison against a carefully designed classical baseline on several image datasets, including MNIST, Fashion-MNIST, KMNIST and the more challenging CIFAR10. Our analysis spans multiple dimensions: data efficiency, robustness to noise, feature space quality via t-SNE, architectural ablation, and error analysis using confusion matrices. The results indicate that while the classical baseline achieves slightly higher accuracy on clean, full datasets, the QNN exhibits a notable advantage in the low-data regime across different domains. These findings suggest that while quanvolutional layers can produce powerful features from clean data, granting a potential advantage in real-world scenarios with limited data, their sensitivity to input perturbations and the required computational costs presents a critical challenge for practical applications. Code is available at github.com/jaysenoner99/quanv_nn.

Future Distribution Permission

The author(s) of this report give permission for this document to be distributed to Unifi-affiliated students taking future courses.

1. Introduction

Convolutional Neural Networks (CNNs) have become the state-of-the-art for a wide array of computer vision tasks. Their success lies in the convolutional layer, which applies local filters across an image to learn hierarchical patterns. If the input is an image, small local regions are sequentially processed with the same kernel. The results obtained for each region are usually associated to different channels of a single output pixel. The union of all the output pixels produces a new image-like object, which can be further processed by additional layers. The paper "Quanvolutional Neural Networks: Powering Image Recognition with Quantum Circuits" by Henderson et al. [1] introduces a quantum analogue to this fundamental building block.

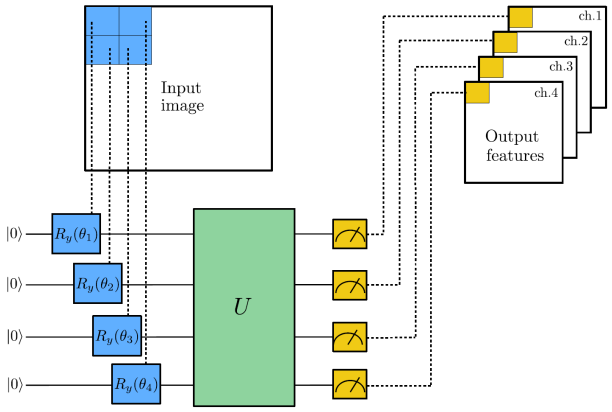


Figure 1: A quanvolutional layer processing image patches to produce feature maps

The core idea is to replace or augment a classical convolutional layer with a "quanvolutional" layer. This layer uses a quantum circuit to transform local patches of an image into feature maps. The motivation is that quantum circuits, through principles like superposition and entanglement, can generate highly complex and non-linear transformations that might be difficult to replicate clas-

sically, potentially leading to more powerful feature representations.

This report details a project undertaken to implement and, more importantly, rigorously evaluate this hybrid model. Our work goes beyond a simple reimplementation by conducting controlled experiments to answer the central question: *Under what conditions, if any, does the quanvolutional feature extractor offer a tangible advantage over a comparable classical one?*

To this end, we compare the QNN against a baseline classical CNN on multiple datasets. We analyze their performance across several axes:

- **Data Efficiency:** How well do the models learn from limited data?
- **Robustness:** How gracefully does performance degrade on noisy images?
- **Feature Quality:** What does the learned feature space look like?
- **Architectural Ablation**: How good are the quantum features on their own, without a deep classical network to process them?

This analysis provides a comprehensive picture of the QNN's practical strengths and weaknesses, revealing a critical trade-off between its data efficiency and its robustness to noise.

2. Quanvolutional Neural Networks

The quanvolutional layer operates by scanning a random non-trainable quantum circuit over an input image. Given a grayscale image u as input, the process consists of several steps for each local patch u_x . The first step is data encoding, where the pixel values of each patch are translated into an initialized state of the quantum circuit q by applying an encoding function $e(\cdot)$, obtaining the initialized state $i_x = e(u_x)$. After the quantum circuit is applied to the initialized state i_x, the result of the quantum computation will be an output quantum state o_x, with the relationship $o_x = q(i_x) = q(e(u_x))$. To extract scalar values to be used by the following classical convolutional layers, a decoding function $d(\cdot)$ is applied to the output quantum state o_x, obtaining the final decoded state $f_x = d(o_x) = d(q(e(u_x))$. The obtained scalar values are integrated into a feature map, which subsequently serves as input for further processing by classical layers within the deep neural network architecture. This step is crucial in hybrid quantum–classical models, as it enables the extraction of quantum-informed features that can be effectively leveraged by classical learning mechanisms. The design of the quanvolutional filter provides flexibility in selecting its components. For instance, the user can modify the encoding and decoding functions, as well as the sets of single- and two-qubit quantum gates employed to construct the random quantum circuit, among other elements.

