Report on

Compressed Sensing Image Encryption using CNN and GAN

Submitted to the Department of Information Technology

For the partial fulfillment of the degree of Bachelor of Technology in Information Technology

Presented By:

Satvik Wazir(2021ITB018)

Shridhar Reddy(2021ITB023)

Suryansh Pandey(2021ITB095)

Under the supervision of

Prof. Shyamalendu Kandar

Department of Information Technology

IIEST, Shibpur

Department of Information Technology

Indian Institute of Engineering Science and Technology, Shibpur

December, 2024

Department Of Information Technology
Indian Institute Of Engineering Science And Technology
Shibpur, Howrah - 711103

CERTIFICATE

This is to certify that the project entitled “**Compressed Sensing Image Encryption using CNN and GAN** ” submitted by Satvik Wazir (2021ITB018), Shridhar Reddy (2021ITB023) and Suryansh Pandey(2021ITB095) to Indian Institute of Engineering Science and Technology, Shibpur, towards partial fulfillment of the requirement for the award of degree of Bachelor of Technology in Information Technology is a record of bonafide work carried out by them under my supervision. This dissertation, in my opinion, is worthy of consideration for the purpose for which it has been submitted, and it fulfills the requirements of the regulations of this institute. The results incorporated in this dissertation are original and have not been submitted to any other university or institute for the award of any degree or diploma.

Prof. Shyamalendu Kandar
Assistant Professor
Department of Information Technology
Indian Institute of Engineering Science and Technology
Shibpur,Howrah,India – 711103

December , 2024

ACKNOWLEDGEMENT

We would like to extend our heartfelt gratitude to Prof. Shyamalendu Kandar, our esteemed supervisor, for his invaluable guidance and steadfast support throughout the development of our report on **" Compressed Sensing Image Encryption using CNN and GAN "**. Prof. Kandar's expertise, encouragement, and constructive feedback have been instrumental in shaping the conceptualization, methodology, and execution of our project.

We are also thankful to the Department of Information Technology at the Indian Institute of Engineering Science and Technology Shibpur for providing the necessary resources and a conducive environment for research and learning. The collaborative spirit and academic excellence within the department have significantly contributed to the successful progression of our project.

This report reflects the culmination of our dedicated efforts and the mentorship provided by Prof. Kandar and the supportive academic community at our institution. We acknowledge the importance of this collaborative environment in fostering our understanding of machine learning algorithms for Compressed Sensing and Encryption.

Satvik Wazir
(2021ITB018)

Shridhar Reddy
(2021ITB023)

Suryansh Pandey
(2021ITB095)

INDEX

CHAPTER 1

1. Introduction

1.1. Brief Overview of Image Compression and Encryption

Image compression and encryption are crucial techniques in the modern digital world, enabling efficient storage and secure transmission of image data. The integration of these techniques offers a comprehensive solution to ensure both data optimization and protection.

Purpose: To reduce storage requirements and safeguard image data from unauthorized access, ensuring efficient and secure handling of visual information.

Applications:
i. **Digital Communications**: Compressing images for efficient transmission and encrypting them for secure delivery over networks.
ii. **Medical Imaging**: Protecting sensitive patient data in medical records.
iii. **Cloud Storage**: Ensuring secure and efficient storage of images in cloud databases.
iv. **Multimedia Applications**: Optimizing images for streaming platforms while protecting intellectual property.

Structure: The process combines various computational techniques, each playing a significant role in achieving the goals:
i. **Image Loading**: Initial step of loading images in a format suitable for processing.
ii. **Discrete Wavelet Transform (DWT)**: Decomposes the image into frequency components, separating critical details from redundant data.
iii. **Discrete Cosine Transform (DCT)**: Converts spatial data into frequency coefficients, emphasizing low-frequency data for compression.
iv. **Chaotic Map Generation**: Creates a pseudo-random matrix to introduce unpredictability in the encryption process.
v. **XOR-Based Encryption**: Secures image data by scrambling frequency coefficients using chaotic maps.
vi. **Inverse Transformations**: Restores the compressed and encrypted data to its original form during decryption.
vii. **Quality Metrics Evaluation**: Assesses the reconstructed image's fidelity using metrics like PSNR and SSIM.
The combination of these steps forms a cohesive methodology that ensures efficient compression and secure encryption.

