Quantum Computing, Industrial and Systems Engineering, 2025

Quantum Neural Networks for Overload Detection and Power Flow Optimization*

Gianna Baker

Washington and Jefferson College, Columbia University
bakergg@washjeff.edu ggb2120@columbia.edu

Start: June 2025 / End: Aug 2025

Abstract

Quantum computing offers a fundamentally new approach to computation, with capabilities that may soon surpass classical algorithms in solving complex, high-dimensional problems. Machine learning, particularly neural networks, provides powerful tools for approximating solutions and learning patterns in systems where explicit optimization is mathematically and resource intensive. In electrical power systems, one such challenge is the AC Optimal Power Flow (ACOPF) problem, which remains difficult to solve efficiently due to its nonlinear and constrained nature. We propose a comparative study between classical and quantum neural networks (QNNs) to explore their ability to detect and correct power line overloads (PLO) as a step toward solving ACOPF. By benchmarking and evaluating performance under matched resource availability, QNNs outperform classical networks, suggesting advantageous scalability as quantum hardware matures. Our multi-stage solution first identifies overloads, then suggests corrective power injection adjustments that respect power flow constraints. Overall, these models aim to support smarter grid management, real-time system optimization, and load balancing, offering greater overall stability in modern power networks.
Keywords: n-bus power system, machine learning, neural networks, quantum optimization

1 Quantum Computing

Quantum computing is a rapidly evolving field that seeks to manipulate quantum bits, or qubits, to unlock computational capabilities that extend beyond the reach of classical systems. At its core, the power of quantum computing stems from

*This research was supported by NSF Grant CNS-2244512

the quantum mechanical principles of superposition and entanglement, enabling parallelism and dependent relationships that classical bits cannot replicate.

Whereas classical bits are constrained to exist strictly in one of two discrete states: 0 or 1, qubits can exist in a linear combination of both states simultaneously. This property, known as superposition, allows a single qubit to encode a continuum of possibilities between 0 and 1. Upon measurement, the superposition collapses probabilistically to one of the basis states, but before that point, the system evolves as if it holds both values concurrently. This phenomenon is perhaps best illustrated through Schrödinger's famous thought experiment, where a cat sealed inside a box is simultaneously alive and dead... its fate only determined when the box is opened. Similarly, in quantum systems, we only observe a definite outcome upon measurement.

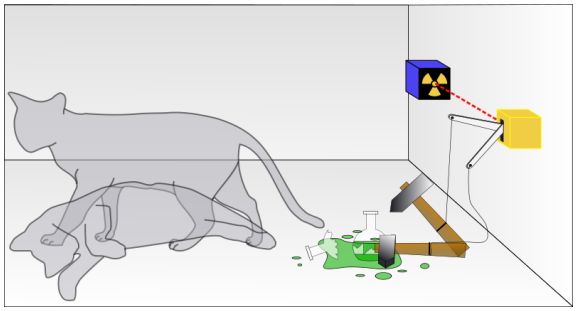

Figure 1: Schrödinger's Cat Metaphor.

Measurement in quantum computing can yield different forms of classical information, including binary collapse (to 0 or 1), expectation values (the weighted average outcome of a repeated measurement), or amplitude probabilities (the likelihoods associated with each basis state). These outcomes inform how quantum algorithms process data and make decisions, particularly in machine learning contexts.

To build quantum algorithms, multiple qubits are linked together in circuits that evolve under unitary transformations. Crucially, qubits can be entangled, a uniquely quantum phenomenon where the state of one qubit becomes inseparably correlated with another, such that the measurement outcome of one instantaneously affects the other, regardless of spatial separation. Entanglement enables the exponential scaling of quantum state space and is essential for outperforming classical counterparts in specific problem domains.

The geometric intuition behind qubit manipulation is elegantly captured by the Bloch sphere, a unit sphere that represents all possible pure states of a single qubit. Points on the surface correspond to different superpositions of 0 and 1, with

poles representing the classical basis states. Quantum operations such as rotations, bit flips, and phase shifts are visualized as movements along the surface of this sphere, enacted through gates in a quantum circuit. These manipulations are not arbitrary; in the context of quantum machine learning, for instance, the goal is to orient qubit states such that their final positions—post-rotation—align closely with the underlying structure of the data. In doing so, measurements collapse the state in a way that is probabilistically aligned with the correct classification or prediction.