3. Implementation Details and Experimental Setup

To empirically evaluate the performance of the Quanvolutional Neural Network, we established a comprehensive experimental framework using a combination of quantum and classical machine learning libraries, namely *PyTorch* [2] and *Pennylane* [3]. All the code is written in Python, with results systematically logged using the Weights & Biases platform for reproducibility and analysis [4]. A complete list of experimental results, together with detailed logs, can be accessed at *wandb.ai/jaysenoner/quanvolutional-nn-mnist*.

3.1. Quanvolutional Layer

The core of this project is the quanvolutional layer, which was implemented using the *PennyLane* library for quantum machine learning [3]. This layer serves as a fixed feature extractor, applying a quantum transformation to local patches of an input image.

Architecture: We chose a 4-qubit quantum circuit, designed to process 2×2 pixel patches from the input images. The layer scans the image with a stride of 2, which effectively downsamples a 28×28 image into four distinct 14×14 feature maps. The implementation of this core logic is shown in Listing 1. The quantum simulation was

performed using *PennyLane*'s high-performance *lightning.qubit* CPU-based simulator.

```
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
```

Listing 1: The preprocessing step extracts non-overlapping 2×2 patches from the input image and flattens each patch before passing them into the quantum circuit. The outputs of the circuit are subsequently stored in the corresponding position of the output tensor, resulting in a feature map of dimensions $14 \times 14 \times 4$.

Operational Steps: The transformation process for each patch involves three key stages:

1. **Encoding:** The four classical pixel values ϕ_j from a flattened 2×2 patch, scaled to the range $[0, 1]$, are encoded into the quantum state. This is achieved by controlling the rotation angle of a single-qubit Y-rotation gate, $R_Y(\pi \cdot \phi_j)$, applied to each of the four qubits, which are initialized in the ground state $|0\rangle$.

2. **Transformation:** A random quantum circuit is applied to the encoded state. To ensure this layer acts as a consistent filter across the entire image, the circuit's parameters are generated once with a fixed random seed and remain constant throughout the preprocessing. We utilized *PennyLane*'s *RandomLayers* template, experimenting with both 1-layer and 2-layer configurations to study the impact of circuit depth.

3. **Measurement:** To extract classical information, the expectation value of the Pauli-Z operator is measured for each of the four qubits. This measurement in the computational basis yields a 4-dimensional real-valued vector, which forms the channels of the corresponding output pixel.

The complete implementation of this quantum circuit, expressed as a PennyLane QNode, is shown in Listing 2.

```
n_qubits = 4
n_layers = 1 # or 2

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, interface="torch")
def quanv_circuit(phi):
    for j in range(n_qubits):
        qml.RY(np.pi * phi[j], wires=j)
    RandomLayers(rand_params,
        wires=list(range(n_qubits)))

    return [qml.expval(qml.PauliZ(j)) for j in
        range(n_qubits)]
```

Listing 2: The core 4-qubit quanvolutional circuit implemented in PennyLane. The circuit uses a fixed random seed to act as a deterministic feature extractor.

3.2. Hybrid and Baseline Model Architectures

The classical components of our models were implemented in PyTorch. We designed two main architectures to facilitate a direct and fair comparison.

Quanvolutional Neural Network (QNN): This is our primary hybrid model. It uses the preprocessed feature maps generated by the quanvolutional layer as its input. The backend is a deep classical CNN consisting of two convolutional blocks (Conv2D $\rightarrow$ ReLU $\rightarrow$ MaxPool2D) followed by two fully connected layers with dropout for stochastic regularization.

Classical CNN Baseline: To properly assess the utility of the quanvolutional layer, we designed a purely classical baseline model. The first layer of this model is a standard *nn.Conv2d* layer carefully configured to be a direct analogue of the quanvolutional layer: it uses a 2×2 kernel with a stride of 2, taking a 1-channel image and producing 4 output feature maps. The subsequent deep architecture of the classical baseline is **identical** to that of the QNN, ensuring that any performance difference can be primarily attributed to the initial feature extraction stage.

