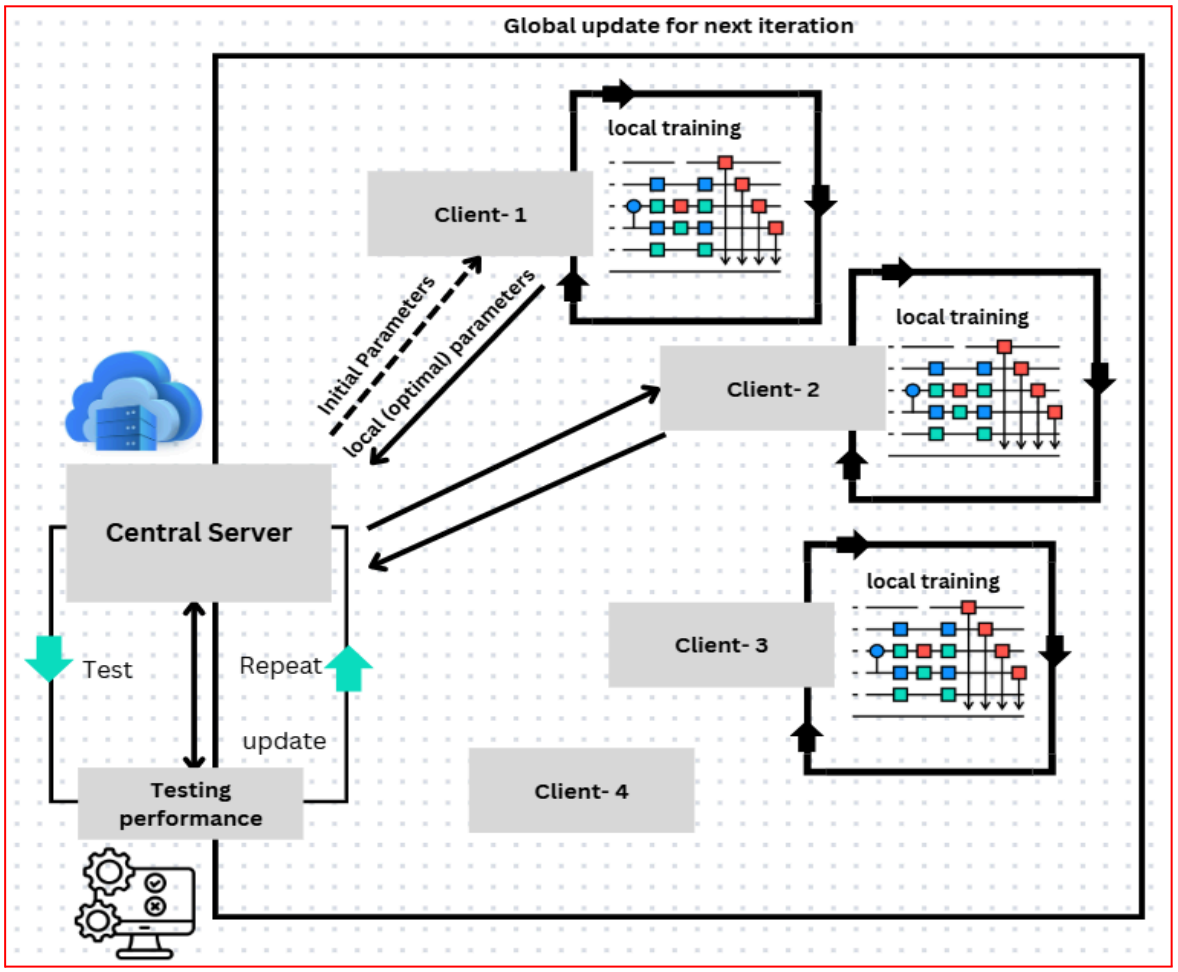
Quantum Federated Learning using QNN and HQNN in QML

Abbreviation:-

QNN - Quantum Neural Network
HQNN - Hybrid Quantum Neural Network
QNN-P - Quantum Neural Network (using pennylane)
QNN-Q - Quantum Neural Network (using qiskit)

In classical federated learning, the data remains decentralized on multiple devices (clients) rather than transferred to a central server. Each client trains a **local copy** of the global model on its data, independently of other clients. This training is done for a specified number of local epochs, during which the client optimizes the model parameters based on its local data. After training, each client's model will have slightly different parameters because each client is learning from different data.

