SELECTIVE QUANTUM ROUTING: A DISTRIBUTED HYBRID NEURAL NETWORK

Henry Robb
Department of Computer Science
Rensselaer Polytechnic Institute
Troy, NY, 12180, USA
robbh@rpi.edu

1 INTRODUCTION

The field of machine learning has trended towards huge architectures built for complicated and general tasks, this is epitomized by the rise of LLMs. However, foundational challenges such as image classification remain important benchmarks for algorithmic efficiency. Deep networks, especially Convolutional Neural Networks (CNNs), have come to be regarded as the standard in the field. When excluding massive models that are relying on transfer learning from external backbones, CNNs consistently achieve field leading performance on benchmark datasets like MNIST and Fashion MNIST [1]. Despite this excellence, classical CNNs are approaching an asymptote when it comes to their capacity to efficiently capture and compute complex feature spaces [1].

Quantum Machine Learning (QML), and more specifically Quantum Convolutional Neural Networks (QCNNs), are offering a solution to this problem. By utilizing the huge capacity of the Hilbert space, these representation challenges can be overcome [4] [6]. Current approaches recognize this potential quantum advantage, however, their approaches to developing a proper hybrid network are flawed. Many proposed architectures simply add a quantum layer to their neural network that forces all of the features through a quantum circuit [2] [5] [6]. This practice in the Noisy Intermediate Scale Quantum (NISQ) era, where noise is prevalent and Quantum Processing Units (QPUs) are scarce, is suboptimal. This methodology is computationally wasteful and is prone to errors introduced by unnecessary noise.

To address this inefficiency, we propose a novel, dynamic hybrid routing architecture. Rather than simply forcing all of the features to be pushed through a quantum layer to the benefit of some and the detriment of others, our model introduces a trainable router that will evaluate and actively sort the feature representations of an image to be sent to either a classical processor or a QPU for processing. This will safeguard the limited quantum resources from unneeded tasks and only allow the features that will benefit from this processing to be sent to a QPU. Additionally, because the extreme costs of quantum access have created a lack of benchmarks trained on real quantum machines, this project will take the best designs found in simulation and evaluate them on real quantum hardware.

The rest of the paper will continue as follows: Section 2 will review related literature within the field of image classification and QCNNs. Additionally, it will present recent hardware benchmarking and hybrid multiclass classification studies. Section 3 will formulate the core problem and introduce the mathematical and quantum theorems required to best understand the proposed solution. Section 4 details the methodology of the dynamic router and the hybrid pipeline. Section 5 will allow for a better understanding of the evaluation setup and then present the results. Finally, Section 6 will provide a cohesive conclusion for the results and their contributions to the field.

2 RELATED WORK

2.1 FOUNDATIONAL QCNN ARCHITECTURES AND FEATURE EXTRACTION

Early developments in QML established some foundational architectures for combining quantum circuits with the more established classical deep learning tasks. Quantum circuits were focused on especially when it came to feature extraction. A critical part of the foundation for our task was proposed by Lu et al., which showed how parameterized quantum circuits could serve a similar role

to a classical convolutional structure and extract features from images[4]. Yang et al. built upon this idea and proved that QCNNs can serve as capable feature extractors across multiple domains, in their case specifically automatic speech recognition[6]. While these studies were critical for ratifying the viability of QCNNs, their architectures were static. They force the entire data representation to be processed by a quantum circuit, irregardless of whether a specific component actually warranted the complexity that quantum inherently introduces. Rather than relying on a single quantum circuit for extraction, our pipeline uses a classical convolutional funnel followed by a dynamic router. This ensures that only specific features that require detailed processing are sent to the QPU, therefore minimizing the use of expensive quantum computational resources.

