📌 1. Introduction

- This notebook presents my solutions for the Computer Vision Technical Assessment. Each task is implemented with a clear methodology and documented code. Images are enhanced, text is extracted using OCR techniques, super-resolution is applied, and object extraction is demonstrated.

📂 2. Environment Setup

```
!pip install pytesseract
!pip install scikit-image
!pip install easyocr

# !pip install python-doctr[torch]
# !pip install -U ultralytics
```

```
Requirement already satisfied: packaging>=21 in /usr/local/lib/python3.11/dist-packages (from scikit-image->easyocr) (24.2)
Requirement already satisfied: lazy-loader>=0.4 in /usr/local/lib/python3.11/dist-packages (from scikit-image->easyocr) (0.4)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.11/dist-packages (from jinja2->torch->easyocr) (3.0.2)
Downloading easyocr-1.7.2-py3-none-any.whl (2.9 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.9/2.9 MB 24.9 MB/s eta 0:00:00
Downloading nvidia_cublas_cu12-12.4.5.8-py3-none-manylinux2014_x86_64.whl (363.4 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 363.4/363.4 MB 4.2 MB/s eta 0:00:00
Downloading nvidia_cuda_cupti_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl (13.8 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 13.8/13.8 MB 95.4 MB/s eta 0:00:00
Downloading nvidia_cuda_nvrtc_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl (24.6 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 24.6/24.6 MB 70.7 MB/s eta 0:00:00
Downloading nvidia_cuda_runtime_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl (883 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 883.7/883.7 kB 46.7 MB/s eta 0:00:00
Downloading nvidia_cudnn_cu12-9.1.0.70-py3-none-manylinux2014_x86_64.whl (664.8 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 664.8/664.8 MB 856.8 kB/s eta 0:00:00
Downloading nvidia_cufft_cu12-11.2.1.3-py3-none-manylinux2014_x86_64.whl (211.5 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 211.5/211.5 MB 4.9 MB/s eta 0:00:00
Downloading nvidia_curand_cu12-10.3.5.147-py3-none-manylinux2014_x86_64.whl (56.3 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 56.3/56.3 MB 15.2 MB/s eta 0:00:00
Downloading nvidia_cusolver_cu12-11.6.1.9-py3-none-manylinux2014_x86_64.whl (127.9 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 127.9/127.9 MB 7.1 MB/s eta 0:00:00
Downloading nvidia_cusparse_cu12-12.3.1.170-py3-none-manylinux2014_x86_64.whl (207.5 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 207.5/207.5 MB 5.7 MB/s eta 0:00:00
Downloading nvidia_nvjitlink_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl (21.1 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 21.1/21.1 MB 78.3 MB/s eta 0:00:00
Downloading ninja-1.11.1.4-py3-none-manylinux_2_12_x86_64.manylinux2010_x86_64.whl (422 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 422.8/422.8 kB 26.9 MB/s eta 0:00:00
Downloading pyclipper-1.3.0.post6-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (969 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 969.6/969.6 kB 43.9 MB/s eta 0:00:00
Downloading python_bidi-0.6.6-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (292 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 292.9/292.9 kB 23.5 MB/s eta 0:00:00
Installing collected packages: python-bidi, pyclipper, nvidia-nvjitlink-cu12, nvidia-curand-cu12, nvidia-cufft-cu12, nvidia-cuda-
  Attempting uninstall: nvidia-nvjitlink-cu12
    Found existing installation: nvidia-nvjitlink-cu12 12.5.82
    Uninstalling nvidia-nvjitlink-cu12-12.5.82:
      Successfully uninstalled nvidia-nvjitlink-cu12-12.5.82
  Attempting uninstall: nvidia-curand-cu12
    Found existing installation: nvidia-curand-cu12 10.3.6.82
    Uninstalling nvidia-curand-cu12-10.3.6.82:
      Successfully uninstalled nvidia-curand-cu12-10.3.6.82
  Attempting uninstall: nvidia-cufft-cu12
    Found existing installation: nvidia-cufft-cu12 11.2.3.61
    Uninstalling nvidia-cufft-cu12-11.2.3.61:
      Successfully uninstalled nvidia-cufft-cu12-11.2.3.61
  Attempting uninstall: nvidia-cuda-runtime-cu12
    Found existing installation: nvidia-cuda-runtime-cu12 12.5.82
    Uninstalling nvidia-cuda-runtime-cu12-12.5.82:
      Successfully uninstalled nvidia-cuda-runtime-cu12-12.5.82
  Attempting uninstall: nvidia-cuda-nvrtc-cu12
    Found existing installation: nvidia-cuda-nvrtc-cu12 12.5.82
    Uninstalling nvidia-cuda-nvrtc-cu12-12.5.82:
      Successfully uninstalled nvidia-cuda-nvrtc-cu12-12.5.82
  Attempting uninstall: nvidia-cuda-cupti-cu12
    Found existing installation: nvidia-cuda-cupti-cu12 12.5.82
    Uninstalling nvidia-cuda-cupti-cu12-12.5.82:
      Successfully uninstalled nvidia-cuda-cupti-cu12-12.5.82
  Attempting uninstall: nvidia-cublas-cu12
```

```
import cv2
import os
import numpy as np
import matplotlib.pyplot as plt
```

```
from PIL import Image
import pytesseract
import easyocr
# Create the reader object (English only)
reader = easyocr.Reader(['en'])



from easyocr import Reader
from skimage.metrics import structural_similarity as ssim
```

```
WARNING:easyocr.easyocr:Neither CUDA nor MPS are available - defaulting to CPU. Note: This module is much faster with a GPU.
```

3.1 Task 1: Image Enhancement

Objective:

Enhance the provided images using multiple image processing techniques.

```
def adaptive_gamma_correction(img, is_img1=True):
    """
    Apply adaptive gamma correction with parameters tuned for each image.
    """
    img_normalized = img / 255.0
    mean_intensity = np.mean(img_normalized)
    # Different gamma tuning for each image
    if is_img1:
        gamma = 3.0 / (1.0 + mean_intensity) if mean_intensity > 0 else 1.0  # Stronger for img1 (very dark)
    else:
        gamma = 1.5 / (1.0 + mean_intensity) if mean_intensity > 0 else 1.0  # Milder for img2 (better lighting)
    img_gamma = np.power(img_normalized, gamma)
    img_gamma = np.uint8(img_gamma * 255)
    return img_gamma



def enhance_methods(img, is_img1=True):
    """
    Apply enhancement methods with parameters tuned for each image.
    """
    # Step 1: Denoising using Non-Local Means
    if is_img1:
        # Stronger denoising for img1 due to high noise
        denoised = cv2.fastNlMeansDenoisingColored(img, None, h=15, hColor=15, templateWindowSize=7, searchWindowSize=21)
    else:
        # Lighter denoising for img2 as noise is less
        denoised = cv2.fastNlMeansDenoisingColored(img, None, h=5, hColor=5, templateWindowSize=7, searchWindowSize=21)

    # Convert to HSV for Histogram Equalization and CLAHE
    img_hsv = cv2.cvtColor(denoised, cv2.COLOR_BGR2HSV)
    h, s, v = cv2.split(img_hsv)

    # 1. Histogram Equalization (on Value channel)
    hist_eq_v = cv2.equalizeHist(v)
    hist_eq_hsv = cv2.merge([h, s, hist_eq_v])
    hist_eq = cv2.cvtColor(hist_eq_hsv, cv2.COLOR_HSV2BGR)

    # 2. CLAHE (on Value channel)
    if is_img1:
        # Higher clipLimit for img1 to boost contrast
        clahe = cv2.createCLAHE(clipLimit=5.0, tileGridSize=(8, 8))
    else:
        # Moderate clipLimit for img2 to avoid over-enhancement
        clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8, 8))
    clahe_v = clahe.apply(v)
    clahe_hsv = cv2.merge([h, s, clahe_v])
    clahe_eq = cv2.cvtColor(clahe_hsv, cv2.COLOR_HSV2BGR)

    # 3. Bilateral Filter + Sharpening
    if is_img1:
        # Stronger bilateral for img1 to reduce noise further
        bilateral = cv2.bilateralFilter(denoised, d=9, sigmaColor=75, sigmaSpace=75)
    else:
        # Lighter bilateral for img2 to preserve details
        bilateral = cv2.bilateralFilter(denoised, d=9, sigmaColor=30, sigmaSpace=30)
    kernel_sharp = np.array([[0, -1, 0],
                             [-1, 5, -1],
```