Instead of sending the data itself, each client sends only the optimized parameters (weights) of its locally trained model to the central server. The central server **aggregates** (averages) the parameters received from all clients. This aggregation is often done by taking a weighted average. The server then updates the global model with these aggregated parameters. This updated global model now incorporates insights from all clients' training, making it more generalizable across diverse data sources. After aggregation, the global model can be tested on a separate validation or test dataset to evaluate its overall performance.

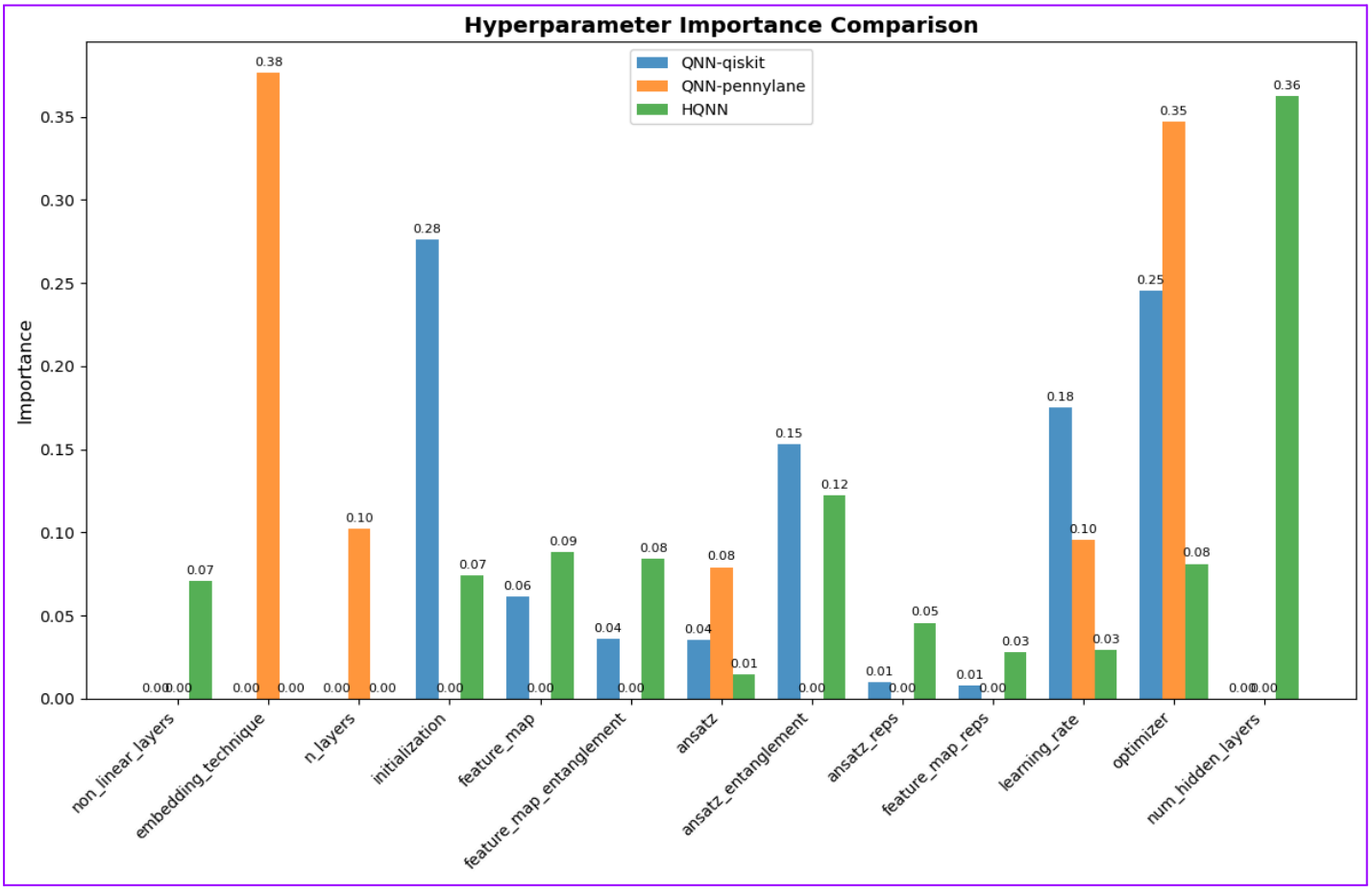


Bayesian Optimization for fine-tuning quantum algorithms in federated learning

Bayesian optimization with the open-source software **Optuna** was used to fine-tune and optimise hyperparameters for quantum neural networks (QNNs) implemented in both Qiskit and Pennylane frameworks. These optimizations were applied within the context of a federated learning setup to enhance model performance across various configurations. Each client can run this before starting the training process. Optuna's default **TPESampler** (Tree-structured Parzen Estimator) was employed, effectively balancing the trade-off between exploration of the hyperparameter space and exploitation of known optimal regions.

