Project Report

Project Title-

Anomaly Detection in Wafer Manufacturing

Submitted to:
Asst Prof Meetha Shenoy
Instructor In-charge, ML for EE (EEE G513)
Department of Electrical and Electronics Engineering, BITS Pilani

Prepared by:
Aaryav Mishra (2020B5A30926P)
Adarsh Mohan Sharma (2020B5A32003P)
Sathwik Gundala (2020B5A31997P)

PROBLEM STATEMENT:

In wafer manufacturing, the goal is to ensure the production of high-quality wafers without defects. Defective wafers can lead to a variety of issues, including reduced yield, increased production costs, and potential damage to downstream processes. Anomalies in the manufacturing process, such as defects in wafers, need to be identified and addressed promptly to maintain product quality and operational efficiency.The problem of anomaly detection in wafer manufacturing involves identifying defective wafers in a production process that generates a substantial amount of data, often referred to as Industrial Internet of Things (IIoT) data. The challenge is compounded by the rarity of anomalies, making them akin to finding a needle in a haystack.

DATA USED:

(Taken from Kaggle: https://www.kaggle.com/datasets/arbazkhan971/anomaly-detection/data)

Dataset Description:

Train.csv - 1763 rows x 1559 columns
Test.csv - 756 rows x 1558 columns

Attribute Description:

Feature_1 - Feature_1558 - Represents the various attributes that were collected from the manufacturing machine
Class - (0 or 1) - Represents Good/Anomalous class labels for the products

APPROACH TAKEN:

In addressing the challenge of anomaly detection in wafer manufacturing, we have chosen to employ Quantum Neural Networks (QNNs) and Generative Adversarial Networks (GANs) to explore novel and effective solutions. The rationale behind this choice lies in the unique capabilities offered by these models in handling complex, high-frequency, and imbalanced datasets, as well as their potential to mitigate biases inherent in labeled datasets.

Quantum Neural Networks (QNNs):
QNNs leverage the principles of quantum computing to process information. In the context of anomaly detection, QNNs offer the advantage of processing vast amounts of high-frequency data simultaneously, exploiting quantum parallelism. The inherent parallelism of quantum computing can potentially enhance the model's ability to identify subtle patterns indicative of anomalies in the manufacturing process. Additionally, QNNs have shown promise in capturing complex relationships in datasets, making them well-suited for the intricate nature of wafer manufacturing data.

Generative Adversarial Networks (GANs):
GANs, consisting of a generator and a discriminator trained in tandem through adversarial learning, are known for their capability to generate realistic data. In the context of anomaly detection, GANs can be employed to learn the underlying data distribution and generate synthetic samples representative of non-anomalous conditions in the manufacturing process. The discriminator is then tasked with distinguishing between real and generated samples, helping identify anomalies by recognizing deviations from the learned distribution. GANs provide a unique perspective on anomaly detection by creating a dynamic interplay between generative and discriminative processes.

OBJECTIVES:

High-Dimensional Data Handling:
QNNs and GANs are particularly well-suited for handling high-dimensional data such as the time-series IIoT data generated in wafer manufacturing. Their architectures allow for the representation of intricate patterns and relationships in the data.
Imbalanced Data Handling:
Both QNNs and GANs have the potential to handle imbalanced datasets effectively. QNNs can leverage quantum parallelism to discern anomalies even in rare instances, while GANs, through adversarial learning, can adapt to the imbalanced nature of the dataset.
Real-Time Anomaly Detection:
QNNs and GANs can be designed to operate efficiently in real-time, enabling timely anomaly detection as wafers progress through the manufacturing process. Their architectures and training dynamics support the rapid processing of incoming data.
Comparative Analysis:

By implementing both QNNs and GANs, we aim to conduct a comprehensive comparative analysis to evaluate the strengths and weaknesses of each approach in the context of wafer manufacturing anomaly detection. This exploration will contribute insights into the efficacy of quantum computing principles and generative adversarial learning for addressing the challenges posed by rare anomalies in high-frequency industrial datasets

Quantum Neural Network Implementation

1. **Data Pre-processing:**

- Standard scaling was applied to normalize the features, ensuring a mean of 0 and a standard deviation of 1.
- Principal Component Analysis (PCA) reduced the dimensionality of the data to 3 principal components, focusing on the most significant features.