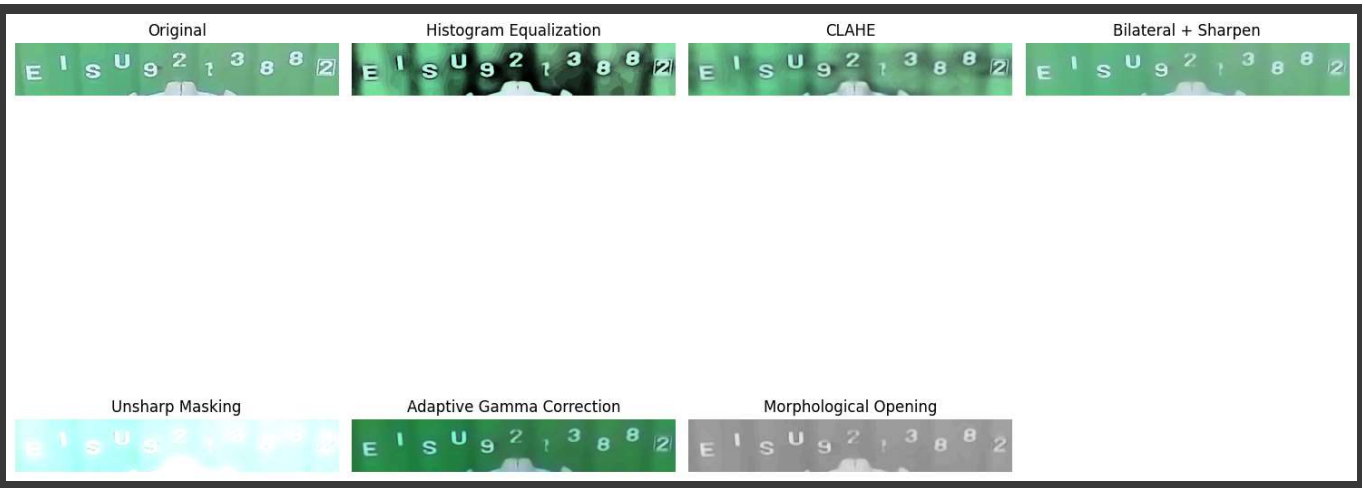
```
                            [0, -1, 0]])
    bilateral_sharp = cv2.filter2D(bilateral, -1, kernel_sharp)

    # 4. Unsharp Masking
    if is_img1:
        # Stronger unsharp for img1 to enhance text edges
        gaussian = cv2.GaussianBlur(denoised, (9, 9), 10.0)
        unsharp = cv2.addWeighted(denoised, 2.5, gaussian, -0.5, 0)
    else:
        # Moderate unsharp for img2
        gaussian = cv2.GaussianBlur(denoised, (5, 5), 5.0)
        unsharp = cv2.addWeighted(denoised, 1.5, gaussian, -0.5, 0)

    # 5. Adaptive Gamma Correction
    gamma_corrected = adaptive_gamma_correction(denoised, is_img1)

    # 6. Morphological Operation (Opening to enhance text)
    gray = cv2.cvtColor(denoised, cv2.COLOR_BGR2GRAY)
    if is_img1:
        # Larger kernel for img1 to clean more noise
        kernel = np.ones((4, 4), np.uint8)
    else:
        # Smaller kernel for img2 to preserve details
        kernel = np.ones((5, 5), np.uint8)
    morph = cv2.morphologyEx(gray, cv2.MORPH_OPEN, kernel)
    morph_color = cv2.cvtColor(morph, cv2.COLOR_GRAY2BGR)

    return hist_eq, clahe_eq, bilateral_sharp, unsharp, gamma_corrected, morph_color
```

```
def show_images(images, titles):
    """
    Display images with their titles using matplotlib.
    """
    plt.figure(figsize=(15, 10))
    for i, (img, title) in enumerate(zip(images, titles)):
        img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        plt.subplot(2, 4, i + 1)
        plt.imshow(img_rgb)
        plt.title(title)
        plt.axis('off')
    plt.tight_layout()
    plt.show()

# Process both images with different parameters
def process_image(image_path, is_img1=True):
    img = cv2.imread(image_path)
    if img is None:
        print(f"Error: Could not load image {image_path}. Check the file path.")
        return
    hist_eq, clahe_eq, bilateral_sharp, unsharp, gamma_corrected, morph = enhance_methods(img, is_img1)
    show_images(
        [img, hist_eq, clahe_eq, bilateral_sharp, unsharp, gamma_corrected, morph],
        ["Original", "Histogram Equalization", "CLAHE", "Bilateral + Sharpen", "Unsharp Masking", "Adaptive Gamma Correction", "Morpholo
    )
```

First Image (task1_img1- E1SU92213882)

- **Issues**: The text is relatively clear, but the contrast between the text and the background can be improved, and there may be slight noise or minor blur.
- **Priorities**:
  - Moderately improve contrast (CLAHE).
  - Increase text sharpness (Sharpening or Unsharp Masking).
  - Reduce slight noise if present (Denoising).

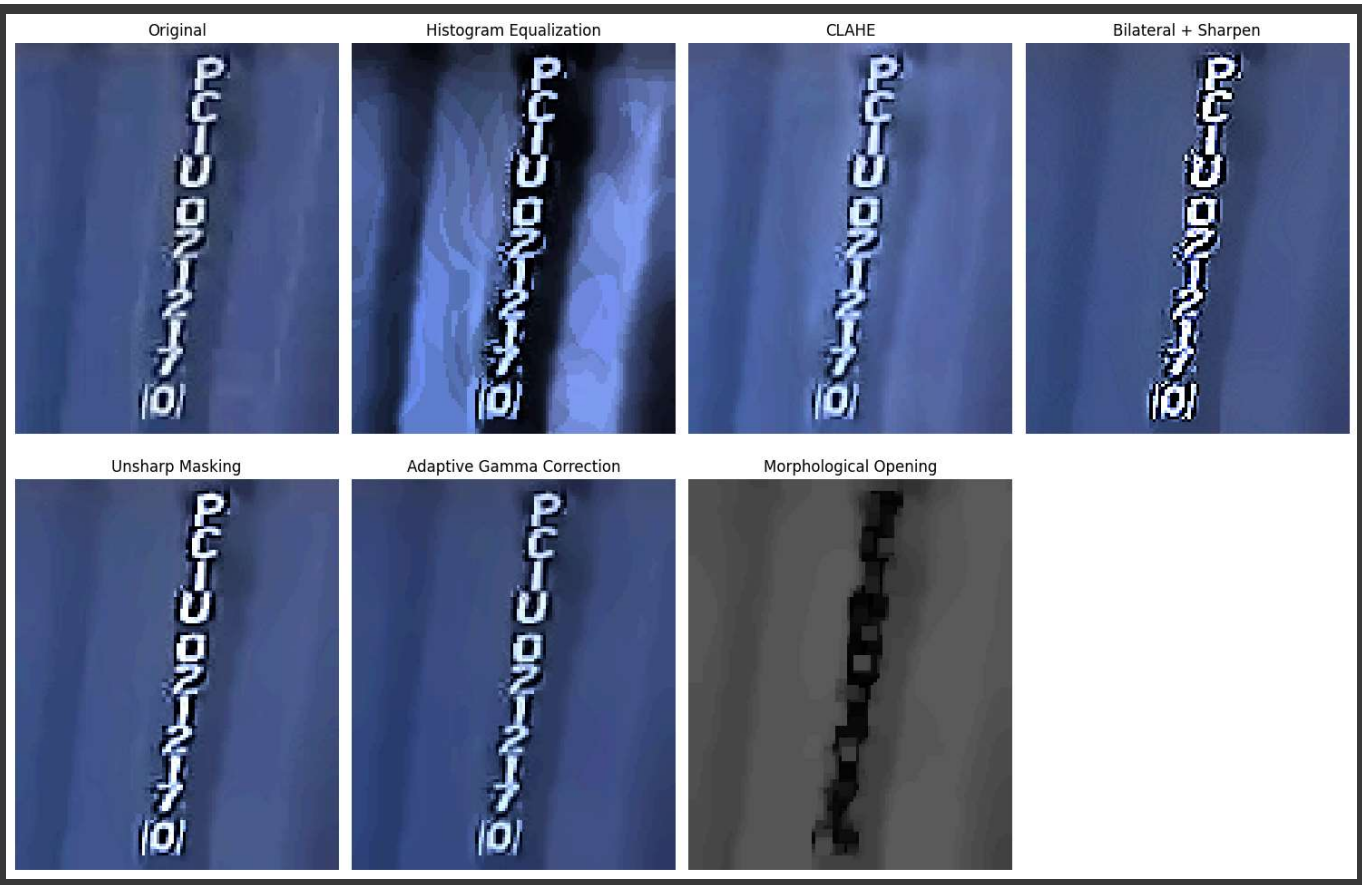
```
# Process both images
task1_img1_process=process_image("task1_img1.jpg", is_img1=True)  # For E1SU92213882 image
task1_img1_process
```



