Analysis of Noise Effects on Chest X-ray Images Using Quantum Fourier Transform in the Frequency Domain

A PROJECT REPORT

Submitted by

Gowripriya R (DL.AI.U4AID24113)
Yaalini R (DL.AI.U4AID24043)
Vepuri Satya Krishna (DL.AI.U4AID24140)

in partial fulfilment for the award of the degree of

BACHELOR OF TECHNOLOGY (B. Tech)
IN ARTIFICIAL INTELLIGENCE AND DATA SCIENCE (AIDS)

Under the guidance of Prof. Dr. Mrittunjoy Guha Majumdar

Submitted to

AMRITA VISHWA VIDYAPEETHAM
AMRITA SCHOOL OF ARTIFICIAL INTELLIGENCE
FARIDABAD – 121002

December 2025

BONAFIDE CERTIFICATE

This is to certify that this project report entitled **"Analysis of Noise Effects on Chest X-ray Images Using Quantum Fourier Transform in the Frequency Domain"** is the Bonafide work of **Gowripriya R, Yaalini R, and Vepuri Satya Krishna**, who carried out the project work under my supervision.

Signature

Prof. Dr. Mrittunjoy Guha Majumdar
School of AI, Faridabad

DECLARATION BY THE CANDIDATE

I declare that the report entitled **"Analysis of Noise Effects on Chest X-ray Images Using Quantum Fourier Transform in the Frequency Domain"** submitted by me for the degree of Bachelor of Technology is the record of the project work carried out by me under the guidance of **Prof. Dr. Mrittunjoy Guha Majumdar** and this work has not formed the basis for the award of any degree, diploma, associateship, fellowship, title in this or any other University or other similar institution of higher learning.

Gowripriya R (DL.AI.U4AID24113)
Yaalini R (DL.AI.U4AID24043)
Vepuri Satya Krishna (DL.AI.U4AID24140)

ACKNOWLEDGMNET

This project work would not have been possible without the contribution of many people. It gives me immense pleasure to express my profound gratitude to our honorable Chancellor Sri Mata Amritanandamayi Devi, for her blessings and for being a source of inspiration. I am indebted to extend my gratitude to our Principal, Dr. Lakshmi Mohandas Amrita School of Computing and Engineering, for facilitating us all the facilities and extended support to gain valuable education and learning experience.

I wish to express my sincere gratitude to my supervisor Prof. Dr. Mrittunjoy Guha Majumdar for his personal involvement and constant encouragement during the entire course of this work.

I am grateful to Project Supervisor, Review Panel Members, and the entire faculty of the Department of Computer Science and Engineering, for their constructive criticisms and valuable suggestions which have been a rich source to improve the quality of this work.

Gowripriya R (DL.AI.U4AID24113)
Yaalini R (DL.AI.U4AID24043)
Vepuri Satya Krishna (DL.AI.U4AID24140)

Contents

1 Introduction

Chest X-ray imaging is a fundamental diagnostic tool in medical practice, widely used for detecting respiratory and cardiac abnormalities. However, these images are susceptible to various types of noise during acquisition or transmission, which can degrade image quality and affect diagnostic accuracy. Analyzing the impact of noise in the frequency domain is crucial for understanding image degradation and developing effective processing techniques.

The emergence of quantum computing introduces new possibilities for image analysis through algorithms like the Quantum Fourier Transform (QFT), which offers potential advantages over classical methods in certain computational tasks. Quantum image processing techniques enable the representation of classical images as quantum states, allowing quantum algorithms to be applied directly to imaging problems.

This project, titled "Analysis of Noise Effects on Chest X-ray Images Using Quantum Fourier Transform in the Frequency Domain", explores the effects of Gaussian and salt-and-pepper noise on a chest X-ray image patch using QFT. By encoding the image data into quantum states and applying QFT, the project examines how different noise types alter the frequency-domain representation in a quantum simulation environment using Qiskit. The study aims to demonstrate the application of quantum computing concepts to medical image analysis and provide insights into noise-induced changes in the quantum frequency spectrum.