2. **Tuning Parameters:**

- The Quantum Neural Network (QNN) involved a quantum circuit with 3 qubits.
- The exact tuning of quantum gates involved an action on 3 input qubits corresponding to the 3 reduced features achieved after having carried out PCA, the data was encoded with theta values as a quantum circuit parameter which was iteratively optimized as a parameter by the neural network, variationally.

3. **Model Architecture and Optimization:**

3.1. Architecture:

- The QNN is built using Qiskit's EstimatorQNN class, representing a quantum model for estimation tasks.
- It is wrapped in a NeuralNetworkClassifier for classification tasks.

3.2. Optimization:

- The COBYLA optimizer was used with a maximum of 60 iterations. This constraint-optimization method is suitable for noisy, gradient-free optimization often seen in quantum circuits.

4. **Training Parameters**

- Random Seed: Set to 42, ensuring reproducibility in the quantum computing context.
- Data Splitting: 10% of the data was reserved for validation, with stratification to keep the class distribution consistent.
- Training Visualization: A callback function was used for real-time visualization of the training process, tracking the objective function's value.

5. **Performance Parameters**

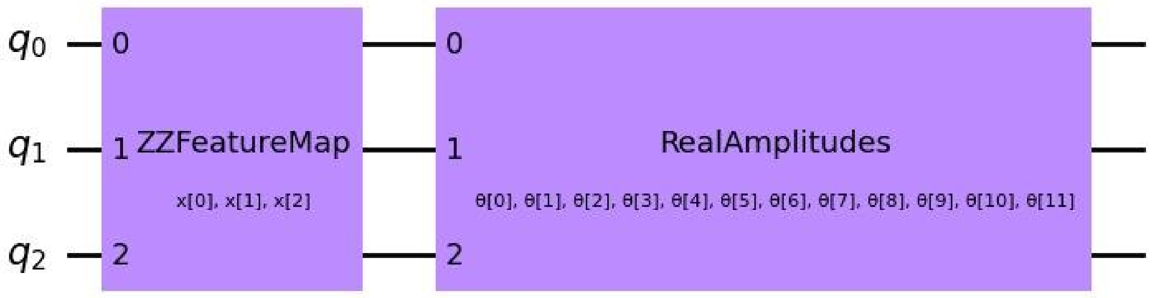
- Accuracy Calculation: Post-training, the model's accuracy was manually calculated by comparing predictions with the validation set.
- Heatmaps: Used to visualize feature correlations both before and after PCA, aiding in understanding feature interactions.

6. **Achieved Performance**

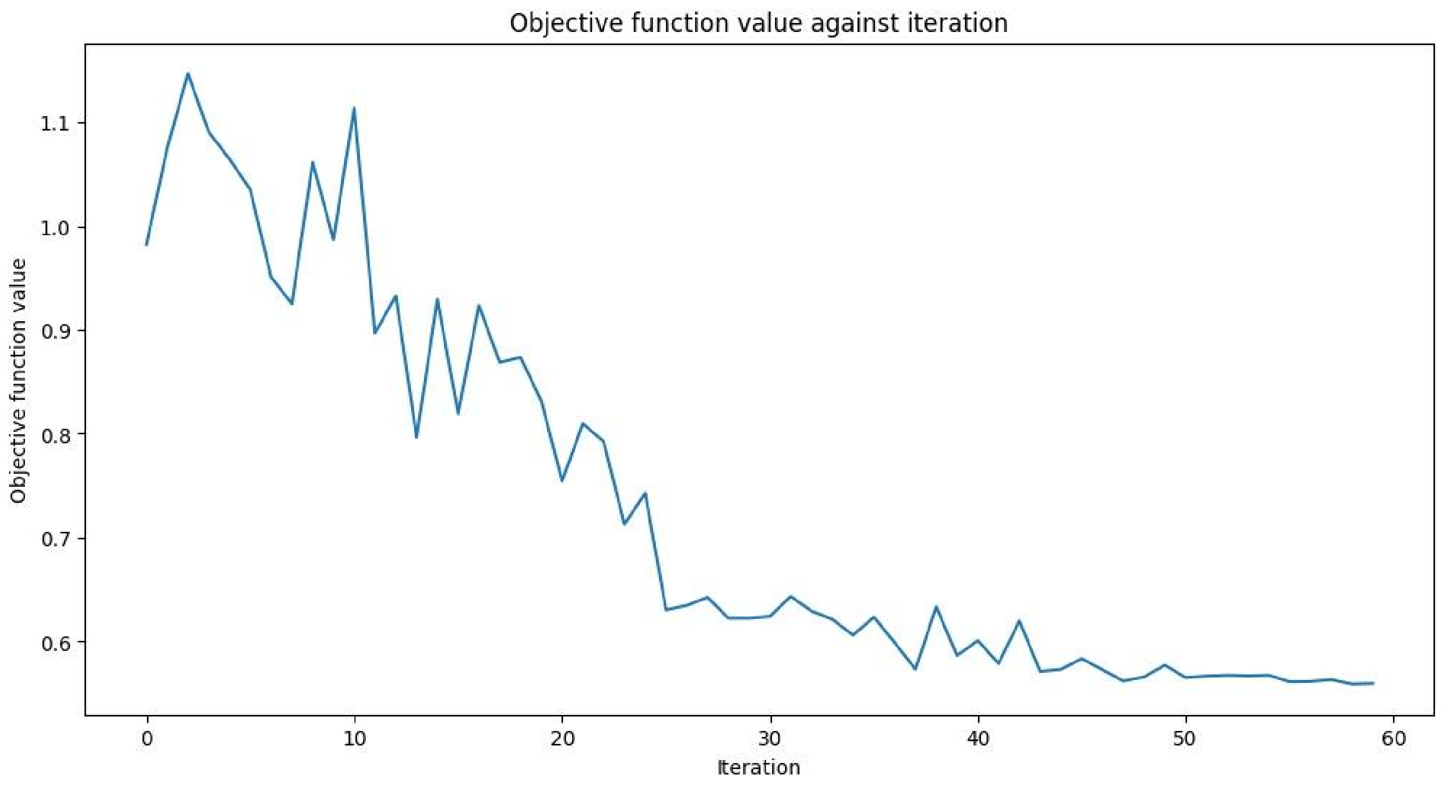
- The model accuracy was calculated and found out to be 82%, being the only performance metric assessed for this project.