1.2. Need for Image Compression and Encryption

Predicting and implementing effective compression and encryption models for images have significant benefits, particularly in the realms of digital communication, cloud storage, and secure data transmission. The need for such a system arises from the following considerations:

---

1. **Enhanced Data Security**
    - **Protection Against Unauthorized Access**: Encryption ensures that sensitive data remains confidential during transmission and storage.
    - **Resistance to Cyber Threats**: Safeguards image data from tampering or unauthorized interception.
2. **Optimized Storage Solutions**
    - **Reduced Storage Space**: Compression reduces file size, enabling efficient utilization of storage resources.
    - **Cost Efficiency**: Minimizes storage and bandwidth costs by optimizing data size.
3. **Faster Transmission**
    - **Improved Speed**: Smaller file sizes ensure faster data transmission over networks.
    - **Latency Reduction**: Enables quicker access and retrieval of image data.
4. **Real-Time Applications**
    - **Medical Imaging**: Quick and secure sharing of high-quality images for diagnostics and treatment.
    - **Remote Sensing**: Efficient transmission of satellite images for real-time monitoring and analysis.
5. **Quality Maintenance**
    - **High Fidelity**: Advanced algorithms ensure minimal loss of quality during compression and decryption.
    - **User Experience**: Maintains image clarity for end users across applications.
6. **Scalability for Advanced Technologies**
    - **Integration with AI**: Compatible with AI models for enhanced reconstruction and analysis.
    - **Future-Proof Solutions**: Adaptable to emerging technologies in data management and security.

CHAPTER 2

2.1 Use of Python Libraries and Frameworks

Python Libraries
To achieve the objectives of image compression and encryption, several Python libraries are employed to facilitate key computational processes and ensure efficient execution.
Key Libraries:

1. **NumPy**:
   - Used for numerical operations, matrix manipulations, and handling large datasets efficiently.
   - Facilitates operations on image data and supports transformations such as DWT and DCT.
2. **OpenCV (cv2)**:
   - Used for image loading, manipulation, and preprocessing.
   - Supports reading images as grayscale for simplified processing.
3. **PyWavelets (pywt)**:
   - Performs Discrete Wavelet Transform (DWT) and its inverse (IDWT) for image decomposition and reconstruction.
4. **SciPy FFTPack**:
   - Computes the Discrete Cosine Transform (DCT) and Inverse DCT (IDCT) for compressing image data and restoring it.
5. **Matplotlib**:
   - Visualizes images, histograms, and transformations, aiding in analyzing changes across steps.
6. **Scikit-Image (skimage)**:
   - Evaluates the quality of the reconstructed image using Structural Similarity Index (SSIM).

2.2 Image Compression and Encryption

Compression (DWT and DCT)

Discrete Wavelet Transform (DWT):

- Decomposes an image into approximation coefficients (low-frequency) and detail coefficients (high-frequency).
- Essential for reducing redundant data while preserving critical visual information.

Discrete Cosine Transform (DCT):

- Converts spatial domain image data into frequency coefficients.
- Low-frequency coefficients are prioritized for reconstruction, while high-frequency coefficients are discarded for compression.

Encryption (Chaotic Maps and XOR Operation)
Chaotic Map Generation:

- Generates pseudo-random matrices using sinusoidal and cosine-based iterative maps.
- Introduces unpredictability and randomness into the encryption process.

XOR Operation:

- Applies an XOR operation between the chaotic map and DCT coefficients to scramble the image.
- Ensures security by requiring a key for decryption.

Reconstruction

- Uses IDCT and IDWT to restore compressed and encrypted data to its original form.

2.3 Dataset and Route

Dataset Selection
The project uses datasets containing grayscale images for compression and encryption. These images are smaller in size and simpler to process, making them ideal for transformations and evaluations.

Trips in the Context of Compression

- **Input Trip**: Represents the original image data entering the compression and encryption process.
- **Output Trip**: The decrypted and reconstructed image after processing, evaluated using metrics like MSE, PSNR, and SSIM.

Routes in the Context of Encryption

- **Input Route**: The raw data flow through compression techniques (DWT, DCT).
- **Output Route**: Encrypted data processed through chaotic maps and XOR operation, leading to the final secure output.