2 Objectives

The primary objective of this project is to analyze the effects of different noise types on chest X-ray images in the quantum frequency domain using the Quantum Fourier Transform (QFT). Specifically, the project aims to extract a small 4×4 patch from a grayscale chest X-ray image and encode it into a 4-qubit quantum state through amplitude encoding in Qiskit. Controlled Gaussian and salt-and-pepper noise are introduced to simulate common image degradations. The QFT is then implemented and applied to the quantum representations of the clean, Gaussian-noisy, and salt-and-pepper-noisy patches. By comparing the resulting measurement probability distributions, the project examines how each noise type alters the quantum frequency-domain representation, particularly the introduction of high-frequency components.

This includes:

- Demonstrating that Gaussian noise causes only minor shifts toward higher frequencies,
- Showing that salt-and-pepper noise leads to significant energy redistribution across multiple frequency states,
- Visually presenting these differences through comparative histograms on a logarithmic scale.

Overall, the work highlights the potential of quantum computing techniques for studying noise degradation in medical imaging while aligning the observations with established principles of classical signal processing.

3 Methodology

The project was implemented using Python in a Jupyter Notebook environment with the Qiskit library for quantum circuit simulation. The methodology consists of classical image preprocessing, noise addition, quantum state preparation, application of the Quantum Fourier Transform (QFT), and result analysis.

A grayscale chest X-ray image was loaded using the Pillow library and resized to 64×64 pixels. A 4×4 patch was extracted from the central region to yield 16 pixel values, suitable for encoding into a 4-qubit quantum state via amplitude encoding. The pixel values were flattened and L2-normalized to ensure the vector had unit norm, a requirement for valid quantum amplitudes.

Two types of noise were introduced to separate copies of the patch:

- Gaussian Noise: Additive noise drawn from a normal distribution with mean 0 and standard deviation 10.
- Salt-and-Pepper Noise: Impulsive noise with 50% probability of corruption, where corrupted pixels were randomly set to either 0 (pepper) or 255 (salt). A fixed random seed was used for reproducibility.

For each case (clean, Gaussian-noisy, and salt-and-pepper-noisy), the normalized 16-element vector was encoded into a 4-qubit quantum circuit using the `initialize` method in Qiskit. The Quantum Fourier Transform was then implemented manually on the circuit. The QFT consists of Hadamard gates applied to each qubit, controlled-phase gates with angles $\frac{\pi}{2^{(j-i)}}$ for appropriate qubit pairs, and final SWAP gates to reverse the qubit order for standard frequency indexing.

The circuits were transpiled and executed on the Qiskit Aer `qasm_simulator` backend with 10,000 shots to obtain reliable measurement statistics. Measurement instructions were added to all qubits before execution. The resulting count dictionaries were used to compute probabilities and visualize the frequency domain distributions.

Histograms of the measurement outcomes were plotted using Qiskit's `plot_histogram` function. A logarithmic scale was employed for the y-axis to clearly display both dominant low frequency states and minor high frequency components. A final comparative plot with three subpanels (clean, Gaussian, and salt-and-pepper) was generated to highlight the differences in frequency-domain behavior.

All experiments were performed through classical simulation of quantum circuits, as the small scale (4 qubits) allowed efficient execution on standard hardware. The methodology focused on accurate quantum encoding and QFT application to enable direct comparison of noise-induced changes in the quantum frequency domain representation.

4 Results And Discussion

This section presents and analyzes the effects of different noise models on the quantum frequency-domain representation of a chest X-ray image patch using the Quantum Fourier Transform (QFT). A 4×4 grayscale patch was extracted from a chest X-ray image and evaluated under three conditions: clean (noise-free), Gaussian noise, and salt-and-pepper noise. The objective is to observe how each noise type influences the distribution of frequency components in the quantum domain.

