Stage 1: Implementing a QRNG on IBM QPUs

Overview if quantum comp, superposition, importance,

- Describe the quantum circuit you designed for the QRNG. Mention that the circuit was designed to place qubits in an equal superposition state using Hadamard gates.
- Explain the process and how it measures the qubits to obtain random outcomes.
- Discuss the results obtained from the simulator, such as the distribution of the generated random numbers, and how these results were consistent with expectations.

To perform quantum number generation, we need to prepare a circuit with n number of bits, where n is the length of the bit string we want to generate, and then add a Hadamard gate to get equal superposition, so equal probabilities for each qubit.

Then for the measurement part, we made several tests with it.

1. We used a quantum simulator for Qiskit, to verify that the code is correct, but this was only limited to 20 qubits, hence the length of the bit string.
2. We used the IBM quantum machine, which is limited

Stage 2: Achieving High Accuracy with Classification Models

The goal of this stage is to evaluate and enhance the accuracy of classification models designed to distinguish between Quantum Random Number Generator (QRNG) data and Pseudo-Random Number Generator (PRNG) data. This involves training, evaluating, and optimizing various machine learning models to achieve high classification accuracy.

Models Evaluated:

- Gradient Boosting Classifier: This ensemble technique builds models sequentially to correct errors made by previous models.
- Random Forest Classifier: An ensemble method that combines multiple decision trees to improve accuracy and mitigate overfitting.
- Neural Network (MLPClassifier): A multi-layer perceptron model that utilizes neural networks to capture complex patterns in data.

Evaluation Metrics:

- Accuracy: Measures the proportion of correctly predicted instances out of the total number of instances.

- -Confusion Matrix: Provides a detailed breakdown of true positives, true negatives, false positives, and false negatives, allowing for a deeper understanding of model performance.

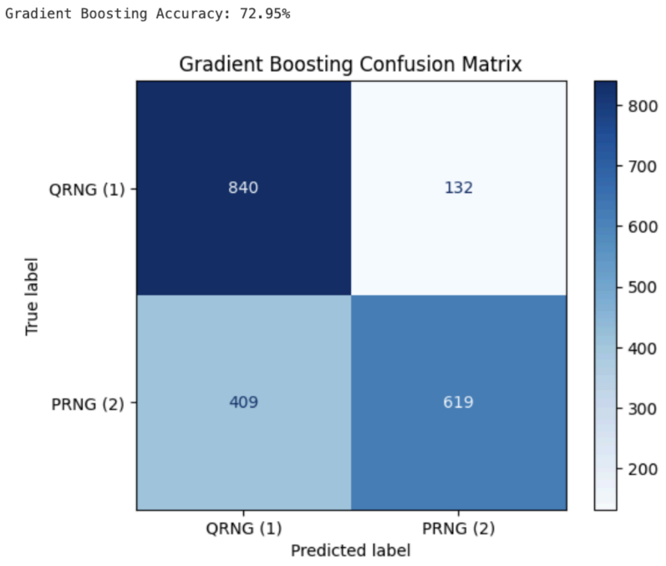
Initial Training:

Each model was trained using the training dataset, followed by evaluation on the test dataset. The accuracy of each model was calculated to assess performance. Confusion matrices were generated to visualize how well each model performed in distinguishing between QRNG and PRNG data.

Model Optimization:

- Random Forest Classifier:
  - Hyperparameters were optimized through a grid search process. This included parameters such as the number of trees, maximum depth, and minimum samples required for splitting and leaf nodes.
  - The best-performing model parameters were identified, and the model was re-evaluated on the test set to confirm improvements in accuracy.
- Gradient Boosting Classifier:
  - Similar to Random Forest, a grid search was used to find the optimal hyperparameters, including the number of estimators, learning rate, and tree depth.

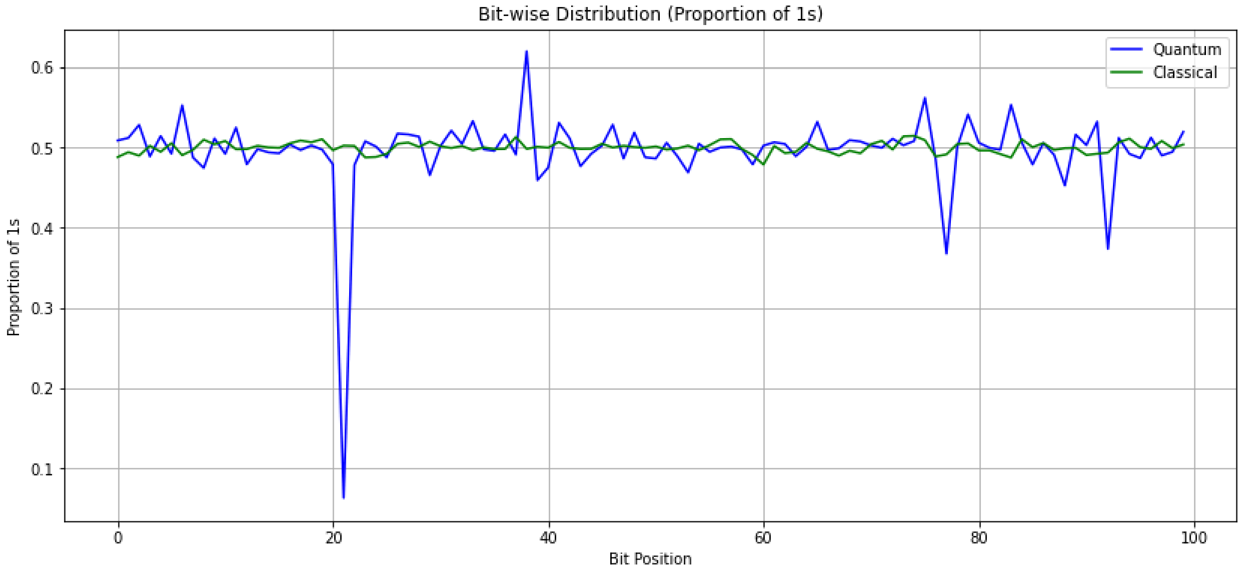
The model was fine-tuned based on these parameters and re-assessed on the test dataset. In the provided notebook, the results can be observed for this part. We stayed with Gradient Boosting classifier with a 73 % of accuracy on the test set.



```
Gradient Boosting Accuracy: 72.95%
```