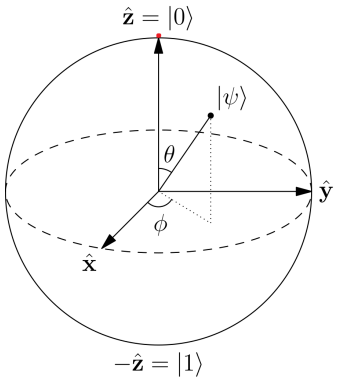

Figure 2: Bloch Sphere

2 Neural Networks

Neural networks, modeled after the brain, learn complex patterns from data, whether through classical layers or quantum circuits, to capture and replicate system behavior. These networks lie at the heart of modern machine learning, offering a flexible architecture composed of tunable parameters designed to learn nonlinear and high-dimensional relationships. By adjusting internal weights and biases, a neural network can approximate a function that maps input data to its corresponding expected or true output. This process is inherently iterative: during training, the model updates its parameters to minimize an objective loss function, and its generalization ability is evaluated on separate test data.

A classical neural network is organized into layers. The *input layer* receives raw features, each node corresponds to a specific input dimension. The *output layer* matches the dimensionality of the target prediction. Sandwiched between these are one or more *hidden layers*, composed of a variable number of artificial neurons. The term *depth* refers to the number of hidden layers, and greater

depth increases representational capacity, enabling the model to learn intricate hierarchical structures. However, an optimal balance must be struck: too shallow a network may underfit, while overly deep architectures risk overfitting or training inefficiencies.

Each neuron in a classical network computes a weighted sum of its inputs, adds a bias term, and passes the result through a nonlinear activation function. Mathematically, a forward pass through a single layer can be written as:

$$\mathbf{z} = \mathbf{W}\mathbf{x} + \mathbf{b}$$
$$\mathbf{a} = \sigma(\mathbf{z})$$

Here, $\mathbf{x}$ is the input vector, $\mathbf{W}$ is a matrix of trainable weights, $\mathbf{b}$ is a bias vector, $\sigma(\cdot)$ is an activation function (such as the sigmoid, $\sigma(z) = \frac{1}{1+e^{-z}}$, or ReLU, $\max(0, z)$), and $\mathbf{a}$ is the activation passed to the next layer.

Learning is driven by *backpropagation*, a recursive application of the *chain rule* from calculus to compute gradients of the loss function with respect to each parameter. For example, if the final output depends on an intermediate variable z, and the loss function is $\mathcal{L}$, then:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}} = \frac{\partial \mathcal{L}}{\partial \mathbf{a}} \cdot \frac{\partial \mathbf{a}}{\partial \mathbf{z}} \cdot \frac{\partial \mathbf{z}}{\partial \mathbf{W}}$$

These gradients are used to update the weights through gradient descent or its variants, iteratively minimizing the error between predicted and true outputs.
In contrast, *quantum neural networks (QNNs)* implement learning via quantum circuits, using quantum mechanical principles to encode and transform information. A QNN begins with a *feature map*, which encodes classical input data into quantum states. Because all qubits begin in the 0 state, we must introduce transformations, typically Hadamard gates to induce superposition and parameterized rotation gates (e.g., $R_Z(x)$) to encode the data along specific axes of the Bloch sphere. While classical networks receive input values directly, quantum networks use this preparatory encoding to represent inputs as quantum states.

The number of *qubits* is generally chosen to match the number of input features. However, *auxiliary or dummy qubits*, which are not directly tied to input data or measured, can be added to increase entanglement and expressivity. This parallels the classical practice of scaling up model size: just as deeper or wider classical networks improve performance through more parameters, quantum models gain modeling capacity with more qubits and entangled interactions.

The next component, the *ansatz*, serves as the quantum analog of hidden layers. This is a parameterized quantum circuit containing gates such as $R_Y(\theta)$, $R_Z(\theta)$, and entangling operations like CNOT. These gates are tunable and represent the learnable parameters of the network. The ansatz determines how the input state is transformed across layers, with deeper ansatz circuits corresponding to deeper classical networks. A general ansatz depth of d would mirror a classical network with d hidden layers.