4.1 Original and Noisy Image Patches

Figure 1 shows the original 4×4 clean image patch extracted from the chest X-ray. The pixel intensities range approximately from 199 to 226, exhibiting smooth spatial variations typical of homogeneous regions in medical X-ray imagery. Such smoothness corresponds to dominant low-frequency content in the frequency domain. Figures 2 illustrates the noisy versions of the same patch. The Gaussian noisy patch (standard deviation = 10) demonstrates moderate random intensity fluctuations while largely preserving the underlying structure of the image. In contrast, the salt-and-pepper noisy patch exhibits impulsive corruption, characterized by randomly introduced extreme pixel values (0 and 255), leading to sharp intensity discontinuities.

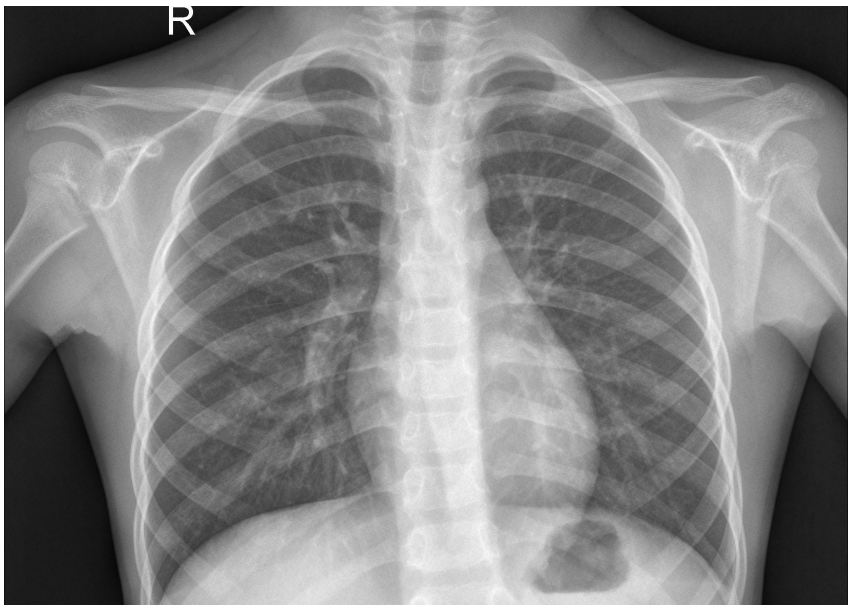
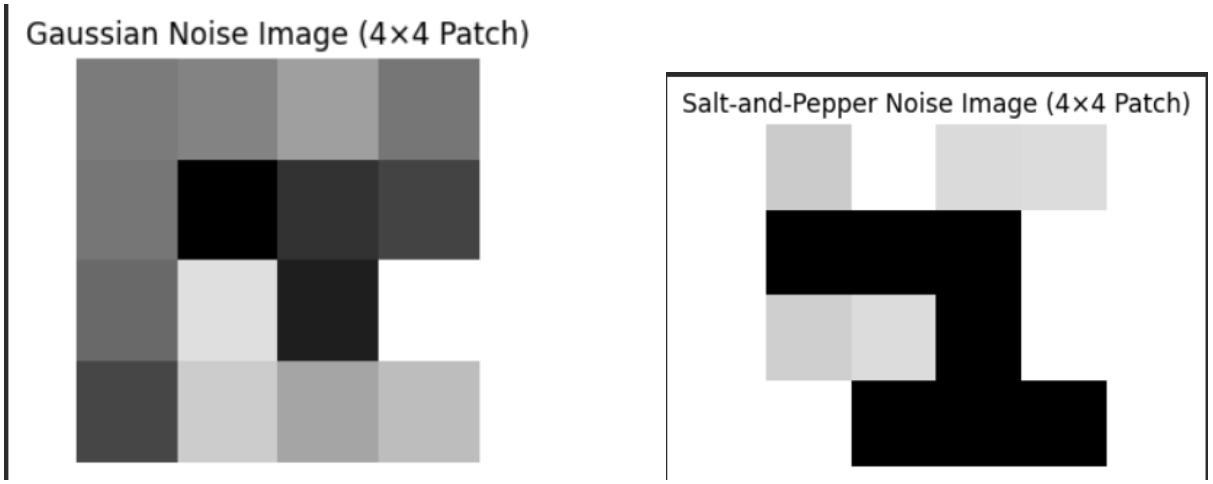


Figure 1: Original image



(a) Gaussian noisy patch (b) Salt-and-pepper noisy patch

Figure 2: Noisy 4×4 Image Patches

4.2 Quantum Fourier Transform Analysis

Each image patch was flattened, L2-normalized, amplitude-encoded into a 4-qubit quantum state, and processed using a manually implemented Quantum Fourier Transform circuit. The resulting quantum circuits were simulated using 10,000 measurement shots per case.