Second Image (task1_img2- PCIU0212170)

- **Issues**: High noise, very low contrast, text is unclear.
- **Priorities**:
  - Significantly reduce noise (Denoising).
  - Strongly increase contrast (CLAHE or Histogram Equalization).
  - Enhance edges to highlight the text (Unsharp Masking or Morphological Operations).

```
task1_img2_process=process_image("task1_img2.jpg", is_img1=False)  # For PCIU0212170 image
task1_img2_process
```




```
def adaptive_gamma_correction(img, is_img1=True):
    """
    Apply adaptive gamma correction with parameters tuned for each image.
    """
    img_normalized = img / 255.0
    mean_intensity = np.mean(img_normalized)
    # Different gamma tuning for each image
    if is_img1:
        gamma = 3.0 / (1.0 + mean_intensity) if mean_intensity > 0 else 1.0  # Stronger for img1 (very dark)
    else:
        gamma = 1.5 / (1.0 + mean_intensity) if mean_intensity > 0 else 1.0  # Milder for img2 (better lighting)
    img_gamma = np.power(img_normalized, gamma)
    img_gamma = np.uint8(img_gamma * 255)
    return img_gamma

def compute_ssim(original, enhanced):
    """
    Compute the SSIM between the original and enhanced images.

    Parameters:
    - original: The original image (grayscale or color).
    - enhanced: The enhanced image (grayscale or color).

    Returns:
    - SSIM score.
    """
    # Convert images to grayscale for SSIM computation
    if len(original.shape) == 3:
        original_gray = cv2.cvtColor(original, cv2.COLOR_BGR2GRAY)
    else:
        original_gray = original
    if len(enhanced.shape) == 3:
        enhanced_gray = cv2.cvtColor(enhanced, cv2.COLOR_BGR2GRAY)
    else:
        enhanced_gray = enhanced

    # Compute SSIM (multichannel=False since we're using grayscale)
```

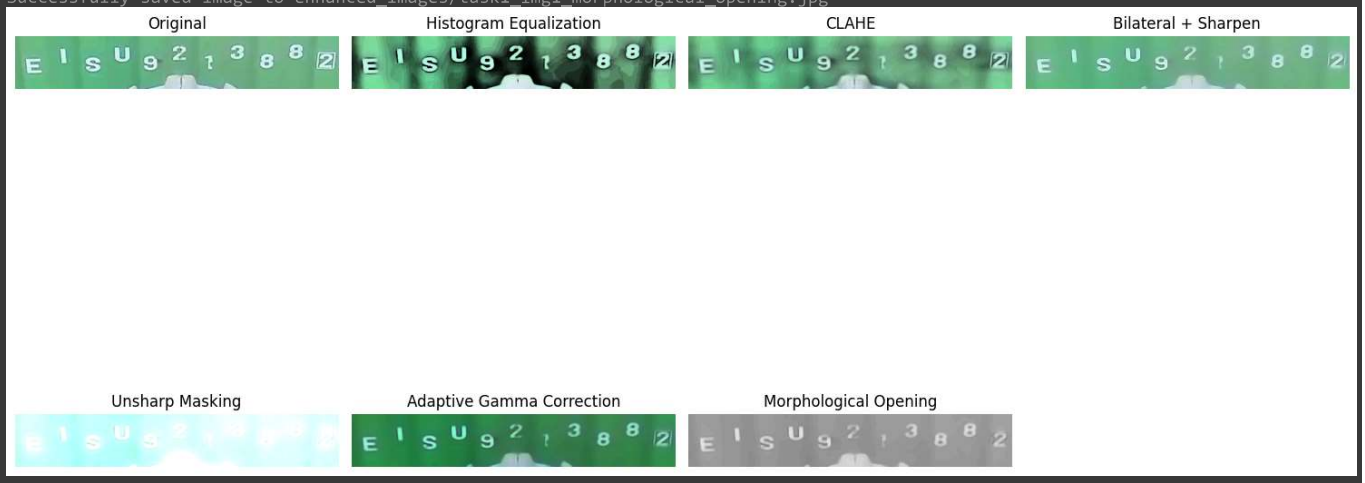
```
    score, _ = ssim(original_gray, enhanced_gray, full=True)
    return score


def enhance_methods(img, is_img1=True):
    """
    Apply enhancement methods with parameters tuned for each image.
    """
    # Step 1: Denoising using Non-Local Means
    if is_img1:
        # Stronger denoising for img1 due to high noise
        denoised = cv2.fastNlMeansDenoisingColored(img, None, h=15, hColor=15, templateWindowSize=7, searchWindowSize=21)
    else:
        # Lighter denoising for img2 as noise is less
        denoised = cv2.fastNlMeansDenoisingColored(img, None, h=5, hColor=5, templateWindowSize=7, searchWindowSize=21)

    # Convert to HSV for Histogram Equalization and CLAHE
    img_hsv = cv2.cvtColor(denoised, cv2.COLOR_BGR2HSV)
    h, s, v = cv2.split(img_hsv)

    # 1. Histogram Equalization (on Value channel)
    hist_eq_v = cv2.equalizeHist(v)
    hist_eq_hsv = cv2.merge([h, s, hist_eq_v])
    hist_eq = cv2.cvtColor(hist_eq_hsv, cv2.COLOR_HSV2BGR)

    # 2. CLAHE (on Value channel)
    if is_img1:
        # Higher clipLimit for img1 to boost contrast
        clahe = cv2.createCLAHE(clipLimit=5.0, tileGridSize=(8, 8))
    else:
        # Moderate clipLimit for img2 to avoid over-enhancement
        clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8, 8))
    clahe_v = clahe.apply(v)
    clahe_hsv = cv2.merge([h, s, clahe_v])
    clahe_eq = cv2.cvtColor(clahe_hsv, cv2.COLOR_HSV2BGR)

    # 3. Bilateral Filter + Sharpening
    if is_img1:
        # Stronger bilateral for img1 to reduce noise further
        bilateral = cv2.bilateralFilter(denoised, d=9, sigmaColor=75, sigmaSpace=75)
    else:
        # Lighter bilateral for img2 to preserve details
        bilateral = cv2.bilateralFilter(denoised, d=9, sigmaColor=30, sigmaSpace=30)
    kernel_sharp = np.array([[0, -1, 0],
                             [-1, 5, -1],
                             [0, -1, 0]])
    bilateral_sharp = cv2.filter2D(bilateral, -1, kernel_sharp)

    # 4. Unsharp Masking
    if is_img1:
        # Stronger unsharp for img1 to enhance text edges
        gaussian = cv2.GaussianBlur(denoised, (9, 9), 10.0)
        unsharp = cv2.addWeighted(denoised, 2.5, gaussian, -0.5, 0)
    else:
        # Moderate unsharp for img2
        gaussian = cv2.GaussianBlur(denoised, (5, 5), 5.0)
        unsharp = cv2.addWeighted(denoised, 1.5, gaussian, -0.5, 0)

    # 5. Adaptive Gamma Correction
    gamma_corrected = adaptive_gamma_correction(denoised, is_img1)

    # 6. Morphological Operation (Opening to enhance text)
    gray = cv2.cvtColor(denoised, cv2.COLOR_BGR2GRAY)
    if is_img1:
        # Larger kernel for img1 to clean more noise
        kernel = np.ones((4, 4), np.uint8)
    else:
        # Smaller kernel for img2 to preserve details
        kernel = np.ones((3, 3), np.uint8)
    morph = cv2.morphologyEx(gray, cv2.MORPH_OPEN, kernel)
    morph_color = cv2.cvtColor(morph, cv2.COLOR_GRAY2BGR)

    return hist_eq, clahe_eq, bilateral_sharp, unsharp, gamma_corrected, morph_color
```

```
def show_images(images, titles):
    """
    Display images with their titles using matplotlib.
    """
    plt.figure(figsize=(15, 10))
    for i, (img, title) in enumerate(zip(images, titles)):
```

```
        img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        plt.subplot(2, 4, i + 1)
        plt.imshow(img_rgb)
        plt.title(title)
        plt.axis('off')
    plt.tight_layout()
    plt.show()

def save_image(image, save_path):
    """
    Save an image using cv2.imwrite with error handling.
    """
    try:
        success = cv2.imwrite(save_path, image)
        if success:
            print(f"Successfully saved image to {save_path}")
        else:
            print(f"Failed to save image to {save_path}")
        return success
    except Exception as e:
        print(f"Error saving image to {save_path}: {str(e)}")
        return False

def process_image(image_path, is_img1=True, save_dir="enhanced_images"):
    """
    Process the image, compute SSIM for each enhancement, display results, and save the images.
    """
    # Load the image
    img = cv2.imread(image_path)
    if img is None:
        print(f"Error: Could not load image {image_path}. Check the file path.")
        return

    # Create the save directory if it doesn't exist
    if not os.path.exists(save_dir):
        os.makedirs(save_dir)

    # Get the base filename
    base_filename = os.path.splitext(os.path.basename(image_path))[0]

    # Save the original image
    original_save_path = os.path.join(save_dir, f"{base_filename}_original.jpg")
    save_image(img, original_save_path)

    # Apply enhancement methods
    hist_eq, clahe_eq, bilateral_sharp, unsharp, gamma_corrected, morph = enhance_methods(img, is_img1)

    # List of enhanced images and their titles
    enhanced_images = [img, hist_eq, clahe_eq, bilateral_sharp, unsharp, gamma_corrected, morph]
    titles = ["Original", "Histogram Equalization", "CLAHE", "Bilateral + Sharpen", "Unsharp Masking", "Adaptive Gamma Correction", "Mor

    # Compute SSIM for each enhanced image compared to the original
    print(f"\nSSIM Scores for {base_filename}:")
    ssim_scores = {}
    for enhanced_img, title in zip(enhanced_images[1:], titles[1:]):  # Skip the original image
        ssim_score = compute_ssim(img, enhanced_img)
        ssim_scores[title] = ssim_score
        print(f"{title}: SSIM = {ssim_score:.4f}")

    # Find the best SSIM score
    best_method = max(ssim_scores, key=ssim_scores.get)
    print(f"\nBest SSIM Score for {base_filename}: {best_method} with SSIM = {ssim_scores[best_method]:.4f}\n")

    # Save each enhanced image
    for enhanced_img, title in zip(enhanced_images[1:], titles[1:]):
        filename_suffix = title.lower().replace(" + ", "_").replace(" ", "_")
        save_path = os.path.join(save_dir, f"{base_filename}_{filename_suffix}.jpg")
        save_image(enhanced_img, save_path)

    # Display the images
    show_images(enhanced_images, titles)
```