The analysis included three distinct algorithms: **QNN-Q**, **QNN-P**, and a **Hybrid QNN**, combining classical and quantum components. A detailed study of hyperparameter importance revealed how specific parameters—such as learning rate, batch size, and circuit depth—significantly influenced the performance of each algorithm. To manage computational complexity, only 50 hyperparameters were considered, reflecting the time-intensive nature of quantum simulations and the non-availability of GPUs to accelerate computations.

This study highlights the critical trade-off between computational time and the quality of optimised parameters, emphasizing the need for careful resource allocation in quantum machine learning workflows. Future iterations could incorporate GPU acceleration and explore advanced samplers to further scale the optimization process while maintaining robust results.



Training and testing of Quantum Algorithms

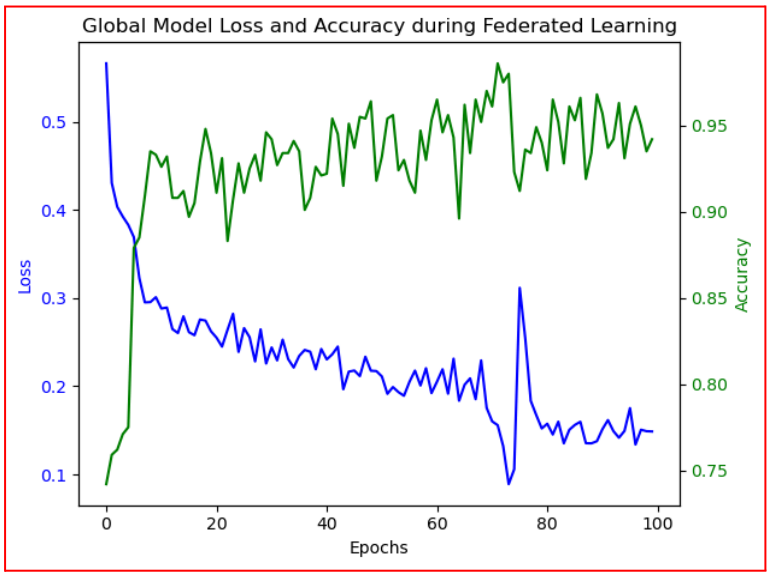
The figures below illustrate the global model's training process across 100 epochs during federated learning, focusing on three quantum algorithms: **QNN-P**, **HQNN**, and **QNN-Q**. These algorithms were tested under consistent client configurations (2 clients), with variations in training and testing dataset sizes.

1. **QNN-P**: Achieved the highest training accuracy of **98%**, demonstrating robust performance. The algorithm utilised **5000 data points** for training and **1500 data points** for testing. The Pennylane framework proved inherently faster than other quantum frameworks used in this study.

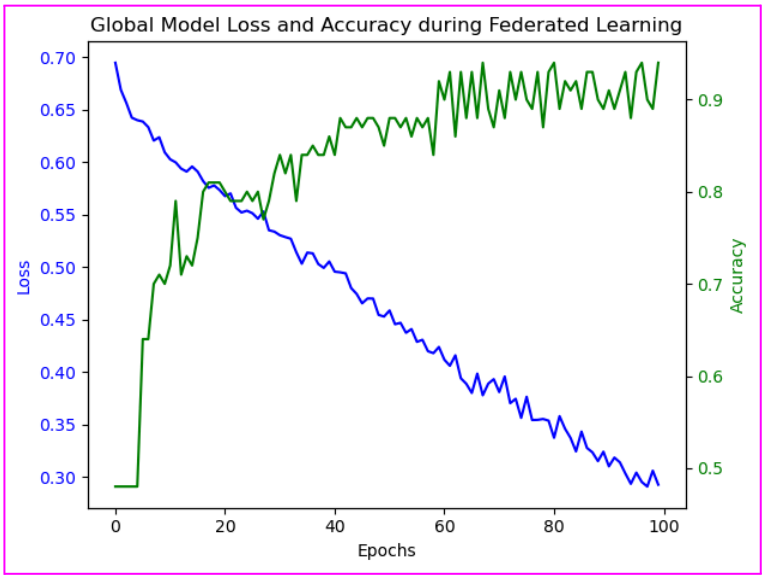
2. **HQNN**: Reached a peak training accuracy of **94%**, slightly lower than QNN-P but still significantly effective. This algorithm was trained using **500 data points** and tested with **200 data points**.