Minimal Classifier: To conduct an ablation study

and evaluate the "raw power" of the quantum features themselves, we designed a *MinimalClassifier*. This is a very shallow model consisting only of a *nn.Flatten* layer followed by a single *nn.Linear* layer for classification. By training this minimal model on the feature maps produced by both the quanvolutional layer and the classical feature extractor, we can directly compare the inherent quality and separability of the features each extractor produces, without the confounding influence of a powerful deep learning backend. [5]

The trainable parameter counts for all tested models are detailed in Table 1.

Model Architecture	Trainable Parameters
Quanv-CNN (QNN)	169,962
Classical-CNN	169,982
Minimal-Classifier	7,850

Table 1: Number of trainable parameters for the main models. The parameter counts of convolutional models are deliberately kept almost identical to ensure a fair comparison of the feature extractors.

3.3. Datasets and Preprocessing Workflow

We evaluated our models on four diverse image classification datasets to test the generalization of our findings: MNIST, Fashion-MNIST (FMNIST), KMNIST, and CIFAR-10.

All images were reshaped and scaled to 28×28 (for CIFAR-10, we converted images to grayscale).

Since all MNIST-like datasets consist of images with a resolution of 28×28, the CIFAR-10 images were center-cropped from their original 32×32 resolution to 28×28 in order to ensure architectural compatibility across all evaluated datasets. Furthermore, the channel dimension was reduced to one by applying a grayscale transformation.

A key aspect of our methodology was the preprocessing workflow for the QNN. As the quanvolutional layer is non-trainable, we applied it as a one-time preprocessing step to the entire training and test sets for each dataset and for each quantum circuit configuration (1-layer and 2-layer). The resulting feature maps were saved to disk as *PyTorch* tensor files ('.pt'). While this is efficient for subsequent training runs, the initial quantum preprocessing step is computationally intensive, taking approximately 8 hours for a training set of 60,000 images on a modern CPU (Intel(R) Core(TM) i7- 10700K CPU @ 3.80GHz). This highlights a significant practical challenge for the simulation of quantum machine learning models.

4. Experiments and Results

To provide a comprehensive evaluation of the Quanvolutional Neural Network, we conducted a series of five key experiments. First, we established the baseline performance of all models on the complete, clean datasets. Subsequently, we examined more specific aspects of the models by assessing data efficiency, robustness to noise, the quality of the learned feature representations, and the intrinsic capacity of the feature extractors through an ablation study.

4.1. Model Performance on Full Datasets

The first experiment establishes the baseline performance of our primary models. We trained the Quanvolutional Neural Network with both 1 and 2 quantum layers and the Classical CNN baseline on 100% of the training data for each of the four datasets: MNIST, FMNIST, KMNIST, and the more challenging CIFAR-10 (converted to grayscale). The models were then evaluated on their respective clean test sets. The final test accuracies are summarized in Table 2.

Dataset	CNN Acc (%)	QNN (1-Layer) Acc (%)	QNN (2-Layer) Acc (%)
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

Table 2: Final test accuracy on full, clean datasets. The classical baseline consistently achieves the highest accuracy, though the margin is small and could be due to training variance. All models struggle with the complexity of CIFAR-10.

The results from these training runs suggest that, under ideal data conditions (full, clean

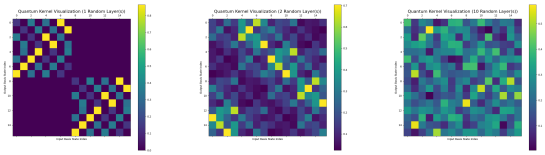


Figure 2: Visualization of the absolute value of the 16x16 unitary matrix for the 1-layer, 2-layer and 10-layer quantum kernels. The 1-layer kernel is visibly more sparse, while the 2- and 10-layer kernel are significantly denser, indicating a more chaotic transformation.

datasets), the fixed random quantum kernel generally does not provide a clear advantage over a trainable classical convolutional layer. For the MNIST, FMNIST, and KMNIST datasets, the classical baseline achieves the highest accuracy. When considering the significant computational overhead required for the quantum preprocessing step (approximately 8 hours per training set, plus 1.5 hours per test set), the classical approach is the more practical and effective choice for these tasks. A notable exception, however, occurs on the more challenging CIFAR-10 dataset, where the 1-layer QNN model achieves the best performance, albeit by a small margin. This result suggests that the features generated by the quantum circuit may offer some utility on more complex image distributions where simple, learned filters might be less optimal. Furthermore, a consistent trend across three of the four datasets is the underperformance of the 2-layer QNN compared to its 1-layer counterpart. This indicates that simply increasing the depth and complexity of the random quantum circuit does not necessarily improve performance and can, in fact, be detrimental, possibly by creating feature maps that are too chaotic (Figure 2). This motivates the subsequent experiments to determine if there are other conditions where the QNN might hold an advantage. [1]

[1]For the sake of clarity and readability, the complete training and validation loss curves are omitted from this report; however, they are available on the Weights & Biases platform.