Following this transformation, the quantum state is measured, often in the *Z basis* (i.e., measuring expectation values of the Pauli-Z operator). This yields classical information which is compared to the true output, and the error is used to guide optimization. However, unlike classical backpropagation, quantum neural networks rely on *parameter shift rules* to compute gradients. These rules involve evaluating the quantum circuit at slightly shifted values of a parameter and using those results to estimate the derivative:

$$\frac{\partial f(\theta)}{\partial \theta} = \frac{f\left(\theta + \frac{\pi}{2}\right) - f\left(\theta - \frac{\pi}{2}\right)}{2}$$

This derivative estimate allows the model to update its parameters using gradient-based optimization strategies without violating the constraints of quantum measurement.

To enable fair comparisons between classical and quantum networks, architectural matching is essential. In this framework, the *number of qubits* is equated with the *number of hidden neurons*, and the *ansatz depth* corresponds to the *number of hidden layers*. While both paradigms aim to learn mappings from data to labels, quantum neural networks introduce fundamentally different mechanisms for information encoding, transformation, and gradient estimation, offering potential advantages in representation, generalization, and computational complexity.

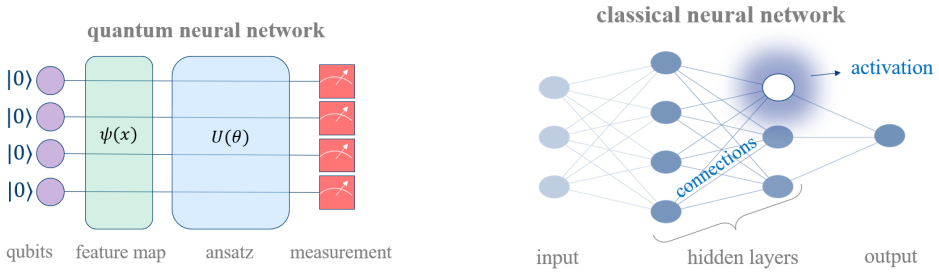

Figure 3: Neural Network Differences

3 Power Systems

An n-bus power system models the flow of electrical energy through a network of *generators* (which produce energy), *loads* (which consume energy), and a designated *slack bus* (which balances generation and load, absorbing system losses). These elements are interconnected via *transmission lines*, which transport both real and reactive power across the grid. Each bus is characterized by two key quantities: the *real power* P (associated with useful work) and the *reactive power* Q (associated with voltage support and stability).

A simple yet instructive example is the **4-bus system**, consisting of one generator bus, two load buses, and a slack bus. This compact system captures the essential components of larger power networks and is often used in educational and experimental contexts. Figure 4 illustrates the layout of such a system.

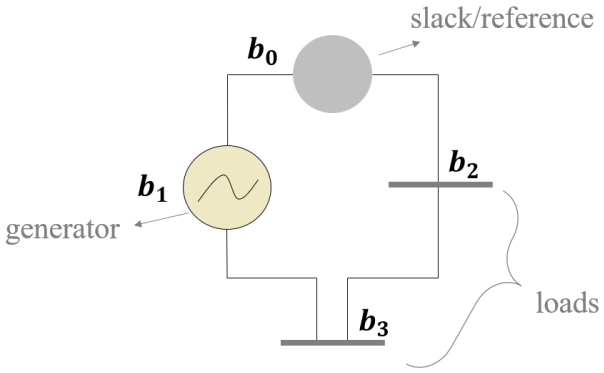

Figure 4: Example of a 4-bus power system.