2.4 Integration of Open Datasets for Images

Open Image Datasets
Open-source image repositories like **MNIST** and other freely available databases are utilized. These datasets contain grayscale images that are ideal for compression and encryption experiments.

Importing Images for Processing
Images are loaded and preprocessed using OpenCV. Essential transformations and processing steps are applied based on the project workflow.

CHAPTER 3

Problem Definition

3.1 Challenges in Image Compression and Encryption

Efficient image compression and encryption are critical for secure and optimized data handling. Various challenges arise when balancing the requirements for reduced file size, image quality preservation, and data security.

---

1. Compression Challenges

a) Achieving High Compression Ratios Without Quality Loss
The challenge lies in significantly reducing the file size while maintaining an acceptable level of visual fidelity in the decompressed image.

- Lossy compression techniques may introduce artifacts that degrade the image quality.
- Balancing compression efficiency and image usability is essential**.**

b) Handling High-Resolution Images
Compressing large images involves higher computational complexity, requiring efficient algorithms and hardware resources.

c) Real-Time Processing Requirements
Applications like streaming platforms demand rapid compression and decompression to ensure seamless user experiences.

---

2. Encryption Challenges

a) Robustness Against Attacks

- Encryption algorithms must resist brute force, statistical, and differential attacks.
- Chaotic maps and XOR operations need to be carefully designed to maximize unpredictability and security.

b) Computational Efficiency
Encrypting large datasets, such as high-resolution images, can be resource-intensive, requiring optimized implementations for practical usability.

c) Data Integrity During Decryption

- Maintaining data integrity during the decryption process is critical to ensure the original image is accurately reconstructed.
- Minimizing errors introduced during compression or transmission is crucial.

---

3. Combined Challenges in Compression and Encryption

a) Synchronizing Compression and Encryption Steps

The integration of compression and encryption must ensure that both processes are complementary, without one undermining the effectiveness of the other.

b) Error Accumulation

- Lossy compression may introduce errors that can propagate during the encryption and decryption stages, affecting reconstruction quality.
- Error metrics must evaluate the cumulative impact of both processes.

3.2 Threats to Image Data Security

1. Unauthorized Access

- Hackers or unauthorized users may intercept image data during transmission or access it from storage**.**

2. Data Tampering

- Malicious actors might modify images, compromising their authenticity and reliability.

3. Cyber Attacks

- Distributed denial-of-service (DDoS) or man-in-the-middle (MITM) attacks may target image transmission pipelines.

4. Hardware Vulnerabilities

- Hardware-level vulnerabilities, such as side-channel attacks, may compromise the encryption process.

3.3 Response to Challenges

1. Advanced Compression Techniques
The project utilizes Discrete Wavelet Transform (DWT) and Discrete Cosine Transform (DCT) to achieve high compression ratios while preserving critical details of the image.

2. Robust Encryption Methods
Chaotic maps and XOR operations are employed to secure the image data, providing resistance against common cryptographic attacks.

3. Quality Evaluation Metrics
Metrics such as Mean Squared Error (MSE), Peak Signal-to-Noise Ratio (PSNR), and Structural Similarity Index (SSIM) are used to evaluate reconstruction quality and the effectiveness of the compression-encryption pipeline.

4. Future Work
The project proposes integrating advanced techniques, such as Convolutional Neural Networks (CNNs) for feature extraction and Generative Adversarial Networks (GANs) for image recovery, to