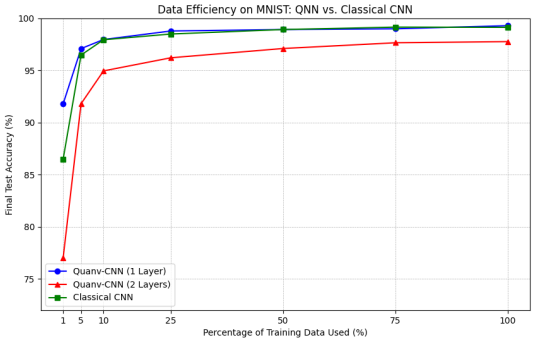


Figure 3: Data Efficiency on MNIST

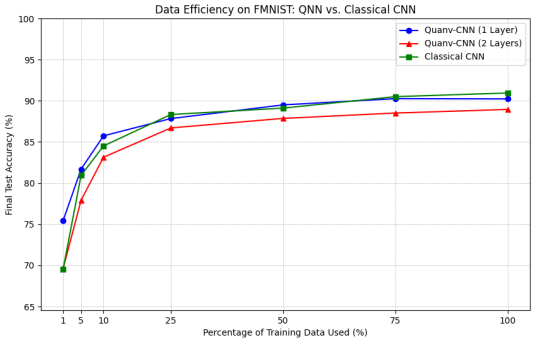


Figure 4: Data Efficiency on FMNIST

4.2. Data Efficiency

To investigate the models' performance in data-limited scenarios, we conducted a data efficiency experiment. For this, we trained each model on random subsets of the training data, ranging from 1% to 100%, and recorded the final test accuracy for each configuration. The objective of this experiment is to determine whether the quantum features are sufficiently powerful and informative to yield a performance advantage over classical feature maps in the low-data regime. The results across all four datasets are presented in Figures 3 - 6.

The data efficiency experiment reveals one of the most compelling advantages of the quanvolutional approach On the MNIST and FMNIST datasets (Figures 3 and 4), the 1-layer QNN demonstrates a significant performance lead over the classical baseline when trained on 10% or less

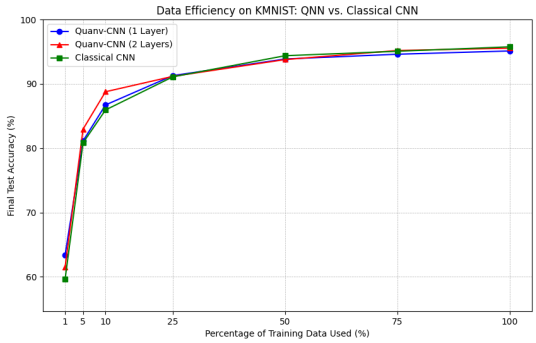


Figure 5: Data Efficiency on KMNIST

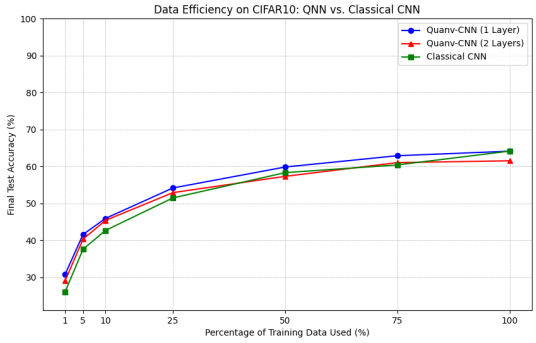


Figure 6: Data Efficiency on CIFAR-10 (Grayscale and Center-cropped)