2.2 Hybrid Dimensionality Reduction for Multi Class Classification

As QCNNs were being applied to ever more complex multi class datasets like Fashion MNIST, researchers started to encounter scalability issues. This was especially evident in the barren plateau problem which appears in high dimension feature spaces. Researchers have addressed this problem within their hybrid classical quantum pipelines. Bokhan et al. demonstrated that it was necessary to heavily scale down classical feature before encoding them into qubits[2]. This is because there is a limited of qubits available. Shi et al. highlighted issues with training quantum models on datasets like Fashion MNIST without having a classical convolutional backbone to manage the dimensionality[5]. These works efficiently mitigated barren plateaus through classical preprocessing. Our work, rather than relying on static and predetermined dimensionality reduction, couples a classical step down projection with a dynamic router. This allows the model to learn on the fly which of the compressed features are best suited to being mapped to quantum angle embeddings.

2.3 Hardware Benchmarking and Hybrid Training in the NISQ Era

The final area of related literature covers both the critical transition from ideal simulations to the noisy physical hardware and the advanced techniques required to train the complex hybrid systems we have today. The study by Lakhdar-Hamina et al. showed the decisive performance differences between simulated quantum environments and noisy QPUs[3]. Additionally, our proposed architecture builds upon their contributions by adapting their Straight-Through Estimator (STE) methodology. Our dynamic router must make discrete, binary decisions. It can only route a feature to a classical or quantum path. Therefore, the gate would normally be non-differentiable and standard backpropagation would fail. Instead, by using STE, we are able to approximate the gradients through the step function, and this unlocks training of the router, and therefore, the entire pipeline. Their benchmarking studies provide important insights into hardware specific noise consequences, but these are done on unrouted quantum circuits. They have developed the closest concept to our router in a parameter that can tune between quantum and classical modes, yet falls short in creating a truly dynamic architecture.

3 Problem Formulation

This section will clearly define the multi class image classification problem and introduces the mathematical algorithms and theoretical frameworks that support the proposed hybrid architecture

3.1 Image Classification Task

We consider a supervised multi class image classification problem defined over a dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$, where $\mathbf{x}_i \in \mathbb{R}^{1\times 28\times 28}$ represents a two dimensional grayscale image input and $y_i \in \{1, 2, \ldots, C\}$ represent the corresponding class label. In the context of this project, the input space corresponds to the Fashion MNIST dataset. The problem is evaluated under two target spaces. The first is $C = 10$ which represents the entire dataset, and the second is $C = 5$ where the 5 classes that represent the subset are the most similar.

The overall goal is to learn a function $F_\Theta : \mathbb{R}^{1\times 28\times 28} \to \mathbb{R}^C$ that outputs a vector of classes $\mathbf{z}$. The network is optimized by minimizing the standard cross-entropy loss function $\mathcal{L}$:

$$\mathcal{L}(\Theta) = -\sum_{i=1}^{N}\sum_{c=1}^{C} y_{i,c} \log\left(\frac{\exp(z_{i,c})}{\sum_{j=1}^{C}\exp(z_{i,j})}\right) \tag{1}$$

where $y_{i,c}$ is the indicator of whether class c is the correct label for observation i.

3.2 Classical Feature Extraction

A core component of the project involves evaluating the dynamic routing behavior across different feature space dimensions. We define a classical convolutional funnel as E_{θ_e} parameterized by classical weights θ_e. This maps the high dimensional image to a dense 1D vector $\mathbf{v} \in \mathbb{R}^d$

$$\mathbf{v} = E_{\theta_e}(\mathbf{x}) \tag{2}$$

We evaluate the architecture under two distinct extraction methods:

- **Uncompressed Baseline ($d = 128$):** This more standard feature extractor creates a high dimensional space. In this environment, mapping $\mathbb{R}^{128}$ to a quantum circuit introduces a plethora of scalability issues. This often leads to the router to bypass the quantum path to avoid barren plateaus.
- **Compressed Architecture ($d = 8$):** To mitigate information loss while also attempting to establish a computational baseline, we apply a step down project layer. By constraining the dimension to $d = 8$, both the quantum and the classical path are forced to operate on an information dense feature space

3.3 Dynamic Feature Wise Routing and the STE

The novel part of the proposed architecture is a dynamic, trainable routing mechanism. Rather than routing an entire image, the router evaluates the extracted vector $\mathbf{v}$ and makes feature wise decisions. It assigns specific components to either a classical feed forward network, C_{θ_c}, or a quantum variational circuit, Q_{θ_q}.