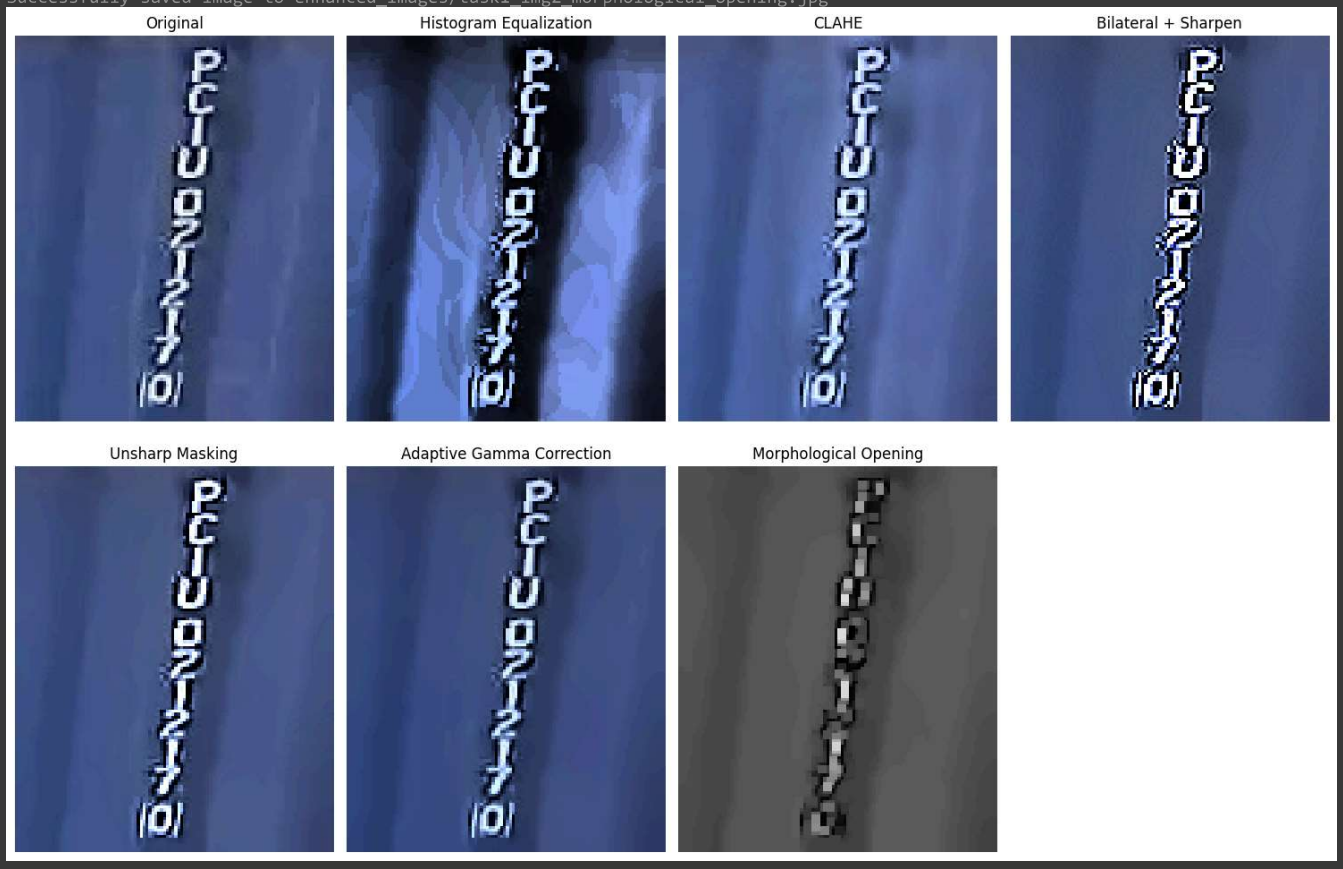
```
# Process task1_img1 (PCU)
process_image("task1_img1.jpg", is_img1=True, save_dir="enhanced_images")
```

```
Successfully saved image to enhanced_images/task1_img1_original.jpg

SSIM Scores for task1_img1:
Histogram Equalization: SSIM = 0.5146
CLAHE: SSIM = 0.7925
Bilateral + Sharpen: SSIM = 0.8667
Unsharp Masking: SSIM = 0.7005
Adaptive Gamma Correction: SSIM = 0.8369
Morphological Opening: SSIM = 0.8295

Best SSIM Score for task1_img1: Bilateral + Sharpen with SSIM = 0.8667

Successfully saved image to enhanced_images/task1_img1_histogram_equalization.jpg
Successfully saved image to enhanced_images/task1_img1_clahe.jpg
Successfully saved image to enhanced_images/task1_img1_bilateral_sharpen.jpg
Successfully saved image to enhanced_images/task1_img1_unsharp_masking.jpg
Successfully saved image to enhanced_images/task1_img1_adaptive_gamma_correction.jpg
Successfully saved image to enhanced_images/task1_img1_morphological_opening.jpg
```




```
# Process task1_img2 (E1SU92213882)
process_image("task1_img2.jpg", is_img1=False, save_dir="enhanced_images")
```

```
Successfully saved image to enhanced_images/task1_img2_original.jpg

SSIM Scores for task1_img2:
Histogram Equalization: SSIM = 0.5693
CLAHE: SSIM = 0.9141
Bilateral + Sharpen: SSIM = 0.9076
Unsharp Masking: SSIM = 0.9591
Adaptive Gamma Correction: SSIM = 0.9601
Morphological Opening: SSIM = 0.8938

Best SSIM Score for task1_img2: Adaptive Gamma Correction with SSIM = 0.9601

Successfully saved image to enhanced_images/task1_img2_histogram_equalization.jpg
Successfully saved image to enhanced_images/task1_img2_clahe.jpg
Successfully saved image to enhanced_images/task1_img2_bilateral_sharpen.jpg
Successfully saved image to enhanced_images/task1_img2_unsharp_masking.jpg
Successfully saved image to enhanced_images/task1_img2_adaptive_gamma_correction.jpg
Successfully saved image to enhanced_images/task1_img2_morphological_opening.jpg
```