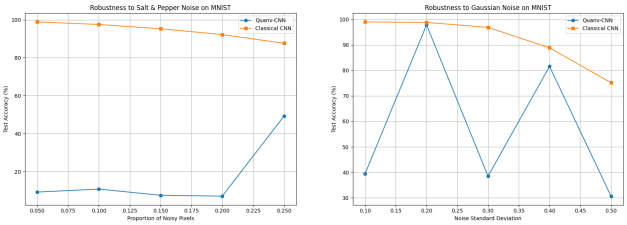
of the data. This strongly suggests that the quantum circuit's transformation provides a powerful inductive bias. Its fixed, complex feature mapping creates a high-quality representation immediately, allowing the classical backend to generalize effectively from very few examples. The classical model, which must learn its own filters from scratch, requires a larger volume of data to achieve a comparable representation. The KMNIST dataset (Figure 5) shows a similar but less pronounced trend, with all models performing very closely, though the QNNs still maintain a slight edge at the lowest data percentages. Most notably, on the more complex CIFAR-10 dataset (Figure 6), the advantage of the 1-layer QNN is sustained across the *entire* range of data percentages. Even with 100% of the training data, it outperforms the classical baseline. This is a significant finding, suggesting that for more complex image distributions, the features produced by the quanvolutional kernel may be inherently more powerful than the simple 2×2 filters learned by the classical analogue. Across almost all scenarios, except for KMNIST in the low data regime, the 2-layer QNN (red curve) underperforms the 1-layer version. This reinforces our earlier finding that increasing the complexity of the random quantum circuit is not guaranteed to improve, and can even hinder, overall model performance.

4.3. Robustness to Input Noise

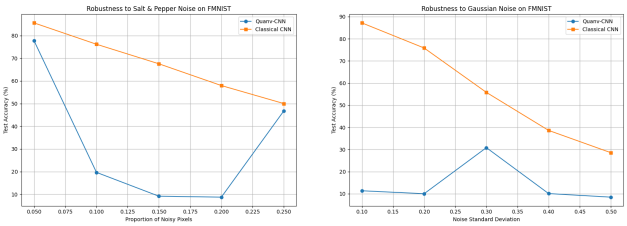
The objective of this experiment is to evaluate the performance of the QNN architecture in classifying images corrupted with varying types and levels of input noise.

Classical convolutional networks are known to be sensitive to input noise. When test images are corrupted, their performance typically degrades as a function of both the magnitude and the type of noise applied. In this experiment, we aim to investigate whether quantum features confer any degree of resilience to input noise by comparing the QNN architecture against our baseline classical network. To test this, we evaluated the fully pre-trained models on test sets corrupted with varying levels of two noise types: Gaussian (continuous perturbation) and Salt & Pepper (sparse, extreme errors). The noise was applied to the original images before they were passed to either the classical or quanvolutional feature extractor. The results of this experiment are shown in Figures 7 and 8. **Salt & Pepper Noise:** The performance of the classical model degrades gradually as the proportion of noisy pixels increases. In contrast, the QNN exhibits extreme fragility. On MNIST, FMNIST, and KMNIST, its accuracy collapses to near-random levels even at low noise intensities (5–10%). We hypothesize that this brittleness arises from the sensitive $R_Y(\pi \cdot \phi_j)$ encoding combined with the use of a fixed, non-trainable kernel. A single extreme pixel value (e.g., 1.0 from a "salt" pixel) induces a maximal qubit rotation, thereby altering the quantum input state in a drastic manner. By contrast, a trainable classical filter can adapt its weights to

suppress such sparse outliers, whereas the fixed QNN lacks this mechanism and is therefore easily overwhelmed. An interesting anomaly emerges at the highest noise level (25%) on the MNIST-like datasets, where the QNN's performance unexpectedly improves. This could suggest a complex interaction in which extreme noise may randomly map inputs into a region of the feature space that is, albeit incorrectly, more separable. **Gaussian Noise:** Under Gaussian noise, the classical model again demonstrates predictable degradation. The QNN, however, behaves not only worse but also in a highly erratic and unstable manner. On MNIST and KMNIST, the QNN's accuracy is initially low but then exhibits sudden spikes at specific noise levels (standard deviation 0.2 for MNIST, 0.3 for KMNIST), before sharply declining again. This behavior is unlikely to reflect genuine robustness; rather, it is plausibly an artifact of the non-linear encoding. For certain noise distributions, the perturbed pixel values may be mapped by the R_Y gate into regions of the feature space that are, by coincidence, more separable. Such fragile alignments do not occur in the more complex FMNIST and CIFAR-10 datasets, where the QNN's performance remains consistently and significantly below the classical baseline. Taken together, these results provide strong evidence that, while the QNN architecture can be effective on clean, limited data, it is not inherently robust. Its fixed and highly sensitive nature makes it particularly vulnerable to input perturbations, representing a substantial disadvantage compared to trainable and adaptive classical models, whose performance degradation under increasing noise levels is more stable and predictable.