Let the router be defined as a parameterized gating function $R_{\theta_r}(\mathbf{v})$ that outputs a continuous vector $\mathbf{g} \in \mathbb{R}^d$. To enforce a hard routing decision, we use a sign function to generate a binary mask vector $\mathbf{m} \in \{-1, 1\}^d$.

$$\mathbf{m} = \text{sgn}(\mathbf{g}) \tag{3}$$

Using this mask, the features are split into purely classical and purely quantum subsets, which are processed independently, then unified after:

$$F_\Theta(\mathbf{x}) = U_{\theta_u}\left(Q_{\theta_q}\left(\mathbf{v}\odot\frac{1+\mathbf{m}}{2}\right) + C_{\theta_c}\left(\mathbf{v}\odot\frac{1-\mathbf{m}}{2}\right)\right) \tag{4}$$

where $U_{\theta_u} : \mathbb{R}^d \to \mathbb{R}^C$ represents the final unified classifier. Because the derivative of the sign function is zero almost everywhere, standard backpropagation would fail. This would make the router θ_r untrainable. Therefore, the STE is used. During the forward pass, the discrete mask $\mathbf{m}$ is applied as defined above. During the backward pass, the STE approximates the gradient of the sign function as a Hard Tanh function:

$$\nabla_{\mathbf{g}}\mathcal{L} \approx \nabla_{\mathbf{m}}\mathcal{L} \tag{5}$$

This modification allows gradients to flow back into the routing layer and this enables optimization of the attention gate.

3.4 Quantum Angle Encoding and Variational Ansatz

When features are routed to the quantum path, $\mathbf{m}_j = 1$, the masked vector needs to be translated into a quantum state. We define a quantum system of $n = 4$ qubits.

Dimensional Translation The masked d-dimensional vector, where $d \in \{8, 128\}$, is first projected down to match the qubit count via a linear weight matrix W_{down}. This creates $\mathbf{h} \in \mathbb{R}^n$. The features are then normalized using a scaled hyperbolic tangent activation to ensure validity for rotation gates.

$$\phi = \tanh(\mathbf{h}) \cdot \pi \tag{6}$$

Angle Embedding and VQC: The normalized features are then encoded into a quantum state using R_x rotation gates. The state is processed by the Variational Quantum Circuit (VQC) on which we employ a data re-uploading strategy. The feature encoding state preparation, defined as $S_x(\phi) = \bigotimes_{j=1}^{n} R_x(\phi_j)$, is interleaved with trainable unitary layers $U_l(\theta_l)$. These unitary layers are comprised of single qubit rotations, and multi qubit entangling gates. For a circuit with L layers, the final state would be given by:

$$|\psi_{out}\rangle = S_x(\phi) \left(\prod_{l=1}^{L} U_l(\theta_l) S_x(\phi) \right) |0\rangle \tag{7}$$

Measurement: Then, to extract classical predictions from the quantum system, we need to measure the values of the Pauli-Z observables O_k across the qubits. This results in a quantum output vector $\mathbf{z}_q \in \mathbb{R}^n$:

$$z_{q,k} = \langle \psi_{out} | O_k | \psi_{out} \rangle \tag{8}$$

Finally, $\mathbf{z}_q$ is projected back up to the d-dimensional space using a up project matrix W_{up}. This completes the quantum pipeline and prepares the features for a unification with the classical path.

4 METHOD

To best evaluate the proposed routing architecture, a highly modular five stage pipeline was designed using PyTorch and PennyLane. The modularity allowed for an extremely rigorous ablation study in which the effects of feature dimensionality, dataset complexity, routing strategies, and hardware usages were analyzed.

4.1 DATASET PREPARATION

The experiment utilized the Fashion MNIST dataset. This dataset is comprised of 28×28 grayscale images. To test how well the architecture could separate very similar classes, two classification tasks were created:

- **Full Dataset ($C = 10$):** This was the standard benchmark using all ten of the distinct clothing classes (denoted as **c10**).
- **Restricted Subset ($C = 5$):** This was a highly specific subset consisting of garments that were all visually similar. T-shirt/top, Pullover, Dress, Coat, and Shirt comprised this subset (denoted as **c5**) and they forced the feature extractor to rely on specific details, rather than overall shape, because of their similarity.