To understand the system's behavior, we turn to the *AC Power Flow (ACPF)* equations, which relate each bus's power injections to its voltage magnitude and phase angle. These nonlinear equations are derived from Kirchhoff's laws and Ohm's law applied to the system admittance matrix. For bus i in an n-bus system, the real and reactive power injections are:

$$P_i = \sum_{j=1}^{n} |V_i||V_j| \left[G_{ij} \cos(\delta_i - \delta_j) + B_{ij} \sin(\delta_i - \delta_j)\right]$$

$$Q_i = \sum_{j=1}^{n} |V_i||V_j| \left[G_{ij} \sin(\delta_i - \delta_j) - B_{ij} \cos(\delta_i - \delta_j)\right]$$

Here:

- $|V_i|$ is the voltage magnitude at bus i
- δ_i is the voltage phase angle at bus i
- G_{ij} and B_{ij} are the real and imaginary parts of the admittance matrix Y_{bus}

To standardize system analysis and simplify comparisons, these quantities are often expressed in *per unit (p.u.)* notation. For voltage magnitudes, a typical operating range in a realistic grid is between 0.9 and 1.1 p.u., corresponding to 90%–110% of nominal voltage. Voltage phase angles, representing the phase difference between buses, typically range from $-60°$ to $60°$ in practice.

It is important to note that transmission lines themselves carry power, and the direction of power flow follows a fundamental rule: power naturally flows from buses with higher phase angles to those with lower ones. This directional flow, combined with line admittances, determines the active and reactive power transmitted through each line.

Each transmission line also has a *thermal capacity*, representing the maximum load it can safely carry. Although by convention this capacity is often normalized to 100%, it can vary depending on real-world constraints such as weather, infrastructure age, or network optimization strategies. These capacities are enforced in simulation or planning via line loading limits, ensuring that no individual transmission path is overloaded beyond its rated value.

While 4-bus systems provide insight into fundamental mechanisms, real-world power grids scale to much larger configurations. Regional systems may contain hundreds or thousands of buses. For example, the *IEEE 118-bus system* and the *Polish 2383-bus system* are standard benchmarks used to test algorithms at realistic grid scale. These large-scale systems present significant computational challenges, making them ideal testbeds for advanced modeling techniques, including those that leverage classical and quantum machine learning.

4 System Design and Functionality

This project implements a two-stage neural network pipeline for intelligent power system management: an overload detection network and a correction network. The first identifies whether proposed generator values cause line overloads in a power system, and the second suggests adjustments to restore feasibility while still satisfying load values.

To simulate realistic behavior, I used `pandapower` in Python to model four-bus systems and apply full AC power flow calculations using the Newton-Raphson method. Although these are synthetic test systems, they obey physically accurate power flow equations. Each system was regenerated randomly to vary bus connections and parameters, offering diversity while keeping the number of buses fixed. This approach created a rich dataset while maintaining controlled complexity. Since this data was generated manually and without access to standardized IEEE test cases, care was taken to ensure as much validity and realism as possible. Specifically, to ensure that the learning process was not biased toward frequent cases, additional weight was added to rare configurations. More samples were generated for underrepresented overload patterns based on their frequency, resulting in a more balanced and representative training dataset.

All models were implemented in Python using PyTorch and PennyLane. Data collection and preparation were particularly time-intensive, as their sizes ranged from 300 to 3,000 samples, with quantum neural networks typically requiring far fewer data points to perform well compared to classical models.

Each network uses the same six-dimensional input vector: the proposed real and reactive generator power values (P_1, Q_1) and the fixed load values (P_2, Q_2, P_3, Q_3). The loads remain constant to reflect consumer demand, while the generator values represent proposed setpoints that must be tested for feasibility. This setup is realistic in planning and optimization contexts where generator dispatch values are determined based on cost, expected demand, or other practical constraints.

For output, the detection network returns a four-dimensional binary vector indicating whether each transmission line exceeds its capacity. The correction network outputs a two-dimensional continuous vector representing adjustments to the generator's P and Q values. The four line capacities used every time were [70, 72, 63, 36], empirically chosen to produce all 16 distinct overload combinations (e.g., 0000, 0100, 1110, etc.).

Model sizes were matched carefully to evaluate performance under equivalent resource budgets. Due to hardware constraints, the largest quantum model used was limited to 10 qubits. This determined the structure of the quantum networks (three hidden layers with 10 qubits each) and guided the construction of a "matched-resources" classical model with 10 neurons per hidden layer. For comparison, I also evaluated a larger classical model with hidden layers of sizes 148–84–24, which gave better accuracy but required significantly more parameters.

The two-stage design enables flexible usage: the detection model can quickly flag unfeasible configurations, while the correction model activates only if needed.