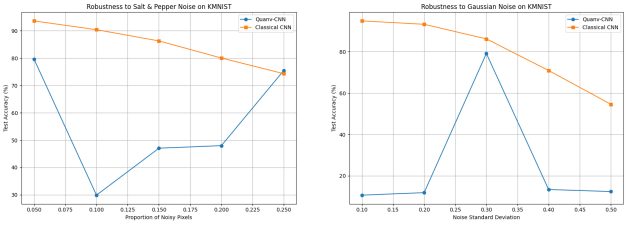


(a) MNIST (Salt & Pepper) (b) MNIST (Gaussian)

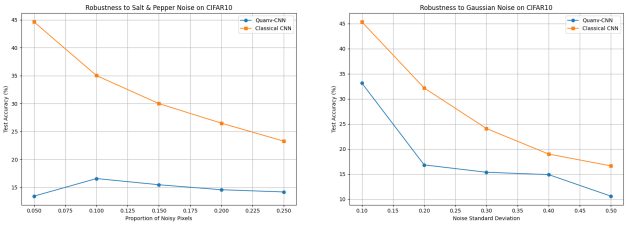


(c) FMNIST (Salt & Pepper) (d) FMNIST (Gaussian)

Figure 7: Noise robustness on MNIST and FMNIST datasets. Each row compares Salt & Pepper vs Gaussian noise.



(a) KMNIST (Salt & Pepper) (b) KMNIST (Gaussian)



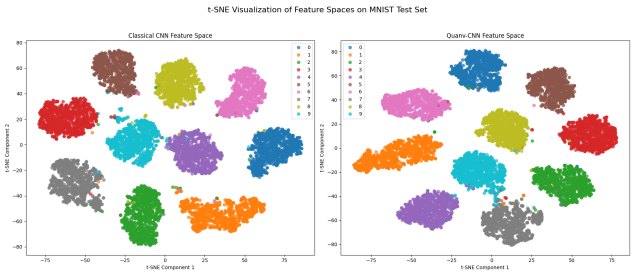
(c) CIFAR-10 (Salt & Pepper) (d) CIFAR-10 (Gaussian)

Figure 8: Noise robustness on KMNIST and CIFAR-10 datasets. Each row compares Salt & Pepper vs Gaussian noise.

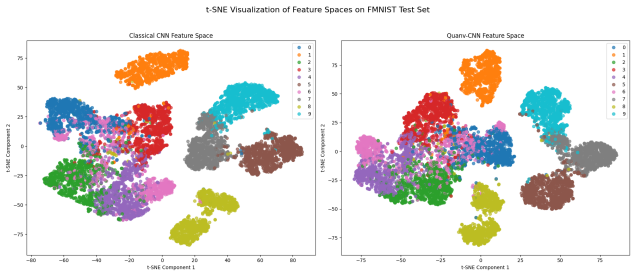
4.4. Analysis of Feature Representation and Error Patterns

To gain a deeper, qualitative understanding of the models' behavior on clean data, we analyzed the feature spaces they produce and the specific error patterns they exhibit. For this, we used the t-SNE dimensionality reduction technique to visualize the feature vectors from the models' last hidden layer, and we computed confusion matrices to identify which (if any) classes are more missclassified then others.

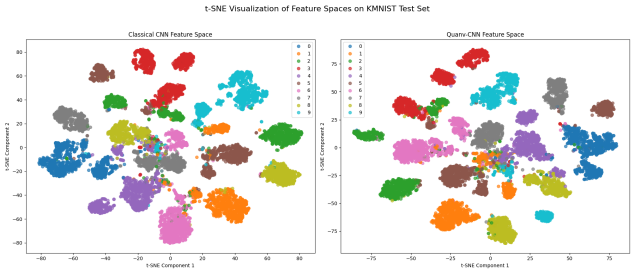
Feature space visualization (t-SNE) The t-SNE plots in Figure 9 provide a compelling visual explanation for the QNN's strong data efficiency. On the MNIST and KMNIST datasets, the QNN feature space exhibits significantly better class