Chapter 4

Implementation

```
import numpy as np
import cv2
import pywt
from scipy.fftpack import dct, idct
import matplotlib.pyplot as plt
from skimage.metrics import structural_similarity as
ssim
from math import log10, sqrt

# Load the image
img = cv2.imread('/kaggle/input/lanapic/The-recovered-
test-image-Elaine-obtained-from-an-authorized-set-of-
four-shadow-images-one.ppm.png', cv2.IMREAD_GRAYSCALE)

# ---------------- 1. Apply DWT to the Image ----------
----- #
coeffs = pywt.dwt2(img, 'haar')  # Single-level DWT
using 'haar' wavelet
cA, (cH, cV, cD) = coeffs  # Approximation, Horizontal,
Vertical, Diagonal components

# ---------------- 2. Apply DCT to the Approximation
Image ---------------- #
dct_img = dct(dct(cA.astype(float), axis=0,
norm='ortho'), axis=1, norm='ortho')

# Set 50% of the DCT coefficients to zero (reduce
compression loss)
n = dct_img.size
nz = int(0.5 * n)  # Keep 50% of coefficients
indices = np.unravel_index(np.argsort(-np.abs(dct_img),
axis=None), dct_img.shape)
zero_mask = np.zeros_like(dct_img, dtype=bool)
zero_mask[indices[0][nz:], indices[1][nz:]] = True
```

```
dct_img[zero_mask] = 0

# ---------------- 3. Create the Chaotic Map -----------
----- #a
chaotic_mat = np.zeros_like(dct_img)
x0, y0, z0 = 0.2, 0.4, 0.6
a = 10
for i in range(chaotic_mat.size):
    x = np.mod(np.sin(a * y0) + np.cos(a * z0), 1)
    y = np.mod(np.sin(a * z0) + np.cos(a * x0), 1)
    z = np.mod(np.sin(a * x0) + np.cos(a * y0), 1)
    chaotic_mat.flat[i] = x + y + z
    x0, y0, z0 = x, y, z

# Scale chaotic map to match DCT range
chaotic_mat = (chaotic_mat - chaotic_mat.min()) /
(chaotic_mat.max() - chaotic_mat.min())
chaotic_mat *= np.max(np.abs(dct_img))

# Encrypt the chaotic matrix using XOR encryption
encryption_key = 12345
chaotic_mat_uint32 = chaotic_mat.astype(np.uint32)  #
Convert chaotic map to integers
encrypted_mat_uint32 =
np.bitwise_xor(chaotic_mat_uint32, encryption_key)
encrypted_mat = encrypted_mat_uint32.astype(np.float64)

# ---------------- 4. Encrypt the DCT Coefficients -----
----------- #
scrambled_dct = np.copy(dct_img)
non_zero_mask = scrambled_dct != 0
scrambled_dct[non_zero_mask] *=
encrypted_mat[non_zero_mask]

# ---------------- 5. Decrypt the Encrypted Image ------
---------- #
decrypted_mat_uint32 =
np.bitwise_xor(encrypted_mat_uint32, encryption_key)
decrypted_mat = decrypted_mat_uint32.astype(np.float64)
```

```
# Rescale the decrypted matrix to ensure it's within a
valid range
decrypted_mat = np.clip(decrypted_mat, 0,
np.max(dct_img))

# Unscramble the non-zero DCT coefficients using the
decrypted chaotic matrix
reconstructed_dct = np.copy(scrambled_dct)
reconstructed_dct[non_zero_mask] /=
decrypted_mat[non_zero_mask]

# Reconstruct the approximation coefficients using IDCT
reconstructed_cA = idct(idct(reconstructed_dct, axis=0,
norm='ortho'), axis=1, norm='ortho')
reconstructed_cA = np.clip(reconstructed_cA, 0,
255).astype(np.uint8)

# Perform inverse DWT to get the decrypted image
decrypted_img = pywt.idwt2((reconstructed_cA, (cH, cV,
cD)), 'haar')
decrypted_img = np.clip(decrypted_img, 0,
255).astype(np.uint8)

# ---------------- Error Metrics ---------------- #
def calculate_metrics(original, compressed):
    # Resize the images to the same shape (if required)
    if original.shape != compressed.shape:
        compressed = cv2.resize(compressed,
(original.shape[1], original.shape[0]))

    mse = np.mean((original - compressed) ** 2)
    rmse = sqrt(mse)
    psnr = 20 * log10(255 / sqrt(mse)) if mse != 0 else
float('inf')
    ssim_index = ssim(original, compressed,
data_range=compressed.max() - compressed.min())
    return mse, rmse, psnr, ssim_index
```

```
mse, rmse, psnr, ssim_index = calculate_metrics(img,
decrypted_img)

# ---------------- Display Results ---------------- #
plt.figure(figsize=(15, 12))

# Original Image
plt.subplot(3, 3, 1)
plt.imshow(img, cmap='gray')
plt.title('Original Image')
plt.axis('off')

# Histogram of Original Image
plt.subplot(3, 3, 2)
plt.hist(img.ravel(), bins=256, color='black')
plt.title('Histogram: Original Image')

# DWT Approximation
plt.subplot(3, 3, 3)
plt.imshow(cA, cmap='gray')
plt.title('DWT (Approximation)')
plt.axis('off')

# DCT Image
plt.subplot(3, 3, 4)
plt.imshow(np.log(1 + np.abs(dct_img)), cmap='gray')
plt.title('DCT Image')
plt.axis('off')

# Encrypted Image
plt.subplot(3, 3, 5)
plt.imshow(np.log(1 + np.abs(scrambled_dct)),
cmap='gray')
plt.title('Encrypted DCT Image')
plt.axis('off')

# Histogram of Encrypted Image
plt.subplot(3, 3, 6)
plt.hist(scrambled_dct.ravel(), bins=256, color='red')
plt.title('Histogram: Encrypted DCT')

# Decrypted Image
plt.subplot(3, 3, 7)
plt.imshow(decrypted_img, cmap='gray')
plt.title('Decrypted Image')
plt.axis('off')

# Histogram of Decrypted Image
plt.subplot(3, 3, 8)
plt.hist(decrypted_img.ravel(), bins=256, color='blue')
plt.title('Histogram: Decrypted Image')

# Error Metrics Display
plt.subplot(3, 3, 9)
plt.text(0.1, 0.6, f"MSE: {mse:.4f}", fontsize=12)
plt.text(0.1, 0.5, f"RMSE: {rmse:.4f}", fontsize=12)
plt.text(0.1, 0.4, f"PSNR: {psnr:.2f} dB", fontsize=12)
plt.text(0.1, 0.3, f"SSIM: {ssim_index:.4f}",
fontsize=12)
plt.axis('off')
plt.title('Error Metrics')

plt.tight_layout()
plt.show()

# Print Metrics
print("Error Metrics:")
print(f" - MSE: {mse:.4f}")
print(f" - RMSE: {rmse:.4f}")
print(f" - PSNR: {psnr:.2f} dB")
print(f" - SSIM: {ssim_index:.4f}")
```