3. **QNN-Q**: The Qiskit-based model achieved a maximum training accuracy of **69%**. While this accuracy is comparatively lower, the computational overhead of Qiskit for QML applications contributed to significantly longer training times. As a result, fewer data points (**500 for training, and 200 for testing**) were selected for QNN-Q compared to the other algorithms to manage the extended computational requirements.

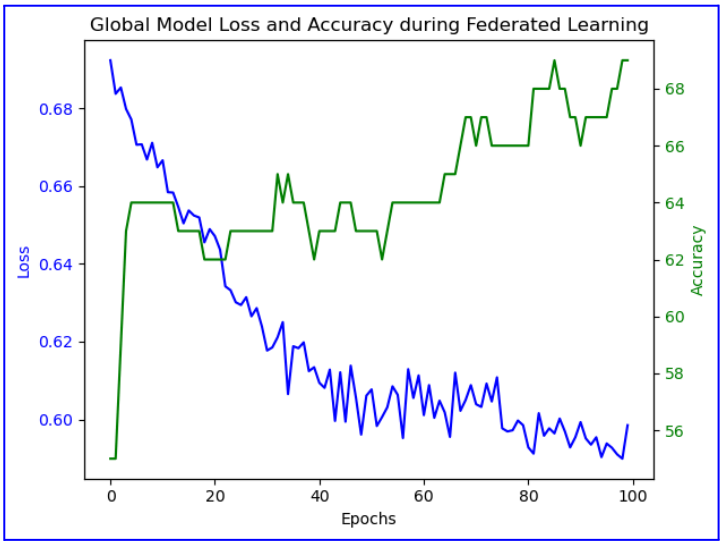
a) QNN-P