Combine height SSIM two Image

```
def adaptive_gamma_correction(img, is_img1=True):
    """
    Apply adaptive gamma correction with parameters tuned for each image.
    """
    img_normalized = img / 255.0
    mean_intensity = np.mean(img_normalized)
    if is_img1:
        gamma = 3.0 / (1.0 + mean_intensity) if mean_intensity > 0 else 1.0
    else:
        gamma = 1.5 / (1.0 + mean_intensity) if mean_intensity > 0 else 1.0
    img_gamma = np.power(img_normalized, gamma)
    img_gamma = np.uint8(img_gamma * 255)
    return img_gamma

def enhance_saturation(img_hsv, factor=1.5):
    """
```

```
    Enhance the saturation of the image in HSV space.
    """
    h, s, v = cv2.split(img_hsv)
    s = np.clip(s * factor, 0, 255).astype(np.uint8)
    return cv2.merge([h, s, v])

def compute_ssim(original, enhanced):
    """
    Compute the SSIM between the original and enhanced images.
    """
    if len(original.shape) == 3:
        original_gray = cv2.cvtColor(original, cv2.COLOR_BGR2GRAY)
    else:
        original_gray = original
    if len(enhanced.shape) == 3:
        enhanced_gray = cv2.cvtColor(enhanced, cv2.COLOR_BGR2GRAY)
    else:
        enhanced_gray = enhanced
    score, _ = ssim(original_gray, enhanced_gray, full=True)
    return score

def enhance_methods_combined_img2(img):
    """
    Apply a combined enhancement pipeline optimized for task1_img1 (PCIU).
    """
    # Step 1: Denoising
    denoised = cv2.fastNlMeansDenoisingColored(img, None, h=5, hColor=5, templateWindowSize=7, searchWindowSize=21)

    # Step 2: Adaptive Gamma Correction
    gamma_corrected = adaptive_gamma_correction(denoised, is_img1=False)

    # Step 3: Unsharp Masking
    gaussian = cv2.GaussianBlur(gamma_corrected, (5, 5), 5.0)
    unsharp = cv2.addWeighted(gamma_corrected, 1.5, gaussian, -0.5, 0)
    unsharp = np.clip(unsharp, 0, 255).astype(np.uint8)

    # Step 4: Convert to HSV for CLAHE and saturation enhancement
    img_hsv = cv2.cvtColor(unsharp, cv2.COLOR_BGR2HSV)
    h, s, v = cv2.split(img_hsv)

    # Step 5: CLAHE
    clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8, 8))
    clahe_v = clahe.apply(v)

    # Step 6: Recombine HSV
    img_hsv_enhanced = cv2.merge([h, s, clahe_v])

    # Step 7: Enhance saturation
    img_hsv_enhanced = enhance_saturation(img_hsv_enhanced, factor=1.5)

    # Step 8: Convert back to BGR
    final = cv2.cvtColor(img_hsv_enhanced, cv2.COLOR_HSV2BGR)

    return final

def enhance_methods_combined_img1(img):
    # Step 1: Bilateral Filter (Denoising)
    bilateral = cv2.bilateralFilter(img, d=9, sigmaColor=75, sigmaSpace=75)

    # Step 2: Morphological Opening to reduce small noise and enhance text
    gray = cv2.cvtColor(bilateral, cv2.COLOR_BGR2GRAY)
    kernel = np.ones((4, 4), np.uint8)  # Using the same kernel size as in the original code
    morph = cv2.morphologyEx(gray, cv2.MORPH_OPEN, kernel)

    # Step 3: Adaptive Gamma Correction
    morph_color = cv2.cvtColor(morph, cv2.COLOR_GRAY2BGR)
    gamma_corrected = adaptive_gamma_correction(morph_color, is_img1=True)

    # Step 4: Sharpening
    kernel_sharp = np.array([[0, -1, 0],
                             [-1, 5, -1],
                             [0, -1, 0]])
    sharpened = cv2.filter2D(gamma_corrected, -1, kernel_sharp)
    sharpened = np.clip(sharpened, 0, 255).astype(np.uint8)

    return sharpened

def show_images(images, titles):
    """
    Display images with their titles using matplotlib.
    """
    plt.figure(figsize=(15, 5))
```

```
    for i, (img, title) in enumerate(zip(images, titles)):
        img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        plt.subplot(1, len(images), i + 1)
        plt.imshow(img_rgb)
        plt.title(title)
        plt.axis('off')
    plt.tight_layout()
    plt.show()

def save_image(image, save_path):
    """
    Save an image using cv2.imwrite with error handling.
    """
    try:
        success = cv2.imwrite(save_path, image)
        if success:
            print(f"Successfully saved image to {save_path}")
        else:
            print(f"Failed to save image to {save_path}")
        return success
    except Exception as e:
        print(f"Error saving image to {save_path}: {str(e)}")
        return False
```