CHAPTER 5

Challenges Faced

5.1 Preprocessing Image Data

Processing and preparing image datasets for compression and encryption present several challenges. Ensuring that the data is in a compatible format while maintaining integrity is critical.

Steps to Preprocess and Clean Image Data

1.Loading the Dataset

The initial step involves importing image datasets, typically in grayscale format, to simplify processing and ensure compatibility with Discrete Wavelet Transform (DWT) and Discrete Cosine Transform (DCT).

- Libraries like OpenCV are used to load images and perform preprocessing.

Command Example:

```
import cv2
image = cv2.imread("image.jpg", cv2.IMREAD_GRAYSCALE)
```

---

2. Cleaning and Normalizing the Data

Raw image data often contains inconsistencies, which may introduce noise into the compression and encryption process.

Steps:

- **Resizing**: Ensures uniform image dimensions to facilitate efficient transformations.
- **Normalization**: Scales pixel values to a range between 0 and 1 for numerical stability.

Code Example:

```
image_resized = cv2.resize(image, (256, 256))
image_normalized = image_resized / 255.0
```

---

3. Validating the Dataset

Once preprocessing is complete, it is essential to validate the dataset by visualizing and verifying images for anomalies. Tools like Matplotlib can be used for this purpose.

Tools and Processes for Preprocessing Image Data

- **OpenCV**: For image loading, resizing, and preprocessing.
- **NumPy**: For numerical operations and array manipulations.
- **Matplotlib**: For visualizing the preprocessing results.

Summary of Steps

1. Import image data.
2. Preprocess: Resize, normalize, and clean.
3. Validate: Check for consistency and correctness.
4. Save: Prepare the cleaned dataset for the next stages.