4.2 ARCHITECTURAL MODULARITY

The network was divided into interchangeable classes to allow for controlled testing across several different experimental states.

- **Feature Dimensionality ($d \in \{8, 128\}$):** The initial CNN feature extractor used $32 \to 64$ convolutional channels mapping to a fully connected output of $d = 128$. To implement the compressed architecture, we deployed a thinner funnel. This funnel used $16 \to 16$ convolutional channels with a step down layer in order to map to $d = 8$.
- **Routing Strategies:** We implemented three distinct routing modules. The *Static Classical Router* (**q0**) forced a strict -1.0 mask across all features. The *Static Quantum Router* (**q100**) forced a strict 1.0 mask. Finally, the *Dynamic Router* (**qD**) utilized the STE to dynamically route features based on learned weights. To ensure training stability, the dynamic

router's bias was initialized to conservatively favor the classical path during early epochs, and gradients flowing through the STE were scaled by a factor of 0.1 to prevent erratic routing oscillations.

Algorithm 1 Dynamic Feature Routing with Straight-Through Estimator (STE)

Input: Extracted classical feature vector $\mathbf{v} \in \mathbb{R}^d$
Forward Pass:
  Calculate pre-activation weights: $\mathbf{g} \leftarrow R_{\theta_r}(\mathbf{v})$
  Apply hard step function: $\mathbf{m} \leftarrow \text{sgn}(\mathbf{g}) \in \{-1, 1\}^d$
  Route features via masking:
    $\mathbf{v}_{quantum} \leftarrow \mathbf{v} \odot \frac{1+\mathbf{m}}{2}$
    $\mathbf{v}_{classical} \leftarrow \mathbf{v} \odot \frac{1-\mathbf{m}}{2}$
Backward Pass:
  Approximate gradient: $\nabla_{\mathbf{g}}\mathcal{L} \approx \nabla_{\mathbf{m}}\mathcal{L}$ ▷ Bypasses zero-gradient of sign function
  Update router weights θ_r via backpropagation

- **Preprocessing and Unification:** Depending on the router's mask, features were either processed through a classical Feed Forward Network (C_{θ_c}) or a 4 qubit, 2 layer Variational Quantum Circuit (Q_{θ_q}). The outputs were then summed and passed through a final linear classifier.

4.3 Ablation Matrix and Training Protocol

By combining all the aforementioned modules, we generated a thorough experimental matrix. Models are denoted by their configuration, for example, *compressed_c5qD* represents the 8 feature, 5 class, dynamically routed model.

All of the models were trained using the Adam optimizer with a learning rate of 0.001, a Step LR scheduler (step size 5, gamma 0.5) and Cross Entropy Loss. To track the evolution of the router's results, behavior was run for 45 epochs. Additionally, checkpoints for model weights and metric logs containing loss, accuracy, and quantum routing percentages were saved at 15, 30, and 45 epoch intervals.

5 Evaluation

To assess the proposed architecture, the evaluations were split into a simulated benchmarking phase, and a physical hardware execution phase.

5.1 Simulated Environment

All of the simulated experiments were run on PennyLane's `default.qubit` state vector simulator that easily integrated with PyTorch. The environment allowed for a quick and noiseless training of the ablation matrix. During simulation, 3 metrics were logged per epoch: standard Cross Entropy Loss, validation accuracy, and the Quantum Routing Percentage, the proportion of features actively being sent down the quantum pipeline by the attention gate. Additionally, to visualize the dynamic router's decision making, Captum's Occlusion tool was used to create spatial heatmaps. This allowed a visualization of which pixels were triggering a quantum execution.