Plots and Results of QNN Implementation:

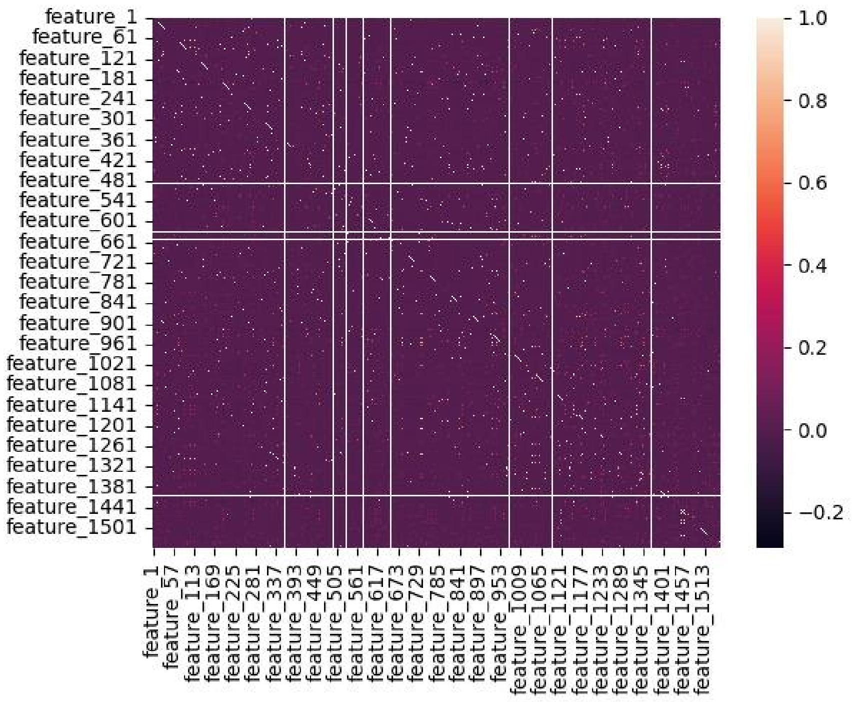
1. **Model Circuit Architecture:**


q_0
q_1
q_2
0
1
2
ZZFeatureMap
x[0], x[1], x[2]
RealAmplitudes
θ[0], θ[1], θ[2], θ[3], θ[4], θ[5], θ[6], θ[7], θ[8], θ[9], θ[10], θ[11]