The usage of both these stages supports smarter grid management, real-time system optimization, and load balancing, offering greater over-

all stability in modern power networks.

4.1 Overload Detection

This stage identifies whether any transmission line in the system is overloaded, using either a classical or quantum neural network with matched resource sizes. The classical model has architecture 6–10–10–10–4, while the quantum model uses 10 qubits. If a line is overloaded, the corresponding output is 1; otherwise, it is 0. Since this is a multi-label binary classification problem, binary cross-entropy loss (`BCEWithLogitsLoss`) is used.

The task is particularly challenging: even if the model predicts each line with 90% accuracy, the joint probability of predicting all four lines correctly is only 65%. This complexity should be kept in mind when interpreting results.

The quantum model begins with Hadamard gates and Z-rotations (AngleEmbedding) applied to each qubit. Then, all qubits, including auxiliary ones, pass through three layers of `StronglyEntanglingLayers` (a PennyLane ansatz), followed by additional CNOT gates. The first four qubits are then measured.

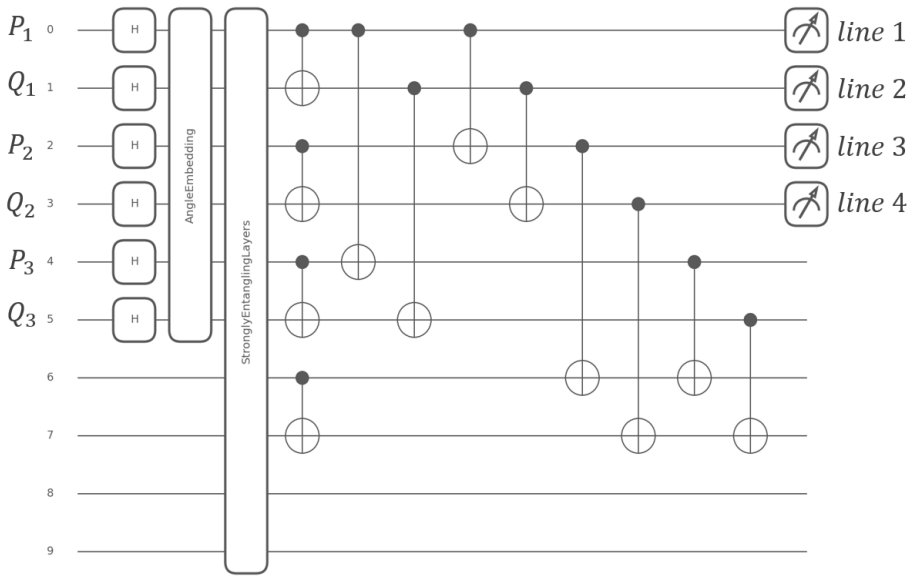

Figure 5: Quantum Circuit Architecture

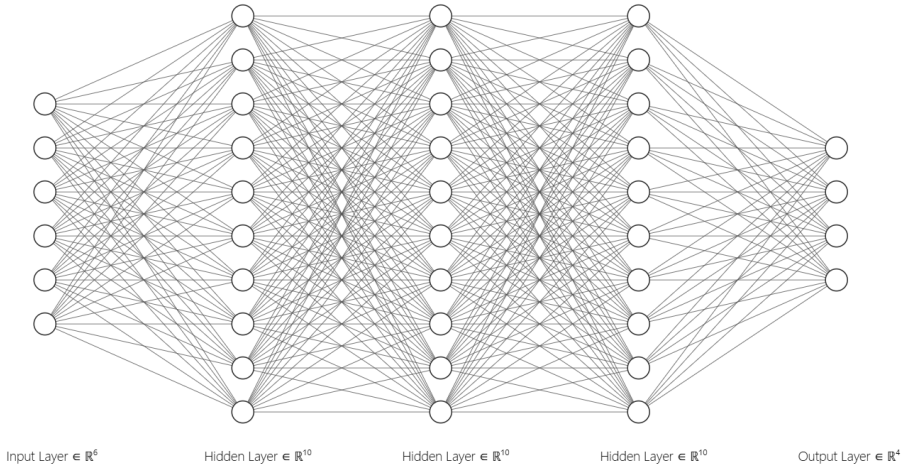

Figure 6: Classical Network Architecture

Results

The classical network reaches 86.57% accuracy with 18.96% variability, while the quantum network attains 73.05% accuracy with only 7.43% variability. Despite having lower accuracy, the quantum model delivers more stable and consistent predictions, critical for reliability in classification tasks. This consistency suggests stronger generalization and learning stability from the quantum network. Classical models can sometimes overfit or underperform depending on initialization and training, but quantum models appear to offer a more dependable learning dynamic, especially promising once hardware scales to support architectures equivalent to the larger classical model (6–148–84–24–4 would require approximately 148 qubits).

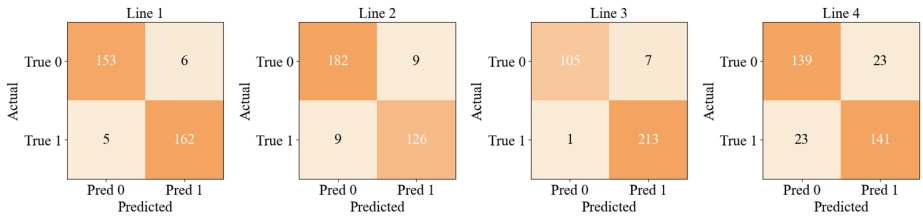

Figure 7: Quantum Overloads Confusion Matrices

The larger classical model, achieves between 92% and 98% accuracy due to its

deeper architecture. Since sufficiently sized classical neural networks can approximate the ACPF equations with high fidelity, this success suggests that quantum models, once scalable, can compete effectively and even surpass classical models through their stability.