5.2 Inference Testing on Hardware

To validate our simulated findings on machine with real quantum noise, the highest performing models were deployed to physical quantum hardware. Using the `qiskit-ibm-runtime` service, the trained PyTorch weights were loaded and evaluated on the IBM Quantum System One at Rensselaer Polytechnic Institute (RPI). This physical benchmarking tested how well the learned angle embeddings and the Variational Quantum Circuit hold up against the noise that comes along with any real quantum hardware.

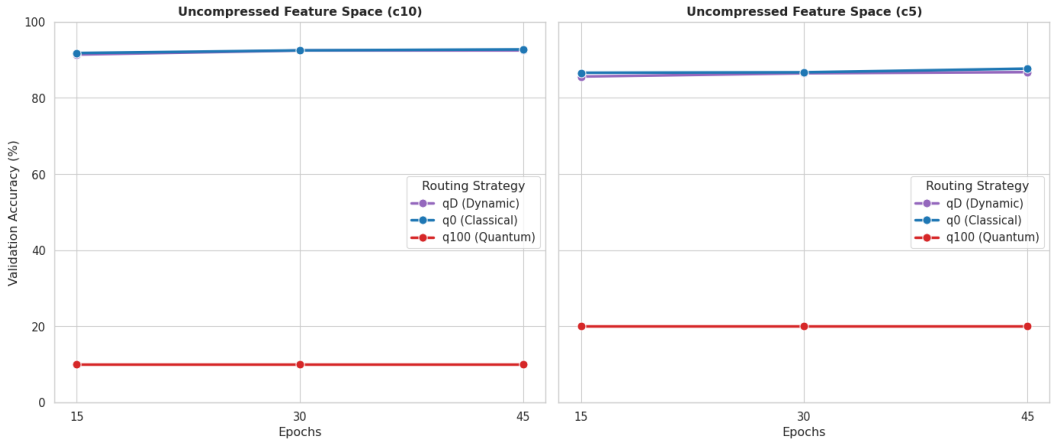

Figure 1: Evaluation accuracy for all 18 evaluation points of the uncompressed feature space

5.3 Initial Hypothesis and the Uncompressed Baseline

The initial thought when it came to this experiment was that in the high dimensional feature space ($d = 128$), the dynamic router would be able to identify the more complex features and send it to the QPU to leverage its abilities. In this original ideation, the thought was that this behavior would be visible in the different routing behaviors on the **c5** subset versus **c10**. This was imagined because the **c5** subset features more correlated images that would require more complex features, therefore, its **q100** would perform better compared to that of **c10** and the percentage routed to quantum of the dynamic runs (**qD**) would also be higher.

Empirical results completely contradicted this hypothesis. When evaluating uncompressed **c10q100** and **c5q100** models, the networks suffered complete failure only being able to achieve 10% and 20% accuracy. This is simply the accuracy of choosing the same class each time. This indicated the models' inability to navigate the Hilbert space, likely because of severe barren plateaus.

Despite this, the dynamically routed models still achieved modest success on their own with over 92% and 86% accuracy for **c10qD** and **c5qD** respectively. However, when visualizing their routing behaviors via a Captum heatmap, it was found that unimportant pixels were simply being jettisoned to the QPU, as to not affect the overall score as seen in Figure 2.

5.4 Mitigating the Barren Plateau

To establish a fair computation baseline and force the router the evaluate based on actual quantum utility, the compressed architecture was introduced. This bottle necked the spatial features down to $d = 8$.

As can be seen in Figure 3, this pivot successfully mitigated the high dimensional scaling issue. One pure quantum model, **c5q100**, was finally able to converge achieving a peak accuracy of 80.58%. The pure quantum model for **c10** still was unable to converge.

Additionally, this fix was further proven when analyzing the new heatmaps of **c5qD**. Rather than the pixels that were being routed to quantum being the edges of the image with little impact on classification, important segments of the image, like sleeves, were being sent to the QPU, like seen in Figure 4.