b) HQNN

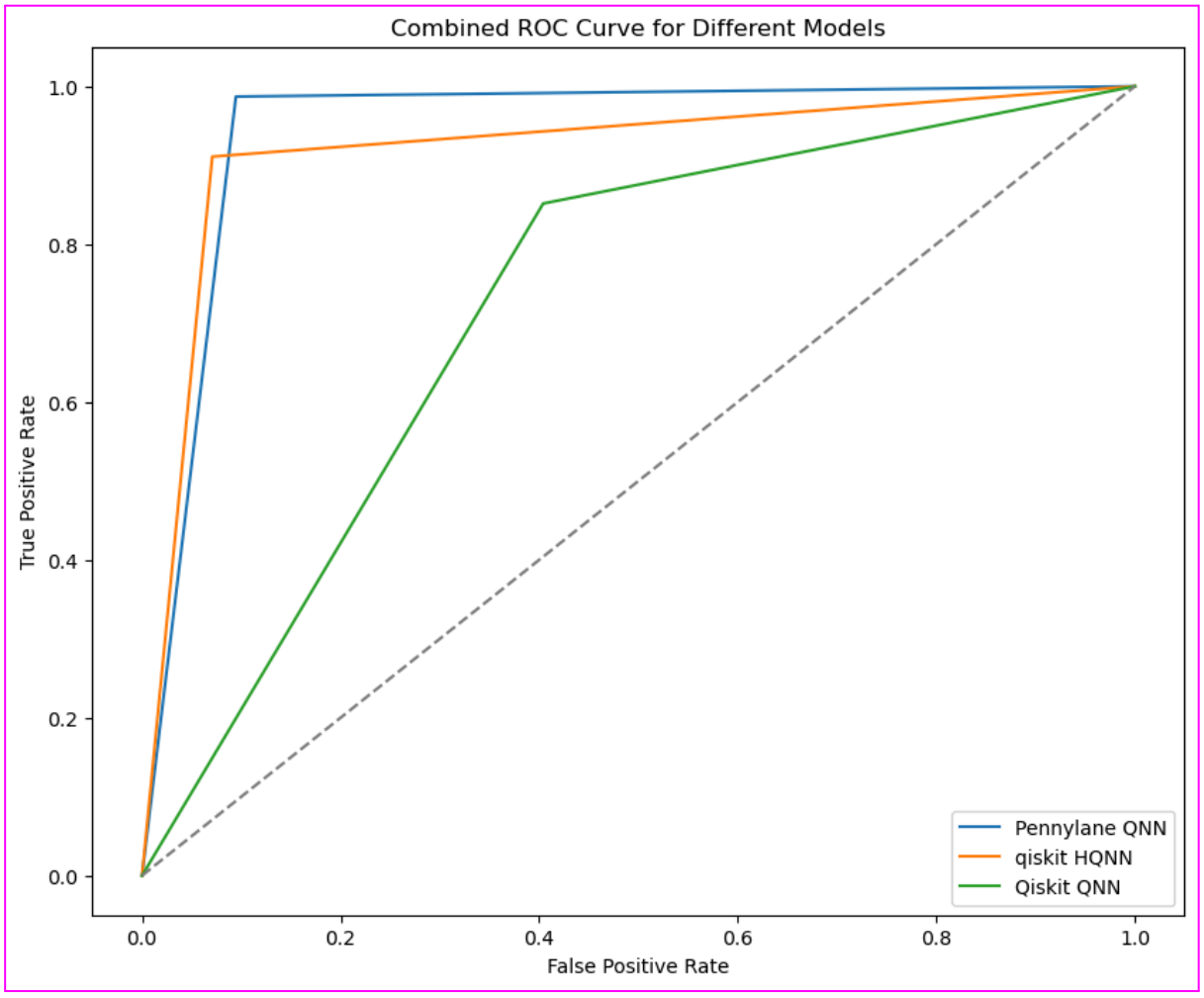


c) QNN-Q



For the testing results, we report metrics including Accuracy, F1 Score, Precision, Recall, and the AUC value for each algorithm. Additionally, a corresponding ROC curve is provided below.

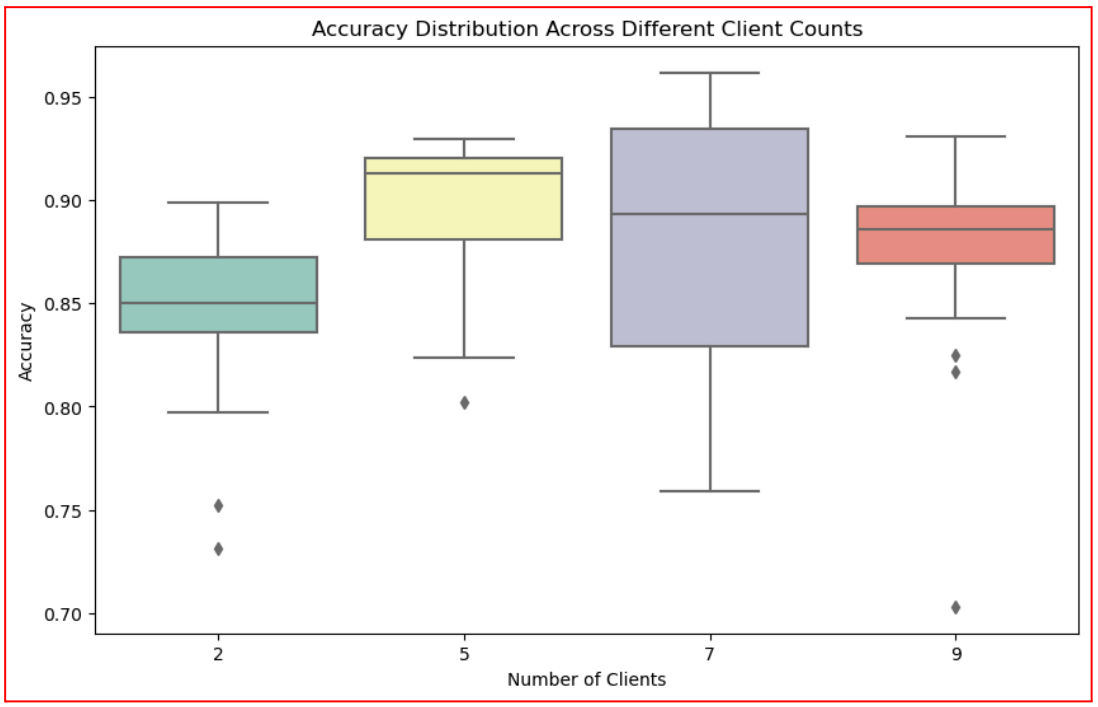
Algorithms	Accuracy	F1 Score	Precision	Recall	AUC
QNN-P	0.9473	0.9506	0.9168	0.9870	0.9462
HQNN	0.9200	0.9200	0.9293	0.9109	0.9201
QNN-Q	0.7250	0.7577	0.6825	0.8515	0.7237