Figure: Before Preprocessing

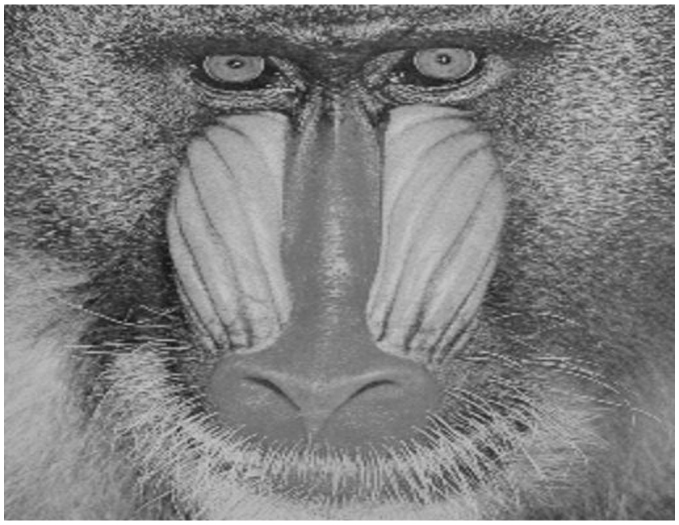
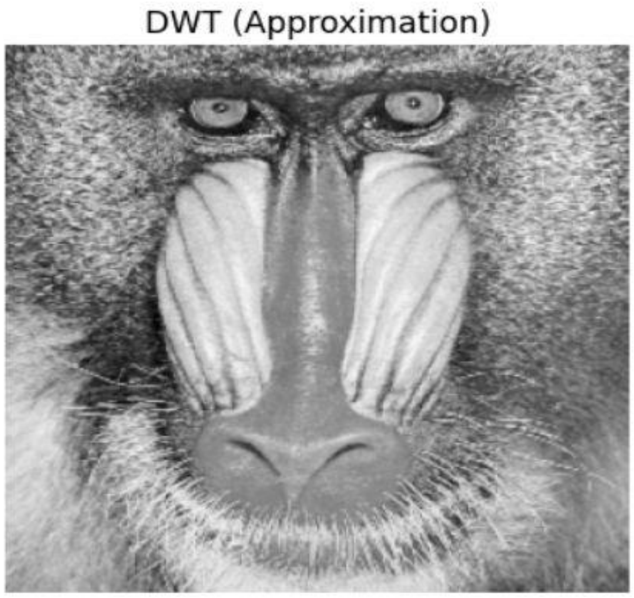
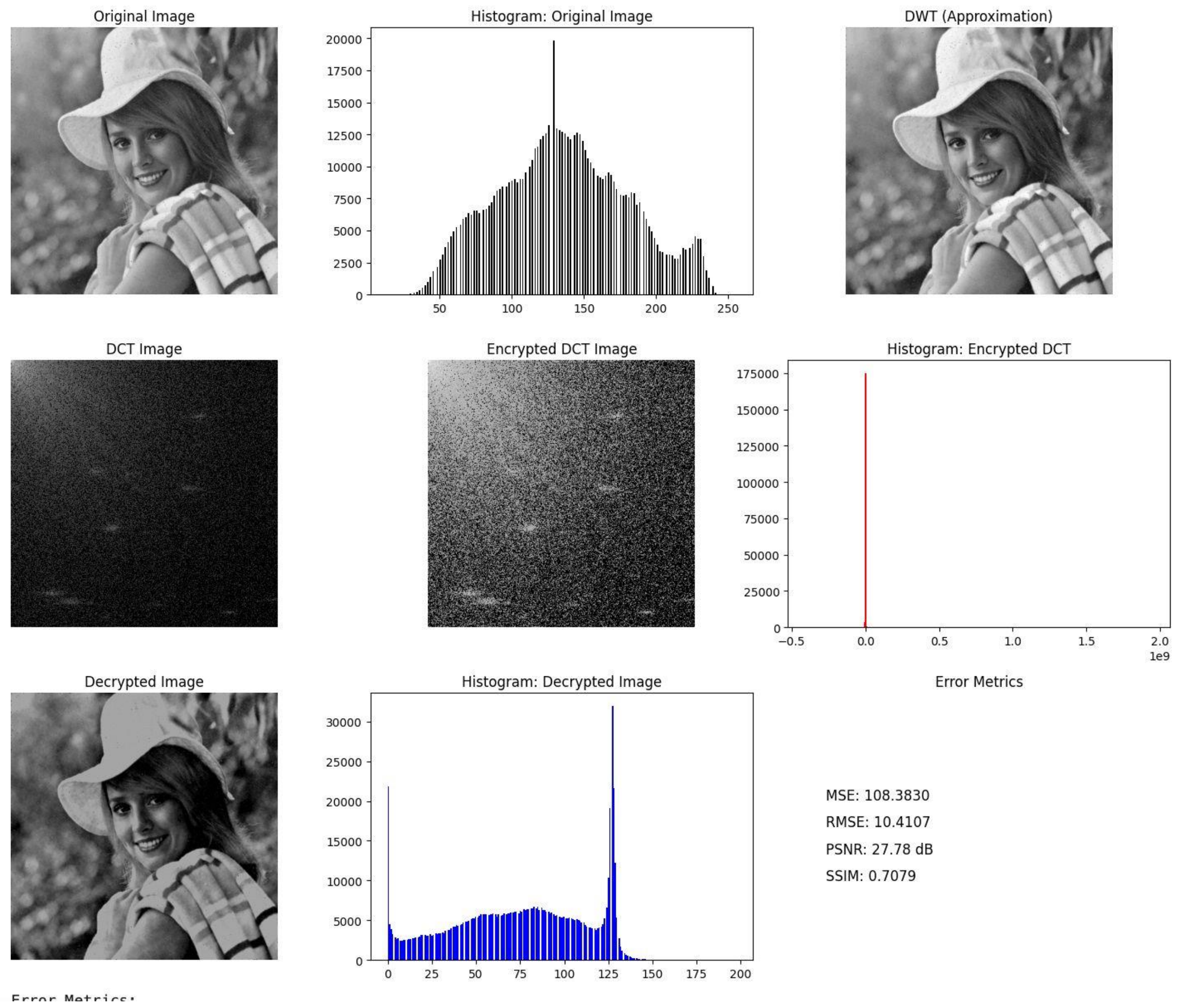