Therefore, from all this, we have found that the quantum architecture performed best on the restricted **c5** subset and required compressed features to be effective. When presented with distinguishing between very highly correlated classes, both the compressed distributed quantum model (**c5qD** and the compressed pure quantum model (**c5q100**) showed highly competitive and robust performance. This confirmed, alongside the heatmaps, that when constrained into a dense, localized feature space,

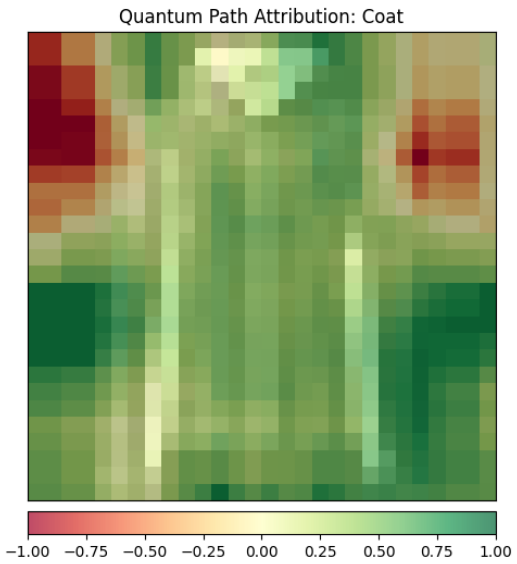

Figure 2: c10qDte45 coat heatmap showing quantum routing being use on nonessential pixels (-1 completely quantum, 1 completely classical)

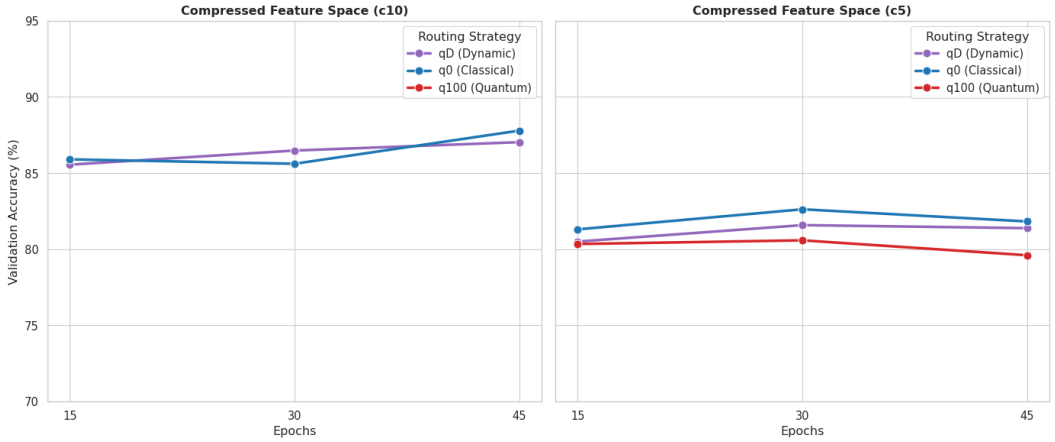

Figure 3: Evaluation accuracy for evaluation points of the compressed feature space (compressed c10q100 is too low and out of frame)

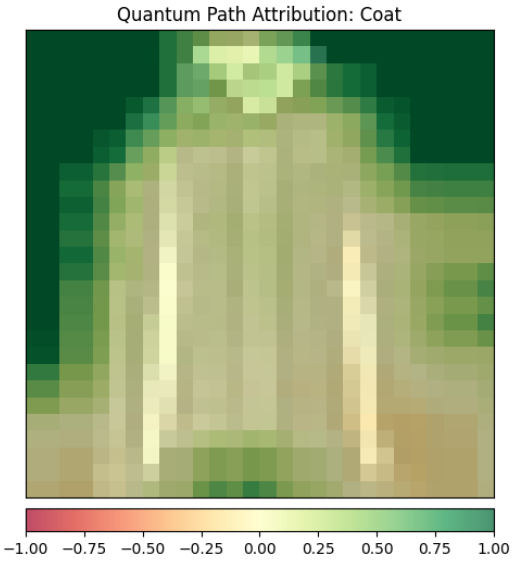

Figure 4: Uncompressed c5qDte45 coat heatmap showing quantum routing being used on important pixels

the Variational Quantum Circuit is adept at capturing the minuscule differences that are required to successfully distinguish complex classes.