Figure 3 presents the individual QFT measurement histograms for the clean, Gaussian noisy, and salt-and-pepper noisy images, along with zoomed insets to emphasize low-probability states.

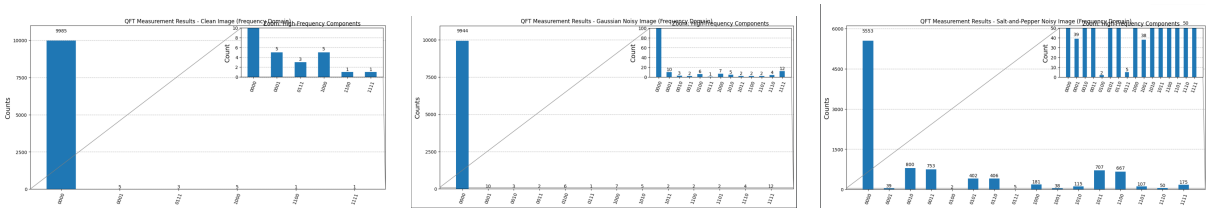
For the clean image, the measurement outcomes are overwhelmingly concentrated in the —0000⟩ basis state, which accounts for approximately 9985 counts (99.85% probability). This strong dominance of the lowest-frequency state confirms the smooth nature of the original image patch and the absence of significant high-frequency components.

In the case of Gaussian noise, the —0000⟩ state remains dominant with approximately 9944 counts (99.44% probability). Only minor leakage into higher-frequency states is observed, such as —1111⟩ and —0001⟩, each with negligible counts. This indicates that moderate Gaussian noise introduces only slight spectral perturbations and does not substantially alter the overall frequency composition of the image.

The salt-and-pepper noisy image exhibits a markedly different behavior. The probability of the —0000⟩ state drops significantly to approximately 5553 counts (55.53%). The remaining probability mass is distributed across multiple mid- and high-frequency states,

including —0010⟩, —0011⟩, —1011⟩, and —1100⟩, each with several hundred counts. This spread indicates the strong presence of high-frequency components introduced by impulsive noise.

The zoomed insets in Figure 3 are essential for visualizing these low-count frequency components, which are otherwise obscured by the dominant —0000⟩ peak when plotted on a linear scale.



(a) Clean Image Patch (b) Gaussian Noisy Image Patch (c) Salt-and-Pepper Noisy Image Patch

Figure 3: Frequency-Domain Representations

4.3 Comparative Frequency-Domain Behavior

Figure 4 presents a comparative visualization of the QFT measurement results on a logarithmic scale. This representation clearly highlights the contrast between the noise models. Both the clean and Gaussian noisy cases show an overwhelming concentration of probability in the lowest-frequency state, reinforcing the preservation of smooth spatial characteristics. In contrast, the salt-and-pepper noise case demonstrates a pronounced spread across mid- and high-frequency states, reflecting the presence of abrupt pixel-level discontinuities.

These observations are consistent with classical Fourier analysis. Gaussian noise introduces relatively smooth, distributed perturbations that minimally affect high-frequency content when the noise variance is moderate relative to the signal intensity. Conversely, salt-and-pepper noise introduces sharp transitions, resulting in significant high-frequency spectral components. The minimal difference between the clean and Gaussian noisy cases can be attributed to the relatively high baseline pixel intensities of the image patch and the chosen noise standard deviation, which leads to subtle variations after normalization.