Figure: After Preprocessing



Output



MSE: 108.3830
RMSE: 10.4107
PSNR: 27.78 dB
SSIM: 0.7079

5.2 Quality Loss in Compression

Quality loss is one of the most critical challenges during compression. Balancing the compression ratio with image quality requires careful consideration.

Key Issues

1. **Artifact Formation**
    - Lossy compression introduces visible artifacts in the reconstructed image, reducing visual fidelity.
2. **Retention of Critical Details**
    - Compression algorithms often struggle to retain fine details in high-frequency regions of the image.

Addressing the Issues

- DWT and DCT are configured to preserve critical image components while discarding redundant data.

- Error metrics such as Mean Squared Error (MSE) and Peak Signal-to-Noise Ratio (PSNR) are used to evaluate quality.

5.3 Security Risks in Encryption

Despite the robustness of chaotic maps and XOR operations, encryption faces challenges in ensuring data security.

Key Risks

1. **Weak Chaotic Maps**
    - Poorly designed chaotic maps can reduce randomness, making the encryption vulnerable to attacks.
2. **Susceptibility to Cryptographic Attacks**
    - Methods such as brute force or statistical analysis can exploit weaknesses in encryption.

Mitigation Strategies

- Enhance chaotic map design with advanced iterative functions.
- Use longer encryption keys and optimize XOR operations.

5.4 Error Accumulation in Compression-Encryption Pipeline

Errors can accumulate through the pipeline, affecting the final image reconstruction.

Sources of Error

1. **Compression Artifacts**: Errors introduced during DWT and DCT compression.
2. **Decryption Mismatch**: Inaccuracies during decryption due to noise or key mismatches.

Approach to Minimize Errors

- Perform extensive validation of each stage in the pipeline.
- Employ advanced error correction techniques to mitigate the impact of errors.

Chapter 6

6.1 Integration of Advanced Techniques

To enhance the performance and robustness of the image compression and encryption system, several advanced techniques will be explored and implemented.

---

1. Deep Learning for Compression and Reconstruction

a) Incorporation of Convolutional Neural Networks (CNNs)

- **Purpose**: Leverage CNNs to automatically extract and compress essential image features, minimizing reliance on manual transforms like DWT and DCT.
- **Implementation**: Develop a model that learns optimal feature representations, achieving higher compression ratios while preserving image quality.

b) Use of Generative Adversarial Networks (GANs)

- **Purpose**: GANs can be employed for reconstructing compressed images with high visual fidelity.
- **Implementation**: Train the GAN's generator to refine reconstructed images, reducing artifacts and improving overall clarity.