5.5 Hardware Benchmarking

To validate the findings on a real QPU and better understand the noise realities of a NISQ era machine, the best performing compressed **c5** models were tested on the physical hardware.

This would transition the model from running in an idealized simulation, to a physical environment that contains systemic gate errors, readout errors, and qubit decoherence. In the noiseless simulation at Epoch 30, both models performed well. **c5qD** had 81.58% accuracy while **c5q100** had a slightly lower 80.58% accuracy.

Despite this competitive performance on the simulator, the **c5q100** suffered a huge drop in performance when run on the quantum computer to 40.34%. From this, it is plain to see that simply pushing all of one's features through the quantum computer is a flawed strategy as the model's capabilities were completely overwhelmed by noise.

Conversely, the **c5qD** performed very well on the quantum hardware. It demonstrated a strong resilience to the physical transition as it not only achieved a stellar accuracy of 82.95%, but by doing so it also outperformed the noiselessly simulated baseline of 81.58%. Because the model learned to only route a specific and highly appropriate subset of features to the quantum circuit, the architecture was insulated from quantum decoherence.

However, we cannot fully trust this metric. This performance boost is likely at least partially driven by a high variance. In the noiseless simulation, the model struggled most with the *Shirt* class and performed less on the *Coat* class which had accuracies of 62.4% and 85.7% respectively. On the IBM System One, these values essentially reversed where *Shirt* accuracy jumped to 78.8% and the *Coat* accuracy dove to 70.6%. The other classes also shifted, but nowhere near as dramatically.

This extreme change in class accuracy highlights the issues that arise with a small subsample. With all of the limitations of the quantum computer, only 176 images were able to be evaluated. This is far less than the 5000 test images available. This smaller test set meant that minor classification shifts had overstated impacts on the final percentage metrics. Therefore, while the physical metrics are promising and definitively demonstrate the noise mitigating ability of the dynamic router, the

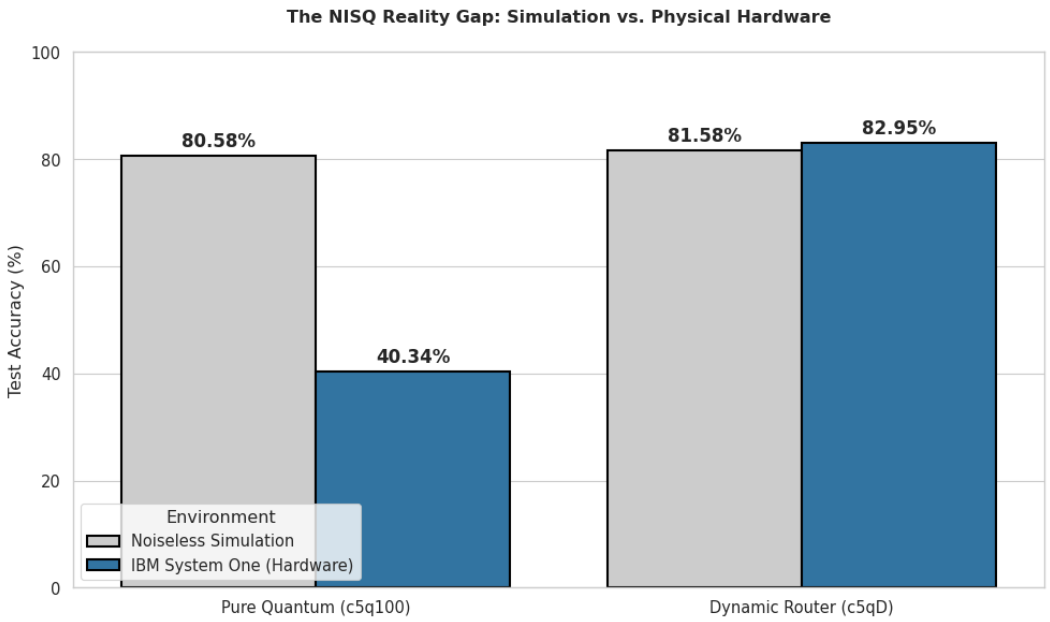

Figure 5: IBM System One results

specific 82.95% accuracy achieved by the physical hardware cannot yet be viewed as a definitive superiority over the simulation.