Real-ESRGAN + OCR + SSIM

```
!pip install realesrgan
```

```
Requirement already satisfied: jinja2 in /usr/local/lib/python3.11/dist-packages (from torch>=1.7->realesrgan) (3.1.6)
Requirement already satisfied: fsspec in /usr/local/lib/python3.11/dist-packages (from torch>=1.7->realesrgan) (2025.3.2)
Requirement already satisfied: nvidia-cuda-nvrtc-cu12==12.4.127 in /usr/local/lib/python3.11/dist-packages (from torch>=1.7->reale
Requirement already satisfied: nvidia-cuda-runtime-cu12==12.4.127 in /usr/local/lib/python3.11/dist-packages (from torch>=1.7->rea
Requirement already satisfied: nvidia-cuda-cupti-cu12==12.4.127 in /usr/local/lib/python3.11/dist-packages (from torch>=1.7->reale
Requirement already satisfied: nvidia-cudnn-cu12==9.1.0.70 in /usr/local/lib/python3.11/dist-packages (from torch>=1.7->realesrgan
Requirement already satisfied: nvidia-cublas-cu12==12.4.5.8 in /usr/local/lib/python3.11/dist-packages (from torch>=1.7->realesrga
Requirement already satisfied: nvidia-cufft-cu12==11.2.1.3 in /usr/local/lib/python3.11/dist-packages (from torch>=1.7->realesrgan
Requirement already satisfied: nvidia-curand-cu12==10.3.5.147 in /usr/local/lib/python3.11/dist-packages (from torch>=1.7->realesr
Requirement already satisfied: nvidia-cusolver-cu12==11.6.1.9 in /usr/local/lib/python3.11/dist-packages (from torch>=1.7->realesr
Requirement already satisfied: nvidia-cusparse-cu12==12.3.1.170 in /usr/local/lib/python3.11/dist-packages (from torch>=1.7->reale
Requirement already satisfied: nvidia-cusparselt-cu12==0.6.2 in /usr/local/lib/python3.11/dist-packages (from torch>=1.7->realesrg
Requirement already satisfied: nvidia-nccl-cu12==2.21.5 in /usr/local/lib/python3.11/dist-packages (from torch>=1.7->realesrgan) (
Requirement already satisfied: nvidia-nvtx-cu12==12.4.127 in /usr/local/lib/python3.11/dist-packages (from torch>=1.7->realesrgan)
Requirement already satisfied: nvidia-nvjitlink-cu12==12.4.127 in /usr/local/lib/python3.11/dist-packages (from torch>=1.7->reales
Requirement already satisfied: triton==3.2.0 in /usr/local/lib/python3.11/dist-packages (from torch>=1.7->realesrgan) (3.2.0)
Requirement already satisfied: sympy==1.13.1 in /usr/local/lib/python3.11/dist-packages (from torch>=1.7->realesrgan) (1.13.1)
```

```
Downloading yapf-0.43.0-py3-none-any.whl (256 kB)
```

```
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 256.2/256.2 kB 10.9 MB/s eta 0:00:00
  Building wheels for collected packages: basicsr, filterpy
```

```
!pip install torchvision==0.16.1
```

```
Collecting torchvision==0.16.1
  Downloading torchvision-0.16.1-cp311-cp311-manylinux1_x86_64.whl.metadata (6.6 kB)
Requirement already satisfied: numpy in /usr/local/lib/python3.11/dist-packages (from torchvision==0.16.1) (2.0.2)
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-packages (from torchvision==0.16.1) (2.32.3)
Collecting torch==2.1.1 (from torchvision==0.16.1)
  Downloading torch-2.1.1-cp311-cp311-manylinux1_x86_64.whl.metadata (25 kB)
Requirement already satisfied: pillow!=8.3.*,>=5.3.0 in /usr/local/lib/python3.11/dist-packages (from torchvision==0.16.1) (11.1.0
Requirement already satisfied: filelock in /usr/local/lib/python3.11/dist-packages (from torch==2.1.1->torchvision==0.16.1) (3.18
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.11/dist-packages (from torch==2.1.1->torchvision==0.16
Requirement already satisfied: sympy in /usr/local/lib/python3.11/dist-packages (from torch==2.1.1->torchvision==0.16.1) (1.13.1)
Requirement already satisfied: networkx in /usr/local/lib/python3.11/dist-packages (from torch==2.1.1->torchvision==0.16.1) (3.4.2
Requirement already satisfied: jinja2 in /usr/local/lib/python3.11/dist-packages (from torch==2.1.1->torchvision==0.16.1) (3.1.6)
Requirement already satisfied: fsspec in /usr/local/lib/python3.11/dist-packages (from torch==2.1.1->torchvision==0.16.1) (2025.3
Collecting nvidia-cuda-nvrtc-cu12==12.1.105 (from torch==2.1.1->torchvision==0.16.1)
  Downloading nvidia_cuda_nvrtc_cu12-12.1.105-py3-none-manylinux1_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cuda-runtime-cu12==12.1.105 (from torch==2.1.1->torchvision==0.16.1)
  Downloading nvidia_cuda_runtime_cu12-12.1.105-py3-none-manylinux1_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cuda-cupti-cu12==12.1.105 (from torch==2.1.1->torchvision==0.16.1)
  Downloading nvidia_cuda_cupti_cu12-12.1.105-py3-none-manylinux1_x86_64.whl.metadata (1.6 kB)
Collecting nvidia-cudnn-cu12==8.9.2.26 (from torch==2.1.1->torchvision==0.16.1)
  Downloading nvidia_cudnn_cu12-8.9.2.26-py3-none-manylinux1_x86_64.whl.metadata (1.6 kB)
Collecting nvidia-cublas-cu12==12.1.3.1 (from torch==2.1.1->torchvision==0.16.1)
  Downloading nvidia_cublas_cu12-12.1.3.1-py3-none-manylinux1_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cufft-cu12==11.0.2.54 (from torch==2.1.1->torchvision==0.16.1)
  Downloading nvidia_cufft_cu12-11.0.2.54-py3-none-manylinux1_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-curand-cu12==10.3.2.106 (from torch==2.1.1->torchvision==0.16.1)
  Downloading nvidia_curand_cu12-10.3.2.106-py3-none-manylinux1_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cusolver-cu12==11.4.5.107 (from torch==2.1.1->torchvision==0.16.1)
  Downloading nvidia_cusolver_cu12-11.4.5.107-py3-none-manylinux1_x86_64.whl.metadata (1.6 kB)
Collecting nvidia-cusparse-cu12==12.1.0.106 (from torch==2.1.1->torchvision==0.16.1)
  Downloading nvidia_cusparse_cu12-12.1.0.106-py3-none-manylinux1_x86_64.whl.metadata (1.6 kB)
Collecting nvidia-nccl-cu12==2.18.1 (from torch==2.1.1->torchvision==0.16.1)
  Downloading nvidia_nccl_cu12-2.18.1-py3-none-manylinux1_x86_64.whl.metadata (1.8 kB)
Collecting nvidia-nvtx-cu12==12.1.105 (from torch==2.1.1->torchvision==0.16.1)
  Downloading nvidia_nvtx_cu12-12.1.105-py3-none-manylinux1_x86_64.whl.metadata (1.7 kB)
Collecting triton==2.1.0 (from torch==2.1.1->torchvision==0.16.1)
  Downloading triton-2.1.0-0-cp311-cp311-manylinux2014_x86_64.manylinux_2_17_x86_64.whl.metadata (1.3 kB)
Requirement already satisfied: nvidia-nvjitlink-cu12 in /usr/local/lib/python3.11/dist-packages (from nvidia-cusolver-cu12==11.4.5
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from requests->torchvision==0
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (from requests->torchvision==0.16.1) (3.10
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from requests->torchvision==0.16.1)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-packages (from requests->torchvision==0.16.1)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.11/dist-packages (from jinja2->torch==2.1.1->torchvision
Requirement already satisfied: mpmath<1.4,>=1.1.0 in /usr/local/lib/python3.11/dist-packages (from sympy->torch==2.1.1->torchvisi
Downloading torchvision-0.16.1-cp311-cp311-manylinux1_x86_64.whl (6.8 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 6.8/6.8 MB 32.2 MB/s eta 0:00:00
Downloading torch-2.1.1-cp311-cp311-manylinux1_x86_64.whl (670.2 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 670.2/670.2 MB 1.2 MB/s eta 0:00:00
Downloading nvidia_cublas_cu12-12.1.3.1-py3-none-manylinux1_x86_64.whl (410.6 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 410.6/410.6 MB 2.5 MB/s eta 0:00:00
Downloading nvidia_cuda_cupti_cu12-12.1.105-py3-none-manylinux1_x86_64.whl (14.1 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 14.1/14.1 MB 86.4 MB/s eta 0:00:00
Downloading nvidia_cuda_nvrtc_cu12-12.1.105-py3-none-manylinux1_x86_64.whl (23.7 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 23.7/23.7 MB 73.8 MB/s eta 0:00:00
Downloading nvidia_cuda_runtime_cu12-12.1.105-py3-none-manylinux1_x86_64.whl (823 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 823.6/823.6 kB 43.4 MB/s eta 0:00:00
Downloading nvidia_cudnn_cu12-8.9.2.26-py3-none-manylinux1_x86_64.whl (731.7 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 731.7/731.7 MB 2.6 MB/s eta 0:00:00
Downloading nvidia_cufft_cu12-11.0.2.54-py3-none-manylinux1_x86_64.whl (121.6 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 121.6/121.6 MB 7.3 MB/s eta 0:00:00
Downloading nvidia_curand_cu12-10.3.2.106-py3-none-manylinux1_x86_64.whl (56.5 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 56.5/56.5 MB 11.7 MB/s eta 0:00:00
Downloading nvidia_cusolver_cu12-11.4.5.107-py3-none-manylinux1_x86_64.whl (124.2 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 124.2/124.2 MB 7.3 MB/s eta 0:00:00
Downloading nvidia_cusparse_cu12-12.1.0.106-py3-none-manylinux1_x86_64.whl (196.0 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 196.0/196.0 MB 5.8 MB/s eta 0:00:00
Downloading nvidia_nccl_cu12-2.18.1-py3-none-manylinux1_x86_64.whl (209.8 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 209.8/209.8 MB 6.5 MB/s eta 0:00:00
Downloading nvidia_nvtx_cu12-12.1.105-py3-none-manylinux1_x86_64.whl (99 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 99.1/99.1 kB 7.6 MB/s eta 0:00:00
Downloading triton-2.1.0-0-cp311-cp311-manylinux2014_x86_64.manylinux_2_17_x86_64.whl (89.2 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 89.2/89.2 MB 8.8 MB/s eta 0:00:00
Installing collected packages: triton, nvidia-nvtx-cu12, nvidia-nccl-cu12, nvidia-cusparse-cu12, nvidia-curand-cu12, nvidia-cufft-
  Attempting uninstall: triton
    Found existing installation: triton 3.2.0
    Uninstalling triton-3.2.0:
      Successfully uninstalled triton-3.2.0
  Attempting uninstall: nvidia-nvtx-cu12
    Found existing installation: nvidia-nvtx-cu12 12.4.127
    Uninstalling nvidia-nvtx-cu12-12.4.127:
      Successfully uninstalled nvidia-nvtx-cu12-12.4.127
  Attempting uninstall: nvidia-nccl-cu12
    Found existing installation: nvidia-nccl-cu12 2.21.5
    Uninstalling nvidia-nccl-cu12-2.21.5:
      Successfully uninstalled nvidia-nccl-cu12-2.21.5
  Attempting uninstall: nvidia-cusparse-cu12
    Found existing installation: nvidia-cusparse-cu12 12.3.1.170
    Uninstalling nvidia-cusparse-cu12-12.3.1.170:
      Successfully uninstalled nvidia-cusparse-cu12-12.3.1.170
  Attempting uninstall: nvidia-curand-cu12
```

```
    Found existing installation: nvidia-curand-cu12 10.3.5.147
    Uninstalling nvidia-curand-cu12-10.3.5.147:
      Successfully uninstalled nvidia-curand-cu12-10.3.5.147
  Attempting uninstall: nvidia-cufft-cu12
    Found existing installation: nvidia-cufft-cu12 11.2.1.3
    Uninstalling nvidia-cufft-cu12-11.2.1.3:
      Successfully uninstalled nvidia-cufft-cu12-11.2.1.3
  Attempting uninstall: nvidia-cuda-runtime-cu12
    Found existing installation: nvidia-cuda-runtime-cu12 12.4.127
    Uninstalling nvidia-cuda-runtime-cu12-12.4.127:
      Successfully uninstalled nvidia-cuda-runtime-cu12-12.4.127
  Attempting uninstall: nvidia-cuda-nvrtc-cu12
    Found existing installation: nvidia-cuda-nvrtc-cu12 12.4.127
    Uninstalling nvidia-cuda-nvrtc-cu12-12.4.127:
      Successfully uninstalled nvidia-cuda-nvrtc-cu12-12.4.127
  Attempting uninstall: nvidia-cuda-cupti-cu12
    Found existing installation: nvidia-cuda-cupti-cu12 12.4.127
    Uninstalling nvidia-cuda-cupti-cu12-12.4.127:
      Successfully uninstalled nvidia-cuda-cupti-cu12-12.4.127
  Attempting uninstall: nvidia-cublas-cu12
    Found existing installation: nvidia-cublas-cu12 12.4.5.8
    Uninstalling nvidia-cublas-cu12-12.4.5.8:
      Successfully uninstalled nvidia-cublas-cu12-12.4.5.8
  Attempting uninstall: nvidia-cusolver-cu12
    Found existing installation: nvidia-cusolver-cu12 11.6.1.9
    Uninstalling nvidia-cusolver-cu12-11.6.1.9:
      Successfully uninstalled nvidia-cusolver-cu12-11.6.1.9
  Attempting uninstall: nvidia-cudnn-cu12
    Found existing installation: nvidia-cudnn-cu12 9.1.0.70
    Uninstalling nvidia-cudnn-cu12-9.1.0.70:
      Successfully uninstalled nvidia-cudnn-cu12-9.1.0.70
  Attempting uninstall: torch
    Found existing installation: torch 2.6.0+cu124
    Uninstalling torch-2.6.0+cu124:
      Successfully uninstalled torch-2.6.0+cu124
  Attempting uninstall: torchvision
    Found existing installation: torchvision 0.21.0+cu124
    Uninstalling torchvision-0.21.0+cu124:
      Successfully uninstalled torchvision-0.21.0+cu124
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This behaviour is the s
torchaudio 2.6.0+cu124 requires torch==2.6.0, but you have torch 2.1.1 which is incompatible.
Successfully installed nvidia-cublas-cu12-12.1.3.1 nvidia-cuda-cupti-cu12-12.1.105 nvidia-cuda-nvrtc-cu12-12.1.105 nvidia-cuda-ru
WARNING: The following packages were previously imported in this runtime:
  [torch,torchgen,torchvision,triton]
You must restart the runtime in order to use newly installed versions.
```