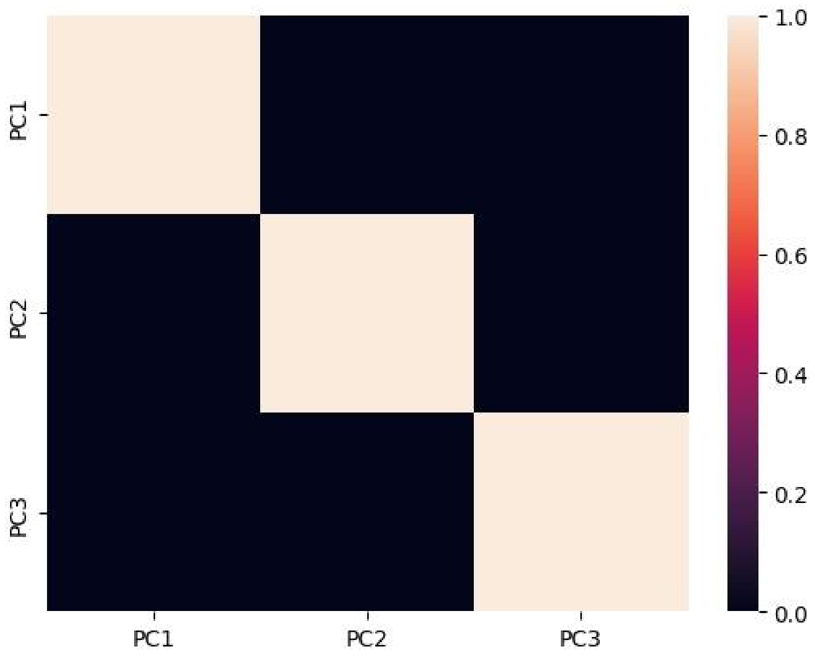
2. **Objective Function Value Vs Iterations**



3. Heatmap before PCA



4. Heatmap after PCA



Additional Observations:
This Project demonstrates a foundational approach to integrating quantum computing with machine learning. However, the intricacies of quantum circuit design are critical for the model's success and require deeper exploration.
Future iterations of the model could benefit from a more detailed exploration of quantum circuit parameters and the integration of hybrid quantum-classical layers, depending on the complexity and nature of the dataset.
Regular evaluation and tuning of the model using quantum-specific metrics and performance measures are recommended for ongoing optimization and validation.

Generative Adversarial Networks (GAN) Implementation

1. **Techniques Applied and Data Pre-processing**

- Data Loading and Pre-processing: The dataset was loaded from a CSV file. A subset of the data where Class is 0 was sampled and separated into a training dataset (df_train). The 'Class' column was removed from this training dataset for unsupervised training.
- Normalization and Reshaping: Data normalization and reshaping was done implicitly in this project. This step is critical for preparing data for neural network models.

2. **Model Architecture and Optimization**
    - Generator Architecture: The generator is a sequential neural network model starting with a dense layer of 1024 neurons, followed by batch normalization and LeakyReLU activation. Additional dense layers with varying neurons are added along with batch normalization and LeakyReLU activations. The generator model is designed to generate data from a noise vector.
    - Discriminator Architecture:The discriminator is a sequential model with dense layers (256, 512, 128 neurons) using LeakyReLU activations and dropout for regularization. The final output is a single neuron, indicating a binary classification typical for GAN discriminators.
    - Model Optimization: The loss function used is Binary Crossentropy, which is standard for binary classification tasks in GANs. Adam optimizer with a learning rate of 1e-4 is used for both the generator and discriminator.

3. **Training Parameters**

- Epochs and Batch Size: The model is trained for 150 epochs with a batch size of 64.
- Noise Dimension: A noise dimension of 128 is used for generating synthetic data in the generator.

4. **Performance Parameters**

- Loss Function: Binary Crossentropy is used for calculating the loss of both generator and discriminator.

- Gradient Updates: Gradients are calculated using tf.GradientTape and applied to update the model weights during training.

5. **Achieved Performance**

- Loss Visualization: The losses of the generator and discriminator are plotted across epochs to visualize the training process.