4.2 Overload Correction

This network approximates the necessary changes to generator power values to eliminate line overloads while still satisfying load demands. Unlike the detection task (a classification problem), this is a continuous approximation task and uses mean squared error (MSE) as its loss function.

To train the correction network, I reused the same input data from the detection task. However, generating target outputs (y_true) posed a significant challenge. Standard optimal power flow (OPF) solvers proved unsuitable, as they are designed to minimize cost functions and not to eliminate line overloads specifically. As a result, OPF solutions often failed to produce the necessary corrections for my data. To address this, I developed a brute-force method that manually explored 4,900 combinations of incremental changes to active (P) and reactive (Q) power values. For each input sample, this method searched for generator power adjustments that would resolve all overloads while still satisfying the load demands. Only combinations that produced feasible power flow solutions were kept, and their corresponding input-output pairs were used to train the model. This approach, implemented as a double nested loop, was computationally intensive but ultimately the most effective strategy for creating a meaningful dataset for learning feasible corrective actions.

The classical network retains the same structure as before: a 6-10-10-10-2 architecture, but now maps to two outputs: ΔP_1 and ΔQ_1. The quantum network also maintains Hadamard and angle embedding layers for input encoding, followed by three layers of parameterized gates (ansatz), applied to all qubits. However, a key difference in this stage is the introduction of **grid topology** into the quantum model architecture.

Here, CNOT gates are arranged to reflect the transmission line connections, mirroring the structure of the power grid. Additionally, the P and Q components of each bus are connected, and two auxiliary qubits are designated as slack/reference buses for these connections. Although this added complexity did not improve performance in the detection network, likely due to the difficulty of 4-label quantum classification, it significantly improves the performance of the correction network.