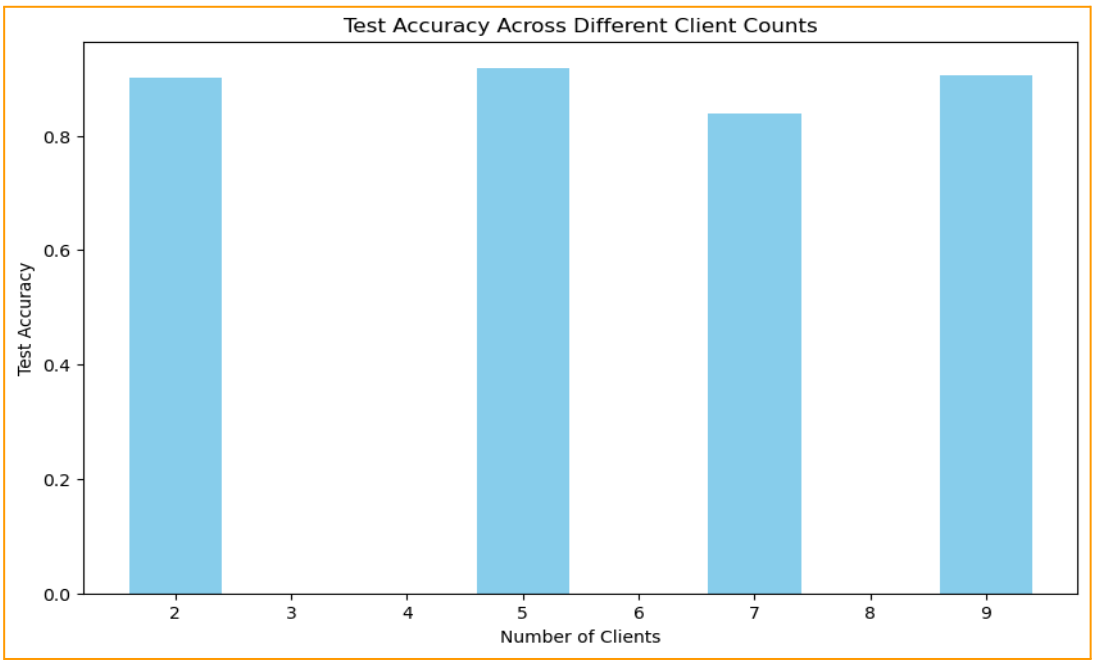


Effect of number of clients on training and testing process

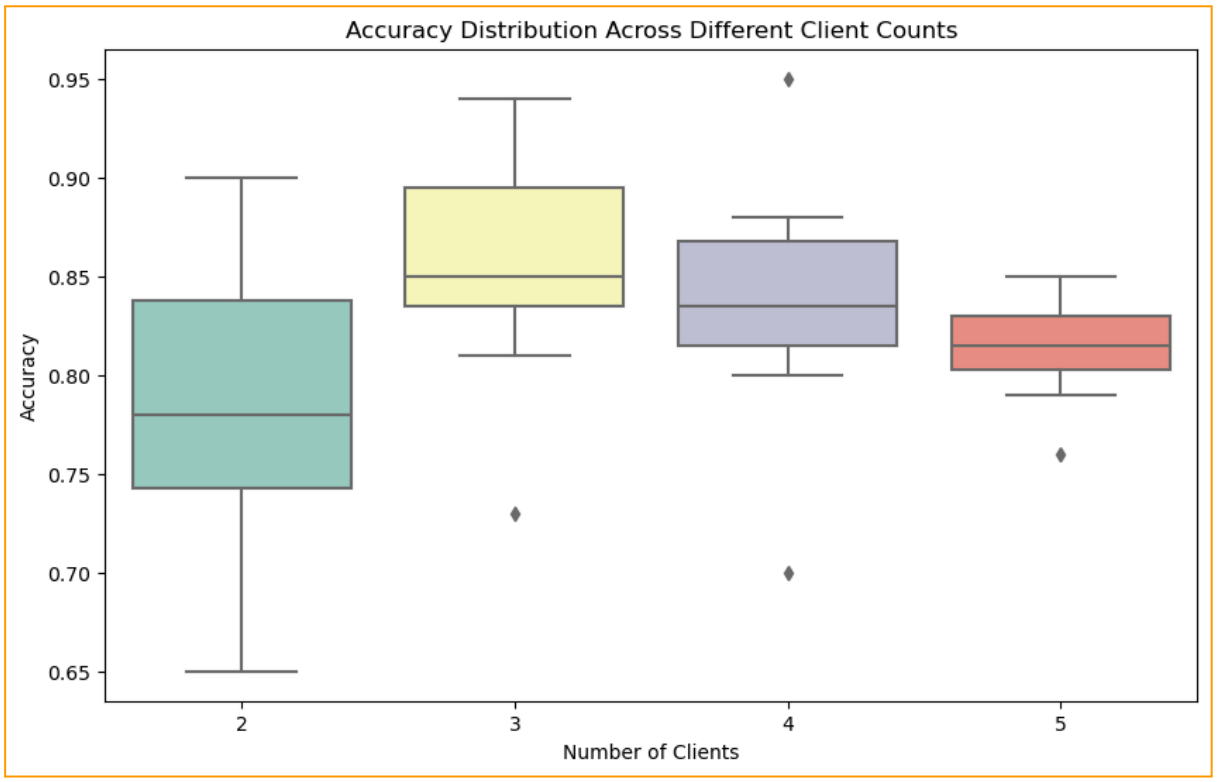
The plots illustrate the performance of quantum algorithms in a federated learning setup as the number of clients increases. The **box plot** highlights the distribution of training accuracies, which shows variability across client numbers and accuracy fluctuations due to diversity in client data and the aggregation model. **The bar plot** shows test accuracy across different client counts, demonstrating consistent performance with minor variations, indicating the robustness of the federated model.