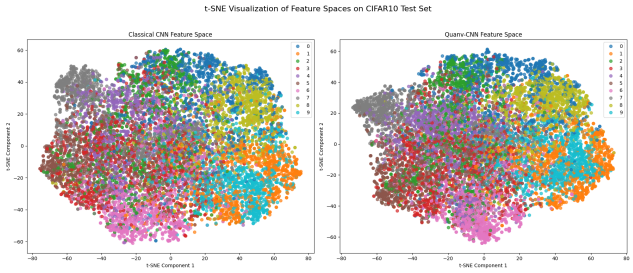
(a) t-SNE Visualization on MNIST



(b) t-SNE Visualization on FMNIST



(c) t-SNE Visualization on KMNIST



(d) t-SNE Visualization on CIFAR-10

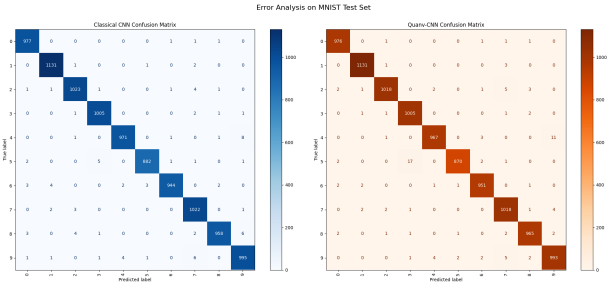
Figure 9: t-SNE visualization of feature spaces for the Classical CNN (left of each pair) and QNN (right of each pair). The QNN produces visibly tighter clusters on MNIST and KMNIST. Both models struggle with the complexity of FMNIST and CIFAR-10.

separation, with tighter and more distinct clusters than the classical baseline. This highly structured representation makes the final classification task easier for the downstream layers, which explains why the model can generalize so well from few examples. On the more challenging FMNIST dataset, the advantage is less pronounced, as both models struggle to separate semantically similar classes like ”Shirt,” ”T-shirt,” and ”Coat.” On CIFAR-10, both feature spaces show significant class overlap, highlighting the limitations of the simple 4-qubit grayscale feature extractor for complex, real-world images.

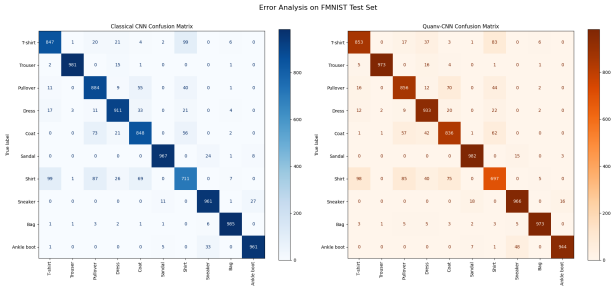
Confusion Matrices The confusion matrices in Figure 10 allow for a microscopic analysis of the models' mistakes. On MNIST, both models are near-perfect, with very few off-diagonal errors, confirming their high overall accuracy. The analysis becomes more insightful on FMNIST, where both models exhibit significant confusion between classes with similar visual features, such as ”Shirt,” ”T-shirt,” ”Pullover,” and ”Coat.” This indicates that while the QNN's feature space is well-structured, it is not immune to the same semantic ambiguities that challenge the classical model. On KMNIST and CIFAR-10, the matrices become progressively denser off the diagonal for both architectures, reflecting the increasing difficulty of the classification tasks. In general, the error patterns are remarkably similar between the two models, suggesting they learn comparable decision boundaries, even if the underlying features are different.

4.5. Architectural Ablation

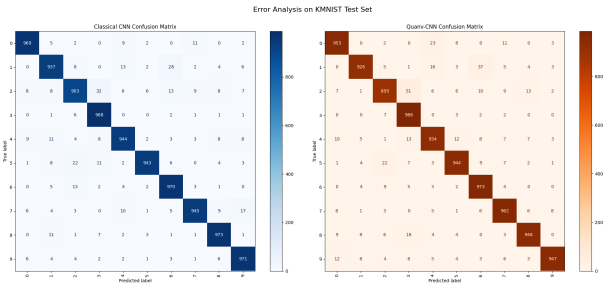
To isolate the ”raw power” of the features themselves, we performed an architectural ablation study. The goal was to understand how effective the features produced by the quantum and classical extractors are before being processed by a deep convolutional network. For this, we trained a **MinimalClassifier**—a model consisting of only a single linear layer—directly on the feature maps produced by both the quanvolutional layer and the