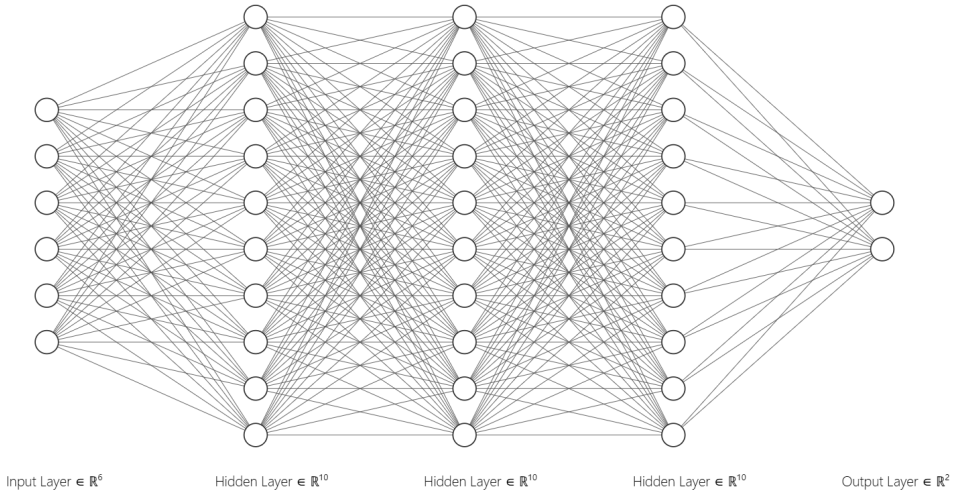

Figure 8: Classical Network Architecture

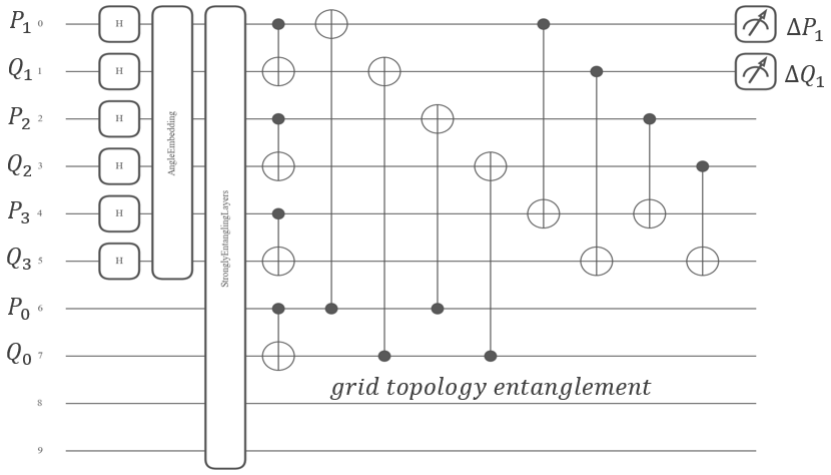

Figure 9: Quantum Network Architecture

Now, the model is injected with actual system information, which reflects the true physical dependencies happening, a technique that proves to really help quantum learn, one that classical models do not have a comparable alternative for. This incorporation of topology gives the quantum model a leg up in capturing physical structure and constraints more faithfully than classical networks.

4.2.1 Results

At the same scale, the classical model plateaus near a training error of 0.78, while the quantum model steadily improves to 0.12. This suggests a smoother optimization landscape and better generalization.

Figure 10: Training Error with Matched Resources

Even when compared against the larger classical model, the same quantum network still outperforms it: achieving 73.85% accuracy versus the larger classical model's 72.31%. This highlights the quantum model's efficiency in capturing complex grid behavior with minimal resources, and shows greater promise if scaled similarly. **These results suggest that QNNs may scale better for grid control tasks**, with the ability to continue learning where classical models stagnate.

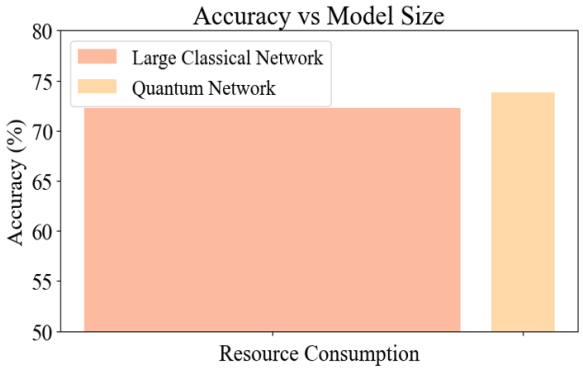

Figure 11: Accuracy of Networks with Mismatched Resources

References

[1] Z. Kaseb, "Quantum neural networks for power flow analysis," *Elsevier BV*, 2024.

[2] T. Pham, "Neural network-based power flow model," *GreenTech*, 2022.

[3] S. Miraftabzadeh, "A survey of machine learning applications for power system analytics," in *International Conference on Environment and Electrical Engineering*, 2019.