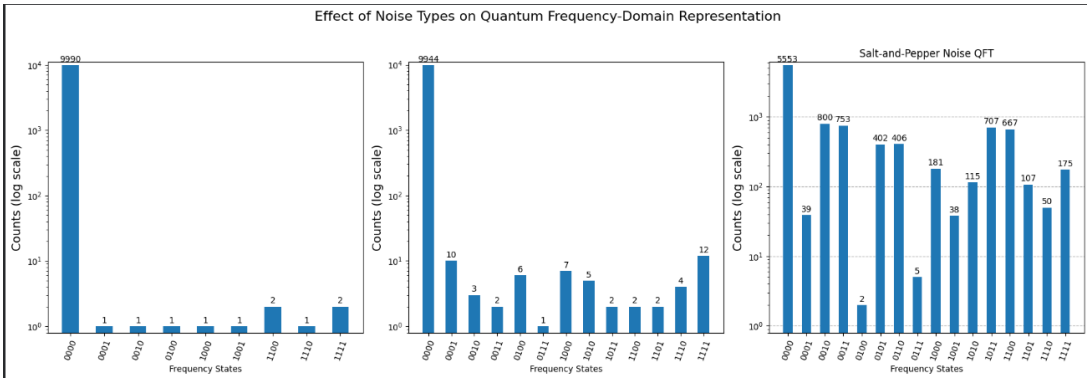


Figure 4: Comparison

4.4 Quantum Circuit Implementation

Figure 5 shows the 4-qubit Quantum Fourier Transform circuit used in this work. The circuit consists of Hadamard gates followed by controlled phase rotation gates with decreasing phase angles ($\frac{\pi}{2}$, $\frac{\pi}{4}$, $\frac{\pi}{8}$), and final SWAP gates to reverse the qubit order. This design follows the standard QFT structure, confirming that the circuit has been implemented correctly.

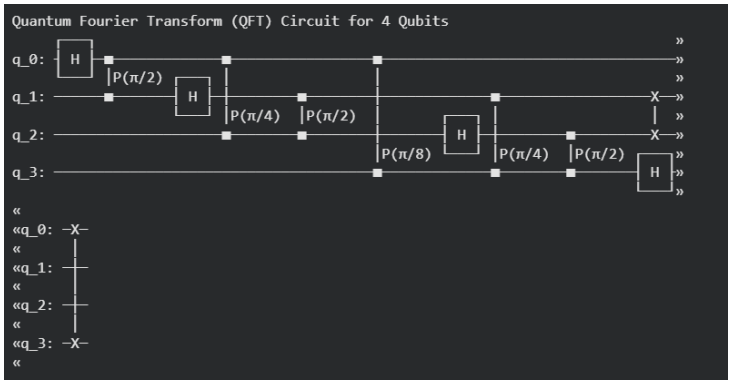


Figure 5: Implemented 4-Qubit Quantum Fourier Transform (QFT) Circuit

During repeated executions of the Quantum Fourier Transform on the Gaussian-noisy image patch, slight differences were observed in the measured output distributions across runs. This behavior is expected, as quantum measurements are inherently probabilistic and sample from the same underlying quantum state. Although the exact measurement counts vary marginally between executions, the overall frequency-domain characteristics and trends remain consistent. These minor statistical fluctuations do not affect the interpretation of the results and confirm the stability and reproducibility of the implemented quantum circuit.

5 Conclusion

In this project, we explored how the Quantum Fourier Transform (QFT) can be used to study the effect of different noise types on a chest X-ray image patch in the quantum frequency domain. By comparing a clean image with Gaussian-noisy and salt-and-pepper–noisy versions, we were able to clearly observe how noise influences the distribution of frequency components after QFT.

The results show that the clean chest X-ray patch is dominated by low-frequency information, with almost all the measurement probability concentrated in the $|0000\rangle$ state. When Gaussian noise, $\sigma = 10$ is added, this overall behavior remains largely unchanged, with only very small contributions from higher-frequency states. On the other hand, salt-and-pepper noise has a much stronger impact, significantly spreading the probability across multiple mid- and high-frequency states and reducing the $|0000\rangle$ probability to around 55–56%.