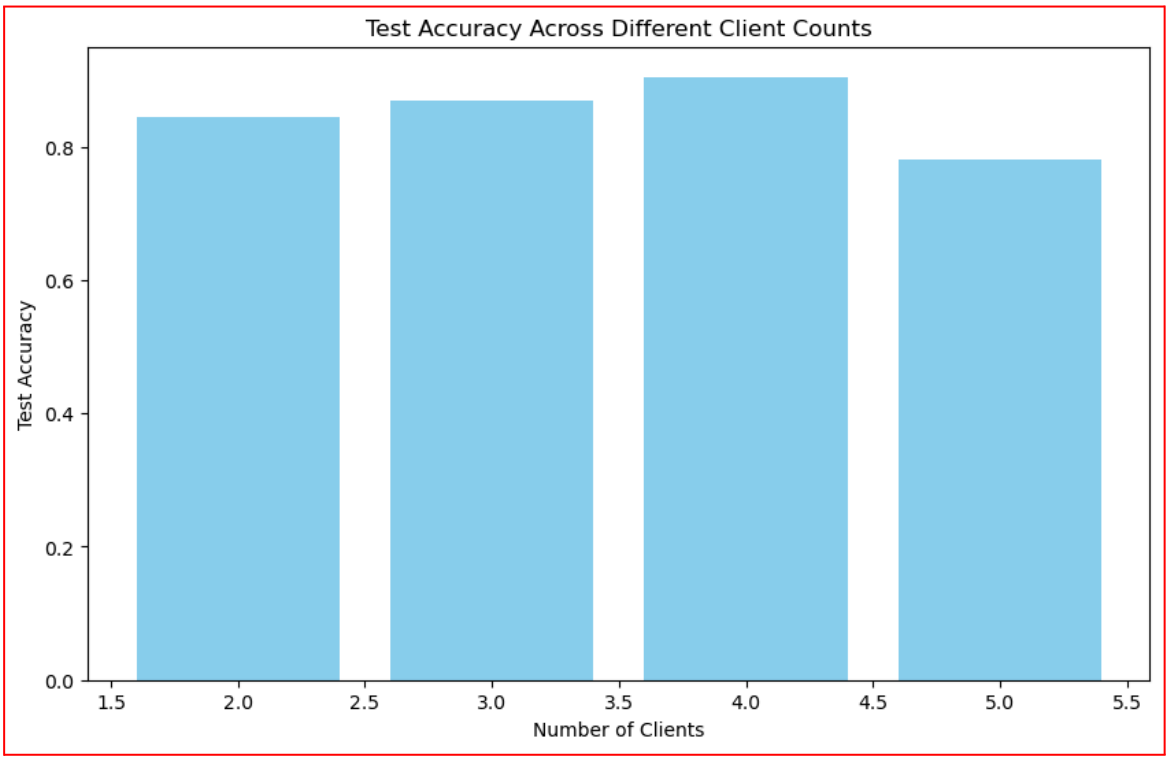
QNN-P model:-





HQNN model:-

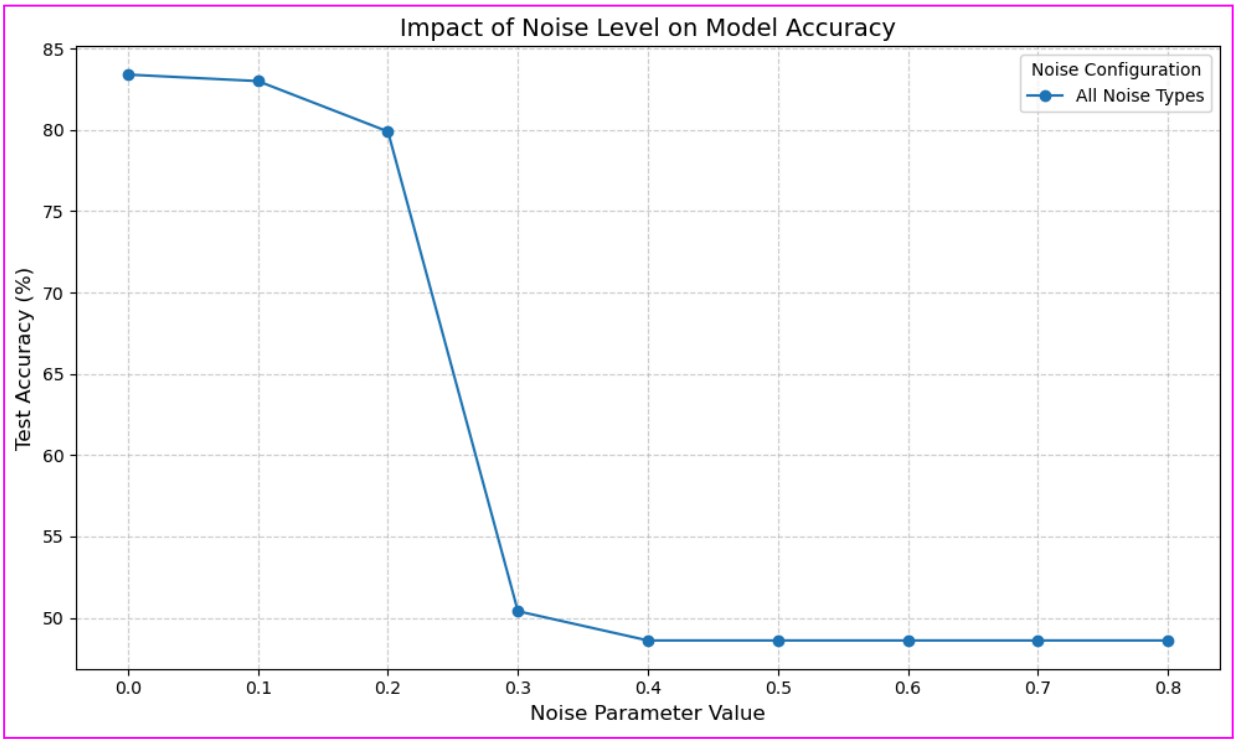

Accuracy Distribution Across Different Client Counts
Accuracy
0.95
0.90
0.85
0.80
0.75
0.70
0.65
2
3
4
5
Number of Clients


Test Accuracy Across Different Client Counts
Test Accuracy
0.8
0.6
0.4
0.2
0.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
5.5
Number of Clients

Effect of quantum hardware on accuracy during the training process

Due to the limited availability of quantum hardware time, the noisy quantum algorithm was evaluated on the simulated backend to analyse its performance under varying noise conditions. The testing accuracy was measured against noise levels ranging from 0.0 to 0.8. Three types of noise channels were considered: **DepolarizingChannel**, **AmplitudeDamping**, and **PhaseDamping**, each applied independently to specific qubits.

The **DepolarizingChannel** was applied to qubit 0, **AmplitudeDamping** to qubit 1, and **PhaseDamping** to qubit 2, as per the defined noise mapping in the quantum circuit. This setup allowed for isolating the effect of each noise type on the model's performance. The resulting accuracy was plotted against the noise levels, offering insights into the robustness of the algorithm under noisy conditions.



Future Work

- Aggregating model parameters based on weighted averaging assign higher weights to clients with better-performing or more extensive data contributions. Experiment with different weighting strategies to see if this improves model performance on the test dataset.

- Investigate how noise affects circuit performance through running on different quantum hardware analysis backend properties.

References

- Innan, N., Khan, M. A. Z., Marchisio, A., Shafique, M., & Bennai, M. (2024). FedQNN: Federated Learning using Quantum Neural Networks. *arXiv preprint* arXiv:2403.10861.

- Tudisco, A., Volpe, D., Ranieri, G., Curato, G., Ricossa, D., Graziano, M., & Corbelletto, D. (2024). Evaluating the computational advantages of the Variational Quantum Circuit model in Financial Fraud Detection. *IEEE Access*.

- Di Pierro, A., & Incudini, M. (2021). Quantum machine learning and fraud detection. In *Protocols, Strands, and Logic: Essays Dedicated to Joshua Guttman on the Occasion of his 66.66th Birthday* (pp. 139-155). Cham: Springer International Publishing.