6. **Model Evaluation:**

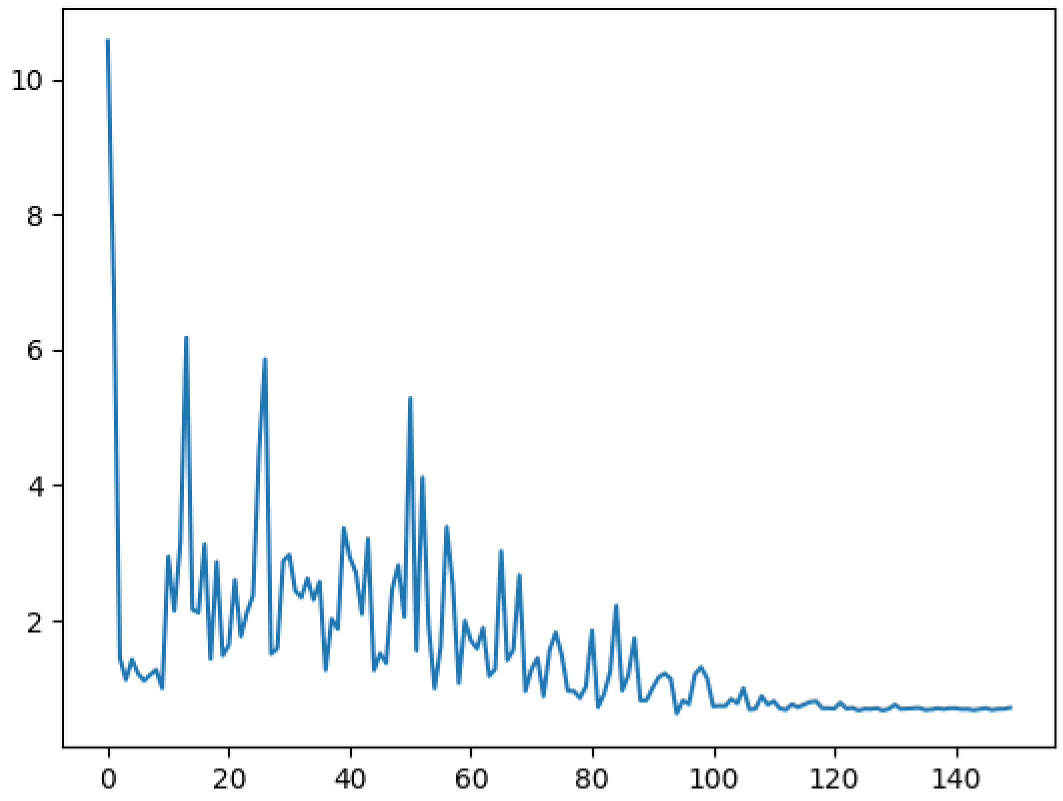
- Predictions are made on a test dataset, and a range of thresholds are used to classify these predictions into binary classes. Performance metrics such as F1 score, accuracy, precision, and recall are calculated and reported for different thresholds.
- For Prob = 0.2 results are:

  f1 score =0.027397260273972605 accuracy=0.5977337110481586
  precision=0.408757437086049 recall=0.013986013986013986

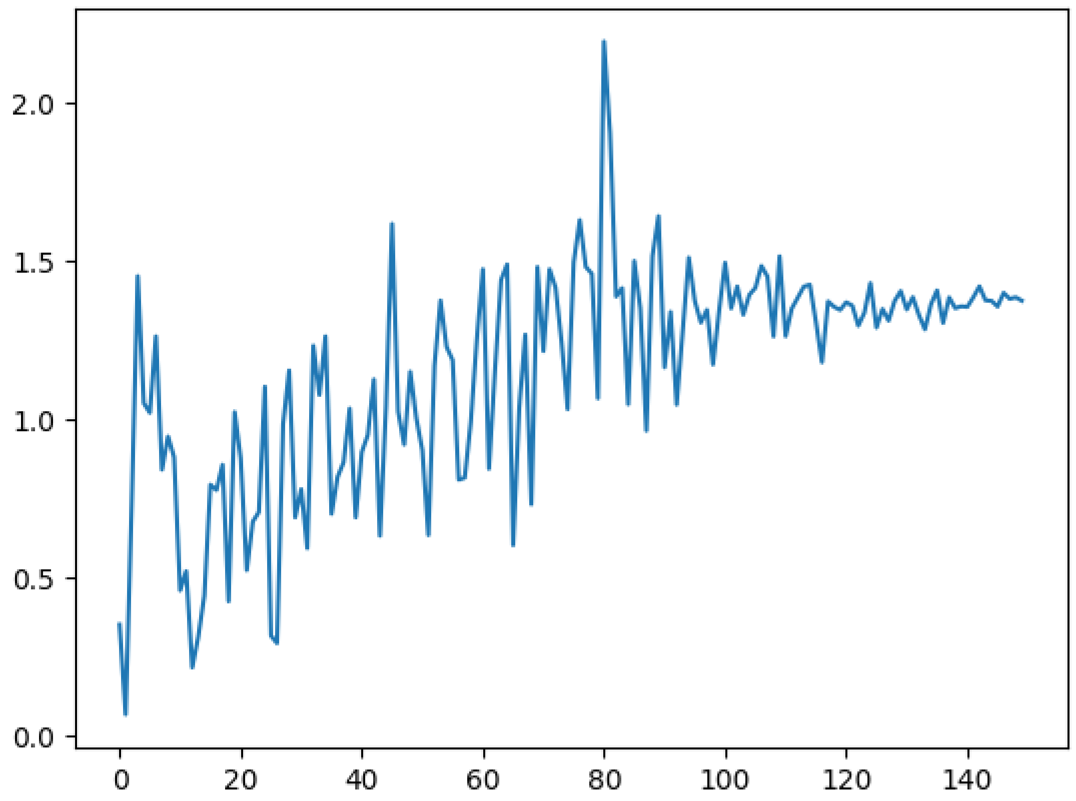
- A confusion matrix is plotted for visual evaluation of the model's performance.