These findings are consistent with what is expected from classical signal processing, where impulsive noise introduces sharp intensity changes and therefore stronger high-frequency components compared to smooth additive noise. The QFT captures this behavior effectively at the quantum level through the resulting measurement distributions.

This work also confirms the correct implementation of amplitude encoding and the manually constructed QFT circuit in Qiskit. Although the experiments were carried out using classical simulation on a small 4-qubit system, the results provide a clear proof of concept and highlight the potential of quantum algorithms for frequency-domain analysis of medical images.

Overall, the project demonstrates how quantum computing techniques can offer meaningful insights into noise effects in medical imaging, serving both as a learning exercise and as a stepping stone toward more advanced quantum-enhanced image processing applications in the future.

6 References

1. Qiskit Documentation, "Quantum Fourier Transform and its Applications," IBM Quantum, 2024. [Online].
https://docs.qiskit.org/api/qiskit/circuit.library.QFT

2. Y. Zhang et al., "Quantum Image Processing: Opportunities and Challenges," in Quantum Image Processing, Springer, 2020, pp. 1–25.

3. S. Caraiman and V. Manta, "Image Processing Using Quantum Fourier Transform," in Proceedings of the International Conference on Quantum Computing, 2018.

4. A. Al-Ta'ani and N. Alqudah, "Implementation and Analysis of Quantum Fourier Transform in Image Processing," Jordan Journal of Electrical Engineering, vol. 5, no. 2, pp. 112–120, 2019.

5. W. Huda, "Noise in Radiographic Imaging," American Journal of Roentgenology, vol. 204, no. 1, pp. W1–W8, 2015. [Online].
https://www.ajronline.org/doi/full/10.2214/AJR.14.13116

7 Appendix

```
!pip install qiskit qiskit-aer --quiet
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image
from qiskit import QuantumCircuit, transpile
from qiskit_aer import Aer
from qiskit.visualization import plot_histogram

# Load image
img = Image.open("/content/chestpqip.jpeg").convert("L")
plt.figure(figsize=(5,5))
plt.imshow(img, cmap="gray")
plt.title("Original Chest X-ray Image")
plt.axis("off")
plt.show()

# Resize image
img_resized = img.resize((64, 64))

# Extract 4x4 patch from center
w, h = img_resized.size
patch = img_resized.crop((30, 30, 34, 34))
patch_array = np.array(patch)
plt.figure(figsize=(3,3))
plt.imshow(patch_array, cmap="gray")
plt.title("4 4  Image Patch")
plt.axis("off")
plt.show()
patch_array

# Normalize pixel values
patch_norm = patch_array / np.linalg.norm(patch_array)
patch_norm

# Add Gaussian noise
gaussian_noise = np.random.normal(0, 10, patch_array.shape)
gaussian_img = patch_array + gaussian_noise

# Show Gaussian noisy image
```

```
plt.figure(figsize=(3,3))
plt.imshow(gaussian_img, cmap="gray")
plt.title("Gaussian Noise Image (4 4  Patch)")
plt.axis("off")
plt.show()

# Print original and noisy matrices
print("Original 4 4  Image Patch Matrix:")
print(patch_array)
print("\nGaussian Noise Matrix:")
print(gaussian_noise)
print("\nGaussian Noisy Image Matrix:")
print(gaussian_img)

# Fix random seed for reproducibility
np.random.seed(42)

# Salt-and-pepper noise
sp_img = patch_array.copy()
prob = 0.5
for i in range(sp_img.shape[0]):
    for j in range(sp_img.shape[1]):
        r = np.random.rand()
        if r < prob/2:
            sp_img[i, j] = 0
        elif r > 1 - prob/2:
            sp_img[i, j] = 255

# Display noisy image
plt.figure(figsize=(3,3))
plt.imshow(sp_img, cmap="gray")
plt.title("Salt-and-Pepper Noise Image (4 4  Patch)")
plt.axis("off")
plt.show()

# Print matrices
print("Original 4 4  Image Patch Matrix:")
print(patch_array)
print("\nSalt-and-Pepper Noisy Image Matrix:")
print(sp_img)