These hardware results have backed the core claim of the project: in the NISQ era, dynamic hybrid routing is not only necessary for the effective usage of quantum, but it also serves also as an effective noise mitigation strategy. It does so by ensuring that finicky quantum resources are only used on the specific features that justify the risks of physical quantum processing.

6 Conclusion

This paper presented a dynamically routed hybrid quantum-classical architecture that was designed to optimize resource usage and minimize noise effects in the NISQ era. By diagnosing the router as ineffective in the high dimensional feature space, we were then able to demonstrate the combining a classical step down compression funnel with the Straight Through Estimator (STE) attention gate effectively mitigated the barren plateau problem. This pivot forced the network to better utilize the Variational Quantum Circuit for complex features rather than bypassing it. Physical execution on the IBM Quantum System One validated the architecture by demonstrating its resilience to real world quantum decoherence. While the quantum baseline suffered from noise, the dynamically routed model achieved a high accuracy of 82.95%. This work establishes a dynamic feature wise routing not only as a tool for mere efficiency, but as an important noise mitigation strategy for modern day quantum machine learning endeavors.

Future work will focus on expanding the boundaries and the stability of this hybrid pipeline. A critical task will be finding the upper ceiling of the dimensionality of the extracted feature space, determining how high the dimensions can be scaled before the quantum pipeline starts suffering from never ending barren plateaus. Complementing this, we plan to see if other more qubit efficient encodings, like amplitude encoding, can also raise the dimensionality ceiling. Finally, minor overfitting was observed during the simulated 5-class tasks, suggesting that implementing some regularization techniques into the hybrid training loop could further boost the model's overall performance.

7 Use of AI

7.1 Generating Ideas

I used LLMs for finding additional papers that could be relevant to my project. I inserted the original two papers [6] and [3] and queried it to find similar papers. After I had picked the direction for my project, I also asked it for relevant papers there so I could ensure the idea was novel.

7.2 Writing

The biggest thing that I used the LLM for here was formatting. I am awful in latex so I would write things in google docs, then have the LLM format it for latex. Additionally, I had LLMs format all of the latex equations. Finally, I had a final spell and grammar check from the LLM.

7.3 Design and Implementing Experiments

I coded most of the original model on my own with assistance from the LLM for qiskit and pennylane syntax. Then, once I had that model up and running, I used the LLM to create almost the entirety of the adjacent scripts. Because they were so similar (changing from 5 to 10 classes or manipulating the routing mask) it seemed like a good use of the LLM.

7.4 Figures

I gave the LLM my results data, described the pyplots I wanted to make, and it generated the code for me.

References

[1] Bbouzidi, S., Hcini, G., Jdey, I., and Drira, F. Convolutional neural networks and vision transformers for fashion mnist classification: A literature review, 2024.

[2] Bokhan, D., Mastiukova, A. S., Boev, A. S., Trubnikov, D. N., and Fedorov, A. K. Multiclass classification using quantum convolutional neural networks with hybrid quantum-classical learning. *Frontiers in Physics 10* (Nov. 2022).

[3] Lakhdar-Hamina, D., Liu, X., Barney, R., Miller, S. H., Green, A. M., Linke, N. M., and Galitski, V. Benchmarking a tunable quantum neural network on trapped-ion and superconducting hardware, 2025.

[4] Lü, Y., Gao, Q., Lü, J., Ogorzałek, M., and Zheng, J. A quantum convolutional neural network for image classification, 2021.

[5] Shi, M., Situ, H., and Zhang, C. Hybrid quantum neural network structures for image multi-classification, 2023.

[6] Yang, C.-H. H., Qi, J., Chen, S. Y.-C., Chen, P.-Y., Siniscalchi, S. M., Ma, X., and Lee, C.-H. Decentralizing feature extraction with quantum convolutional neural network for automatic speech recognition. In *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (June 2021), IEEE, p. 6523–6527.