Plots/Results of GAN Implementation

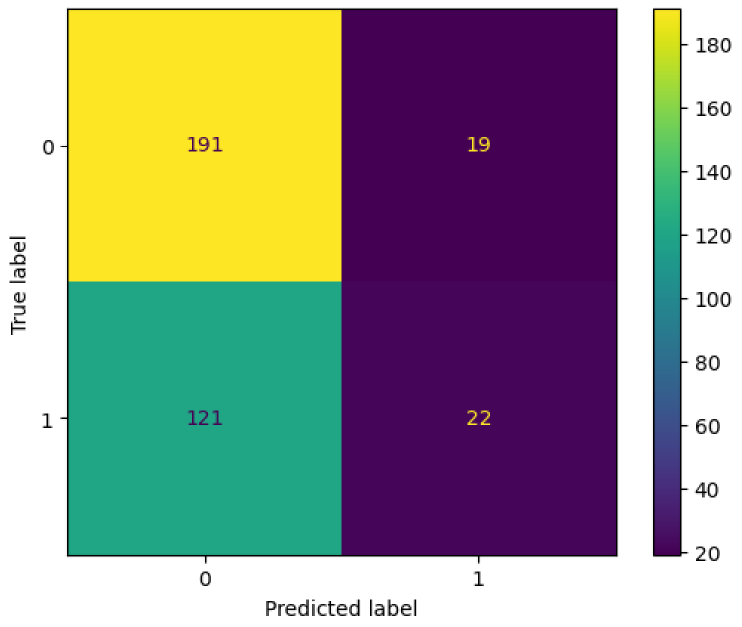
1. **Generator Loss Vs Epochs**



2. Discriminator Loss Vs Epochs



3. Confusion Matrix



Additional Observations: The evaluation suggests that the discriminator can effectively classify the generated images. The variation in performance metrics across different thresholds indicates the model's sensitivity to the classification threshold.

CONCLUSION

The difference in accuracy between the Generative Adversarial Network (GAN) model and the Quantum Neural Network (QNN) model, with the latter outperforming the former (82% vs. 60% accuracy), can be attributed to several factors related to the nature of these models, the data, and their respective implementations:

Model Complexity and Suitability:
QNNs leverage quantum mechanics properties, potentially offering enhanced feature processing capabilities in high-dimensional spaces and more efficient handling of complex patterns in the data.
GANs, while powerful in generating new data and feature extraction, may not be as efficient for direct classification tasks, especially if the discriminator (part of the GAN used for classification) is not optimally tuned.

Data Characteristics:
The nature and complexity of the dataset could inherently favor QNNs. For instance, if the dataset has complex, high-dimensional patterns, QNNs might be better at capturing these due to their quantum state representation.
GANs might underperform if the generated features do not align well with the discriminative aspects of the dataset or if the model focuses on aspects of the data that are less relevant for classification.

Training and Hyperparameter Tuning:
QNNs have been better optimized in terms of hyperparameters, training duration, or learning rate, contributing to higher accuracy.
GANs require careful balancing between the generator and discriminator, and suboptimal training (like mode collapse or failure to converge) can lead to poorer performance.

Imbalanced Dataset:
If the dataset is imbalanced, QNNs inherently manage the imbalance better, especially if specific quantum algorithms are utilized for this purpose.
GANs, while good at generating synthetic data, don’t always generate representative samples of the minority class, leading to less effective learning.

Overfitting and Generalization:
The GAN model might have overfitted to the training data, especially if not regularized properly or if trained for too many epochs.