```

```
from qiskit.quantum_info import Statevector

def prepare_quantum_state(patch_array):
    data = patch_array.flatten().astype(float)
    data = data / np.linalg.norm(data)
    qc = QuantumCircuit(4)
    qc.initialize(data, range(4))
    qc = qc.decompose()
    state = Statevector.from_instruction(qc)
    print("\nQuantum Statevector (Amplitude Encoding):")
    for i, amp in enumerate(state.data):
        print(f"|{format(i, '04b')}> : {amp}")
    return qc

def apply_qft(qc):
    n = qc.num_qubits
    for qubit in range(n):
        for j in range(qubit):
            angle = np.pi / (2 ** (qubit - j))
            qc.cp(angle, qubit, j)
        qc.h(qubit)
    for i in range(n // 2):
        qc.swap(i, n - i - 1)
    return qc

def execute_qft(qc, shots=10000):
    qc_meas = qc.copy()
    qc_meas.measure_all()
    backend = Aer.get_backend("aer_simulator")
    tqc = transpile(qc_meas, backend)
    result = backend.run(tqc, shots=shots).result()
    return result.get_counts()

def analyze_qft_results(counts, title):
    total_shots = sum(counts.values())
    print(f"\n=== {title} ===")
    print(f"Total shots: {total_shots}")
    print(f"Distinct states measured: {len(counts)}")
    probabilities = {k: v/total_shots for k, v in counts.items()}
    sorted_probs = sorted(probabilities.items(), key=lambda x: x
       [1], reverse=True)
```

```
    print("\nTop 5 dominant frequency states:")
    for state, prob in sorted_probs[:5]:
        print(f" |{state}> : {prob:.4f}")
    return probabilities

# QFT on clean patch (raw pixels for demonstration)
qc_clean = prepare_quantum_state(patch_array)
qc_clean = apply_qft(qc_clean)
counts_clean = execute_qft(qc_clean, shots=10000)
clean_probs = analyze_qft_results(counts_clean, "Clean Image QFT Analysis")

# QFT on normalized clean patch
qc_clean = prepare_quantum_state(patch_norm)
qc_clean = apply_qft(qc_clean)
qc_clean.measure_all()
backend = Aer.get_backend('qasm_simulator')
qc_clean = transpile(qc_clean, backend)
result_clean = backend.run(qc_clean, shots=10000).result()
counts_clean = result_clean.get_counts()

print("\nRaw measurement counts:")
print(counts_clean)

fig, ax = plt.subplots(figsize=(12, 6))
plot_histogram(counts_clean, ax=ax)
ax.set_title("QFT Measurement Results - Clean Image (Frequency Domain)")
ax.set_ylabel("Counts")

from mpl_toolkits.axes_grid1.inset_locator import inset_axes, mark_inset
inset_ax = inset_axes(ax, width="40%", height="30%", loc='upper right')
plot_histogram(counts_clean, ax=inset_ax)
inset_ax.set_ylim(0, 10)
inset_ax.set_title("Zoom: High-Frequency Components")
mark_inset(ax, inset_ax, loc1=2, loc2=4, fc="none", ec="0.5")
plt.tight_layout()
plt.show()
```

```
158 total_shots = sum(counts_clean.values())
159 print("\nTop probability states (should show dominant low
       frequencies):")
160 sorted_counts = sorted(counts_clean.items(), key=lambda x: x[1],
       reverse=True)
161 for state, count in sorted_counts[:8]:
162     print(f"|{state}> : {count} counts ({count/total_shots:.4f}
          probability)")
163
164 # Gaussian Noise QFT
165 gaussian_flat = gaussian_img.flatten().astype(float)
166 gaussian_norm = gaussian_flat / np.linalg.norm(gaussian_flat)