RESTART SESSION

```
!pip install numpy==1.24.4
```

```
Collecting numpy==1.24.4
  Downloading numpy-1.24.4-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (5.6 kB)
Downloading numpy-1.24.4-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (17.3 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 17.3/17.3 MB 48.1 MB/s eta 0:00:00
Installing collected packages: numpy
  Attempting uninstall: numpy
    Found existing installation: numpy 2.0.2
    Uninstalling numpy-2.0.2:
      Successfully uninstalled numpy-2.0.2
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This behaviour is the sou
pymc 5.21.2 requires numpy>=1.25.0, but you have numpy 1.24.4 which is incompatible.
tensorflow 2.18.0 requires numpy<2.1.0,>=1.26.0, but you have numpy 1.24.4 which is incompatible.
treescope 0.1.9 requires numpy>=1.25.2, but you have numpy 1.24.4 which is incompatible.
jax 0.5.2 requires numpy>=1.25, but you have numpy 1.24.4 which is incompatible.
blosc2 3.3.0 requires numpy>=1.26, but you have numpy 1.24.4 which is incompatible.
thinc 8.3.6 requires numpy<3.0.0,>=2.0.0, but you have numpy 1.24.4 which is incompatible.
jaxlib 0.5.1 requires numpy>=1.25, but you have numpy 1.24.4 which is incompatible.
Successfully installed numpy-1.24.4
WARNING: The following packages were previously imported in this runtime:
  [numpy]
You must restart the runtime in order to use newly installed versions.
```

RESTART SESSION

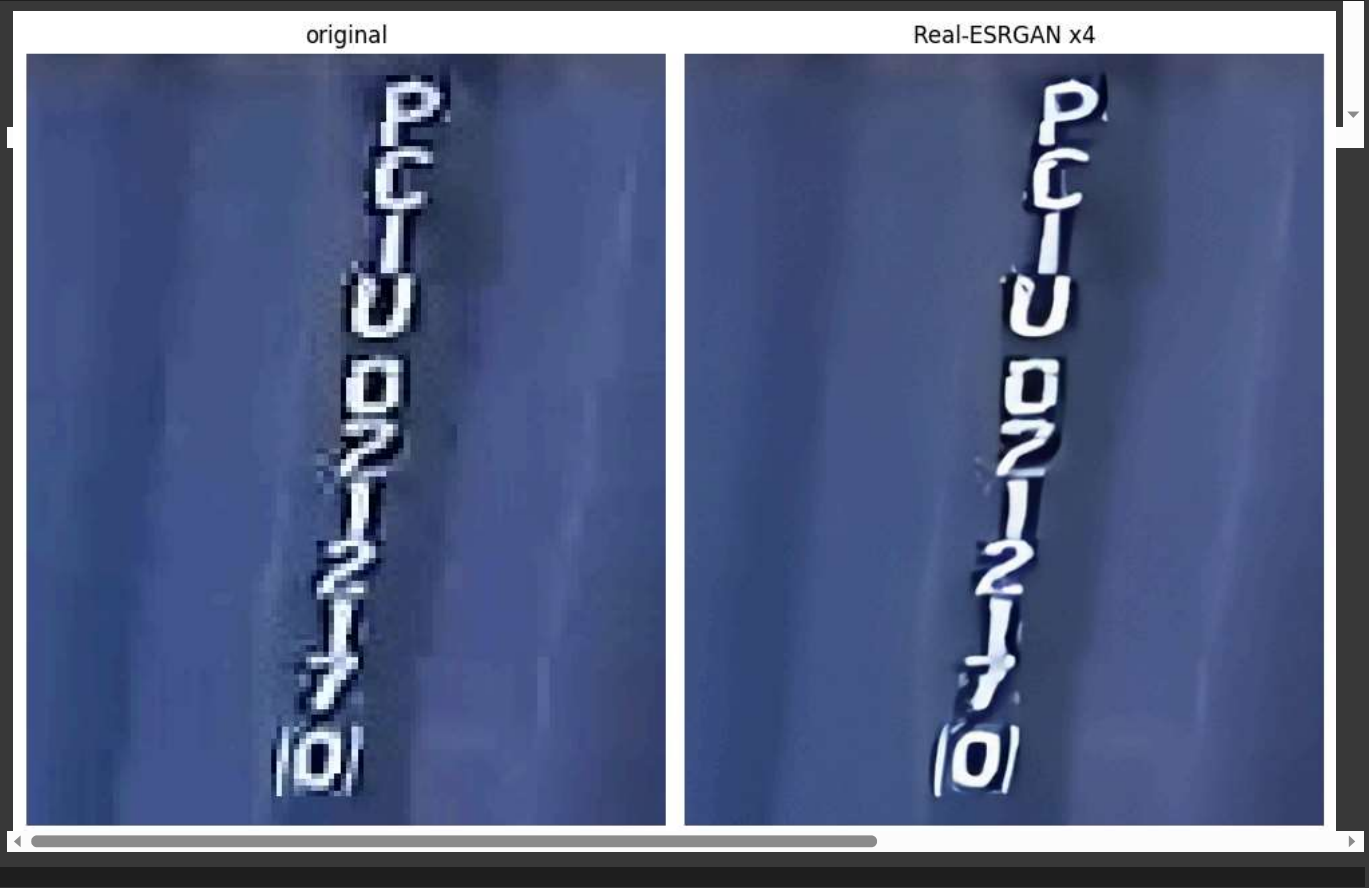
```
#  Real-ESRGAN Super Resolution
# Step 1: Clone the repo and install dependencies
!git clone https://github.com/xinntao/Real-ESRGAN.git
%cd Real-ESRGAN
!pip install -r requirements.txt
!python setup.py develop

# Step 2: Download Pre-trained Model
!wget https://github.com/xinntao/Real-ESRGAN/releases/download/v0.1.0/RealESRGAN_x4plus.pth -P weights
```

```
Adding Markdown 3.7 to easy-install.pth file
Installing markdown_py script to /usr/local/bin

Using /usr/local/lib/python3.11/dist-packages
Searching for grpcio==1.71.0
Best match: grpcio 1.71.0
Adding grpcio 1.71.0 to easy-install.pth file
```

```
# Step 3: Upload your image
from google.colab import files
uploaded = files.upload()

# Step 4: Run Super Resolution
!python inference_realesrgan.py -n RealESRGAN_x4plus -i task1_img2.jpg --outscale 4 --fp32
```