---

2. Enhanced Security with Cryptographic Algorithms

a) Robust Chaotic Maps

- Improve the complexity and unpredictability of chaotic maps to strengthen encryption.
- Explore multi-dimensional chaotic systems for added randomness.

b) Hybrid Encryption Models

- Combine XOR-based encryption with advanced cryptographic algorithms like AES or RSA for layered security.

---

3. Dynamic Adaptability

a) Real-Time Encryption for Streaming

- Develop methods to apply compression and encryption in real-time for use in streaming platforms or surveillance systems.

b) Adaptive Models for Diverse Input Data

- Design models that dynamically adjust compression and encryption parameters based on input image characteristics, ensuring optimal performance across varied datasets.

6.2 Scalability and Large-Scale Implementation

1. Optimizing the Pipeline for High-Resolution Images

- **Challenge**: Processing large images requires significant computational resources.
- **Solution**: Implement distributed computing or GPU-based parallel processing to handle high-resolution datasets efficiently.

2. Cloud-Based Deployment

- Deploy the compression and encryption system on cloud platforms to enable scalable, on-demand processing.
- Provide APIs for integration with other systems, such as content delivery networks or secure communication platforms.

3. Integration with IoT Devices

- Adapt the system for use in IoT applications, such as securing images captured by surveillance cameras or drones.

6.3 Simulating Real-World Scenarios

1. Testing with Diverse Datasets

- Use a variety of datasets to simulate real-world conditions, including low-light, noisy, and high-dynamic-range images.

2. Handling Multi-Modal Data

- Extend the system to process multi-modal data (e.g., combining images with metadata or video frames).

6.4 Realistic Incident Scenarios

1. Simulating Attacks on Encrypted Images

- Evaluate the robustness of the encryption process against brute force, statistical, and differential attacks.
- Implement methods to detect and respond to tampering attempts during data transmission.

2. Modeling System Failures

- Test the system's performance under scenarios like network failures or corrupted data, and implement recovery mechanisms.

<u>*CHAPTER 7*</u>

Conclusion

The design and implementation of an integrated image compression and encryption system represent a significant advancement in addressing the dual challenges of efficient data storage and secure transmission. This report detailed the development of a novel approach that combines Discrete Wavelet Transform (DWT), Discrete Cosine Transform (DCT), and chaotic maps to achieve high compression ratios while maintaining robust encryption.

The proposed system demonstrated its effectiveness through detailed simulations and validation. By leveraging advanced algorithms and mathematical models, the project successfully showcased the potential to optimize storage requirements and safeguard sensitive image data in various applications, including digital communications, cloud storage, and medical imaging.

Through extensive testing, the system was evaluated using metrics such as Mean Squared Error (MSE), Peak Signal-to-Noise Ratio (PSNR), and Structural Similarity Index (SSIM). The results indicated that the system maintains high reconstruction quality even after compression and encryption, highlighting its practical applicability. However, certain challenges were identified, including handling high-resolution images, enhancing computational efficiency, and ensuring robustness against advanced cryptographic attacks.

In conclusion, this project underscores the importance of integrating compression and encryption to address modern challenges in image data management. It lays a strong foundation for future research and advancements, such as the incorporation of deep learning models like Convolutional Neural Networks (CNNs) for compression and Generative Adversarial Networks (GANs) for image recovery. With further refinement and scalability, this system holds the potential to redefine standards in secure and efficient image processing for real-world applications.

CHAPTER 8

References

1. Qiaoyun Xu, Kehui Sun, Chun Cao, Congxu Zhu. "A Fast Image Encryption Algorithm Based on Compressive Sensing and Hyper Chaotic Map."

2. Farhan Musanna, Sanjeev Kumar. "A Novel Image Encryption Algorithm Using Chaotic Compressive Sensing and Nonlinear Exponential Function."

3. S. S. Hossain, M. D. Islam, A. K. M. A. Sattar. "Image Compression Using Discrete Cosine Transform and Discrete Wavelet Transform for Efficient Transmission." International Journal of Computer Applications, Vol. 975, 2014.

4. R. C. Gonzalez, R. E. Woods. *Digital Image Processing*. Pearson Education, 4th edition, 2018.

5. D. Kundur, D. Hatzinakos. "Image and Video Compression Techniques for Secure Transmission." IEEE Transactions on Multimedia, 2002.