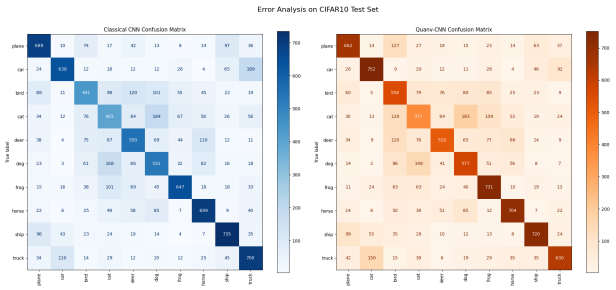
(a) Confusion Matrix on MNIST



(b) Confusion Matrix on FMNIST



(c) Confusion Matrix on KMNIST



(d) Confusion Matrix on CIFAR-10

Figure 10: Confusion matrices for the Classical CNN (left of each pair) and QNN (right of each pair) on the test sets. The off-diagonal entries represent misclassifications.

classical analogue layer. This removes the powerful pattern-recognition capabilities of the deep CNN, allowing for a more direct comparison of the feature quality.

Dataset	Classical Acc (%)	Quantum Acc (%)
MNIST	92.57	**93.62**
FMNIST	84.29	**85.08**
KMNIST	70.30	**73.60**
CIFAR-10	29.64	**33.19**

Table 3: Test accuracy using a minimal classifier (single linear layer) trained on features from the classical and quantum extractors. The best performance for each dataset is highlighted in bold.

The results of the ablation study, presented in Table 3, are definitive and consistent across all four datasets. The features produced by the quantum extractor are significantly more effective for direct, shallow classification than those from the analogous classical layer. The performance gap is substantial, particularly on the more challenging KMNIST and CIFAR-10 datasets.

This experiment provides two crucial insights. First, it strongly reinforces the findings from the t-SNE analysis (Section 4.4). The reason the minimal model performs better with quantum features is that those features are inherently more linearly separable, as was visually suggested by the tighter t-SNE clusters. Second, it demonstrates that the quantum processing is not merely preparing data for a deep network; it is actively producing high-level, discriminative features that are already well-suited for the classification task. This inherent quality of the features is the primary driver of the QNN's strong performance in data-limited scenarios, as a powerful representation is available even before training of the deep backend.

5. Conclusions

This project successfully implemented a Quanvolutional Neural Network and conducted a rigorous comparison against a classical baseline. Our findings present a compelling and nuanced trade-off. On one hand, the QNN demonstrates a remarkable ability to learn from limited data, a

quality explained by the highly structured feature space it generates from clean inputs. On the other hand, this structured mapping proves to be exceptionally brittle, demonstrating a catastrophic failure in performance when confronted with noise in the classical input data—a scenario where the trainable classical baseline is far more robust. The results underscore that the Quanvolutional Neural Network is a flexible framework, where overall performance is highly dependent on the careful selection of its components, particularly the data encoding scheme. The observed fragility to input noise appears to be a direct consequence of the sensitive encoding method used, rather than a fundamental flaw of the quantum processing itself. Interestingly, this observed brittleness to classical input noise stands in contrast to a key theoretical advantage proposed for QNNs: a potential resilience to quantum hardware noise. As noted by Henderson et al., since errors on NISQ-era devices can be modeled as unknown, random gates being added to a circuit, a feature extractor that is already based on a random circuit may not suffer a significant degradation in quality [1]. While our simulation-based experiments could not verify this hypothesis, it remains a compelling motivation for exploring such architectures on real quantum hardware. The primary practical challenge encountered was the large computational cost of quantum circuit simulation, with preprocessing for a simple, academic dataset like MNIST requiring approximately 8 hours. These challenges and findings point towards several key directions for future research. A key progress would be the exploration of trainable, or "variational," quantum layers using parametrized gates. A learnable quantum kernel could potentially adapt to the statistics of the data, learning to be robust to classical input noise while retaining the powerful feature extraction capabilities.

References

[1] Maxwell Henderson, Samriddhi Shakya, Shashindra Pradhan, and Tristan Cook. Quanvolutional neural networks: Powering image recognition with quantum circuits. 2019.

[2] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[3] Ville Bergholm et.al. Pennylane: Automatic differentiation of hybrid quantum-classical computations, 2022.

[4] Lukas Biewald. Experiment tracking with weights and biases, 2020. Software available from wandb.com.

[5] C. M. Wilson, J. S. Otterbach, N. Tezak, R. S. Smith, A. M. Polloreno, Peter J. Karalekas, S. Heidel, M. Sohaib Alam, G. E. Crooks, and M. P. da Silva. Quantum kitchen sinks: An algorithm for machine learning on near-term quantum computers, 2019.