167 qc_gauss = prepare_quantum_state(gaussian_norm)
168 qc_gauss = apply_qft(qc_gauss)
169 qc_gauss.measure_all()
170 qc_gauss = transpile(qc_gauss, backend)
171 result_gauss = backend.run(qc_gauss, shots=10000).result()
172 counts_gauss = result_gauss.get_counts()
173
174 print("\nRaw measurement counts (Gaussian Noise):")
175 print(counts_gauss)
176
177 total_shots = sum(counts_gauss.values())
178 print("\nTop probability states (Gaussian Noise):")
179 sorted_gauss = sorted(counts_gauss.items(), key=lambda x: x[1],
       reverse=True)
180 for state, count in sorted_gauss[:8]:
181     print(f"|{state}> : {count} counts ({count/total_shots:.4f}
          probability)")
182
183 fig, ax = plt.subplots(figsize=(12, 6))
184 plot_histogram(counts_gauss, ax=ax)
185 ax.set_title("QFT Measurement Results - Gaussian Noisy Image (
       Frequency Domain)")
186 ax.set_ylabel("Counts")
187
188 inset_ax = inset_axes(ax, width="40%", height="30%", loc='upper
       right')
189 plot_histogram(counts_gauss, ax=inset_ax)
190 inset_ax.set_ylim(0, 100)
191 inset_ax.set_title("Zoom: High-Frequency Components")
```

```
mark_inset(ax, inset_ax, loc1=2, loc2=4, fc="none", ec="0.5")
plt.tight_layout()
plt.show()

# Salt-and-Pepper Noisy Image QFT
sp_flat = sp_img.flatten().astype(float)
sp_norm = sp_flat / np.linalg.norm(sp_flat)
if np.linalg.norm(sp_norm) == 0:
    sp_norm += 1e-10
sp_norm /= np.linalg.norm(sp_norm)

qc_sp = prepare_quantum_state(sp_norm)
qc_sp = apply_qft(qc_sp)
qc_sp.measure_all()
qc_sp = transpile(qc_sp, backend)
result_sp = backend.run(qc_sp, shots=10000).result()
counts_sp = result_sp.get_counts()

print("\nRaw measurement counts (Salt-and-Pepper):")
print(counts_sp)

fig, ax = plt.subplots(figsize=(12, 6))
plot_histogram(counts_sp, ax=ax)
ax.set_title("QFT Measurement Results - Salt-and-Pepper Noisy
    Image (Frequency Domain)")
ax.set_ylabel("Counts")

inset_ax = inset_axes(ax, width="40%", height="30%", loc='upper
    right')
plot_histogram(counts_sp, ax=inset_ax)
inset_ax.set_ylim(0, 50)
inset_ax.set_title("Zoom: High-Frequency Components")
mark_inset(ax, inset_ax, loc1=2, loc2=4, fc="none", ec="0.5")
plt.tight_layout()
plt.show()

total_shots = sum(counts_sp.values())
print("\nTop probability states (Salt-and-Pepper):")
sorted_counts = sorted(counts_sp.items(), key=lambda x: x[1],
    reverse=True)
for state, count in sorted_counts[:8]:
```

```
    print(f"|{state}> : {count} counts ({count/total_shots:.4f}
        probability)")

# Comparative Plot
fig, axes = plt.subplots(1, 3, figsize=(18, 6))
counts_list = [counts_clean, counts_gauss, counts_sp]
titles = ["Clean Image QFT", "Gaussian Noise QFT", "Salt-and-
    Pepper Noise QFT"]
for ax, counts, title in zip(axes, counts_list, titles):
    plot_histogram(counts, ax=ax, title=title)
    ax.set_yscale('log')
    ax.set_ylim(bottom=0.8)
    ax.set_xlabel("Frequency States")
    ax.set_ylabel("Counts (log scale)")
plt.suptitle("Effect of Noise Types on Quantum Frequency-Domain
    Representation", fontsize=16, y=1.02)
plt.tight_layout()
plt.show()

# Text-based QFT Circuit Diagram
qc_qft = QuantumCircuit(4)
qc_qft = apply_qft(qc_qft)
print("Quantum Fourier Transform (QFT) Circuit for 4 Qubits ")
print(qc_qft.draw('text'))
```