Stage 3: Characterizing Noise and Fidelity

- Describe noise and fidelity
- How these factors affect the quality of the data generated
- Data visualizations
- Analyze the visualizations



In this analysis, we focused on the bitwise distribution of quantum random number sequences generated using IBM's quantum processors, with the goal of assessing their uniformity and identifying any potential deviations that could indicate issues with randomness quality. To do this, we first converted the sequences into a bitwise matrix, where each column represented a specific bit position across all sequences. We then calculated the proportion of 1s at each bit position, expecting that, in a truly random and uniform distribution, these proportions would hover around 0.5 for every bit position.

Upon visualizing the bitwise distribution, we observed that while most bit positions showed a proportion of 1s close to the expected 0.5, there was a notable deviation at the 20th bit position, where the proportion of 1s fell below 0.1, for instance. This significant drop suggests that there is built-in noise affecting this specific bit position, leading to a non-uniform distribution that diverges from what would be expected in high-quality random data. This kind of deviation is often correlated with the noise and fidelity characteristics of the quantum hardware used to generate the sequences. In this case, the noise and possibly lower fidelity at certain points in the quantum system appear to have impacted the randomness at specific bit positions, resulting in these observed irregularities.

Say something regarding the IBM computer metrics

Stage 4: Pre-processing and Post-processing for High Entropy

In our project, the post-processing stage plays a crucial role in converting the raw quantum bit sequence into high-quality output with a uniform bit distribution. The primary goal of this stage is randomness extraction, which corrects biases and correlations inherent in physical random number generators (RNGs). These imperfections can arise even in high-entropy sources.

Key Techniques Used:

Toeplitz Matrix Transformation:
We applied Toeplitz matrix transformations to the raw quantum bit sequences. This technique is designed to enhance the randomness by removing patterns and correlations, resulting in a more uniformly distributed bit stream.

Hash Function:
A hashing function was employed to further process the data. Hashing helps in randomizing the bit sequence and ensures that the output is as close as possible to a uniform distribution, which is essential for achieving high entropy.

NIST Statistical Tests:

The subjected the processed data to the NIST test suite, which evaluates the statistical randomness of the bit sequences. These tests assess various aspects of randomness, including frequency, runs, and patterns.

Results:
Through our analysis and the application of these techniques, we observed that the processed quantum random number sequences exhibited significantly higher entropy compared to classical pseudorandom number generators (PRNGs). The post-processing methods effectively corrected for biases and correlations, confirming that quantum random numbers are indeed more random and uniform than their classical counterparts.

This process not only enhances the quality of the generated quantum random numbers but also demonstrates their superior randomness compared to classical methods, validating the efficacy of our post-processing techniques.

Stage 5: Measuring Entropy and Real-world Implementation

Entropy is a measure of uncertainty or randomness in a dataset. In the context of random number generation, entropy quantifies how unpredictable the data is.

- **Quantum Data Entropy**: 0.9999519681094957
- **Classical Data Entropy**: 0.9999998868811643

Both entropy values are close to 1, which indicates that both datasets exhibit high randomness and unpredictability. The classical data's entropy is slightly closer to 1 compared to the quantum data's entropy. This suggests that the classical data might be marginally more random or less predictable than the quantum data in this particular measurement. However, such difference would only matter in highly sensitive applications. The slight difference from 1 may be due to imperfections in the quantum hardware, noise, or calibration issues.

There are several scenarios where the generation of truly random numbers—those not following a fixed seed or deterministic process—is crucial. Such scenarios demand the highest level of unpredictability to ensure security and integrity.

In secure communication systems, encryption keys must be generated with absolute randomness to protect data from unauthorized access. Encryption algorithms rely on keys that are unpredictable to prevent attacks like brute force or cryptographic analysis.
For example, in modern cryptographic systems such as AES (Advanced Encryption Standard), QRNG can be used to generate keys that secure sensitive data transmissions, such as financial transactions or government communications.

Quantum Random Number Generators (QRNGs) can also be applied in Predicting Winning Numbers and Game Outcomes, by Ensuring fairness and unpredictability in lotteries and gambling systems. And random simulations in algorithms by roviding unbiased randomness for financial modeling and scientific research simulations.

Conclusion

Quantum Random Number Generators (QRNGs) offer unparalleled randomness and security, making them ideal for applications requiring high levels of unpredictability. From ensuring fairness in gambling to enhancing the accuracy of simulations, QRNGs provide a crucial advantage over classical methods by leveraging the inherent unpredictability of quantum mechanics.