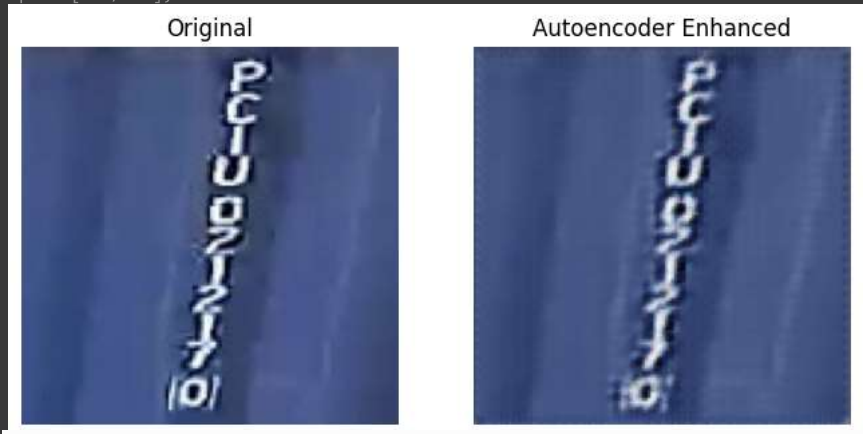
```
Choose Files task1_img2.jpg
• task1_img2.jpg(image/jpeg) - 4873 bytes, last modified: 4/10/2025 - 100% done
Saving task1_img2.jpg to task1_img2 (1).jpg
/usr/local/lib/python3.11/dist-packages/torchvision/transforms/functional_tensor.py:5: UserWarning: The torchvision.transforms.funct
  warnings.warn(
Testing 0 task1_img2
```

```
import cv2
from PIL import Image
import matplotlib.pyplot as plt

# Load images
esrgan = Image.open("/content/Real-ESRGAN/results/task1_img2_out.jpg")
original = Image.open("/content/task1_img2.jpg")

# Plot
titles = [ "original","Real-ESRGAN x4"]
images = [original , esrgan]

plt.figure(figsize=(10, 10))
for i, (img, title) in enumerate(zip(images, titles)):
    plt.subplot(1, 2, i+1)
    plt.imshow(img)
    plt.title(title)
    plt.axis('off')

plt.tight_layout()
plt.show()
```



```
import torch
import torch.nn as nn
import torchvision.transforms as T
from PIL import Image
import matplotlib.pyplot as plt
import numpy as np

# ✅ Step 1: Define Autoencoder
class Autoencoder(nn.Module):
    def __init__(self):
        super(Autoencoder, self).__init__()
        self.encoder = nn.Sequential(
            nn.Conv2d(3, 16, 3, stride=2, padding=1),  # 128 -> 64
            nn.ReLU(),
            nn.Conv2d(16, 8, 3, stride=2, padding=1),  # 64 -> 32
            nn.ReLU()
        )
        self.decoder = nn.Sequential(
            nn.ConvTranspose2d(8, 16, 3, stride=2, padding=1, output_padding=1),  # 32 -> 64
            nn.ReLU(),
            nn.ConvTranspose2d(16, 3, 3, stride=2, padding=1, output_padding=1),  # 64 -> 128
            nn.Sigmoid()
        )

    def forward(self, x):
        x = self.encoder(x)
        x = self.decoder(x)
        return x

# ✅ Step 2: Load and preprocess image
def load_image(image_path):
    img = Image.open(image_path).convert('RGB').resize((128, 128))
    transform = T.ToTensor()
    return transform(img).unsqueeze(0)  # Shape: (1, 3, H, W)

def tensor_to_image(tensor):
    img = tensor.squeeze().detach().numpy().transpose(1, 2, 0)
    return (img * 255).astype(np.uint8)

# ✅ Step 3: Training loop on the same image (simple enhancement)
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model = Autoencoder().to(device)
image_path = "task1_img2.jpg"
img_tensor = load_image(image_path).to(device)

criterion = nn.MSELoss()
optimizer = torch.optim.Adam(model.parameters(), lr=1e-3)

for epoch in range(300):
    output = model(img_tensor)
    loss = criterion(output, img_tensor)
    optimizer.zero_grad()
    loss.backward()
    optimizer.step()
    if epoch % 100 == 0:
        print(f"Epoch [{epoch}/300], Loss: {loss.item():.4f}")

# ✅ Step 4: Get the output
output_tensor = model(img_tensor).cpu()
enhanced_image = tensor_to_image(output_tensor)

# ✅ Step 5: Show result
original = Image.open(image_path).resize((128, 128))
plt.figure(figsize=(8, 4))
plt.subplot(1, 2, 1)
plt.imshow(original)
plt.title("Original")
plt.axis('off')

plt.subplot(1, 2, 2)
plt.imshow(enhanced_image)
plt.title("Autoencoder Enhanced")
plt.axis('off')
plt.show()

# ✅ Save if needed
Image.fromarray(enhanced_image).save("autoencoder_enhanced.jpg")
```

```
Epoch [0/300], Loss: 0.0368
Epoch [100/300], Loss: 0.0117
Epoch [200/300], Loss: 0.0059
```



SWinIR Model

```
# # 1. تحميل مستودع SwinIR واستخدام إصدار محدد
# !git clone https://github.com/JingyunLiang/SwinIR.git
# %cd SwinIR
# !git checkout 8f1811e  # استخدام commit من وقت إصدار النموذج

# 2. تحميل النموذج المدرب مسبقًا (SwinIR-M x4 الكلاسيكي)
# !wget https://github.com/JingyunLiang/SwinIR/releases/download/v0.0/001_classicalSR_DF2K_s64w8_SwinIR-M_x4.pth -P experiments/pretrain
```

```
import os
import shutil
import torch
import cv2
import numpy as np
from PIL import Image
import matplotlib.pyplot as plt
from google.colab import files


# 3. إعداد المجلدات
upload_folder = 'test_images'
result_folder = 'results'

if os.path.isdir(upload_folder):
    shutil.rmtree(upload_folder)
if os.path.isdir(result_folder):
    shutil.rmtree(result_folder)
os.mkdir(upload_folder)
os.mkdir(result_folder)

# 4. تحميل الصورة
uploaded = files.upload()
for filename in uploaded.keys():
    dst_path = os.path.join(upload_folder, filename)
    print(f'Moving {filename} to {dst_path}')
    shutil.move(filename, dst_path)

# 5. تحميل نموذج SwinIR
from models.network_swinir import SwinIR as net

device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
model_path = 'experiments/pretrained_models/001_classicalSR_DF2K_s64w8_SwinIR-M_x4.pth'
model = net(upscale=4, in_chans=3, img_size=64, window_size=8,
            img_range=1., depths=[6, 6, 6, 6, 6, 6], embed_dim=180,
            num_heads=[6, 6, 6, 6, 6, 6], mlp_ratio=2, upsampler='pixelshuffle',
            resi_connection='1conv')
model.load_state_dict(torch.load(model_path)['params'], strict=False)
model.eval()
model = model.to(device)

# 6. تحسين الصورة
image_path = os.path.join(upload_folder, os.listdir(upload_folder)[0])  # أول صورة في المجلد
img = cv2.imread(image_path, cv2.IMREAD_COLOR)
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
img_tensor = torch.from_numpy(np.transpose(img, (2, 0, 1))).float() / 255.0
img_tensor = img_tensor.unsqueeze(0).to(device)
```

```
with torch.no_grad():
    output = model(img_tensor)
output = output.squeeze().cpu().numpy()
output = np.transpose(output, (1, 2, 0))
output = np.clip(output * 255.0, 0, 255).astype(np.uint8)

# 7. تحويل الصور للعرض
original_img = Image.fromarray(img)  # الصورة الأصلية
enhanced_img = Image.fromarray(output)  # الصورة المحسنة

# 8. عرض الصورتين جنب بعض
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
plt.imshow(original_img)
plt.title('Original')
plt.axis('off')

plt.subplot(1, 2, 2)
plt.imshow(enhanced_img)
plt.title('SwinIR Enhanced')
plt.axis('off')

plt.tight_layout()
plt.show()

# 9. حفظ الصورة المحسنة
enhanced_path = os.path.join(result_folder, 'enhanced_' + os.path.basename(image_path))
cv2.imwrite(enhanced_path, cv2.cvtColor(output, cv2.COLOR_RGB2BGR))
print(f'Enhanced image saved at: {enhanced_path}')
```

Choose Files task1_img1.jpg

- **task1_img1.jpg**(image/jpeg) - 13801 bytes, last modified: 4/10/2025 - 100% done

```
Saving task1_img1.jpg to task1_img1.jpg
Moving task1_img1.jpg to test_images/task1_img1.jpg
```


```
Enhanced image saved at: results/enhanced_task1_img1.jpg
```

Summary

Task 1 – Image Enhancement Summary

We applied multiple enhancement techniques to improve the clarity and contrast of the given images:

- **Histogram Equalization**: Used to improve global contrast.
- **CLAHE (Contrast Limited Adaptive Histogram Equalization)**: Applied to enhance local details without overexposure, especially effective on image 1 which had poor lighting.
- **Non-local Means Denoising**: Removed image noise while preserving edges.
- **Bilateral Filtering + Sharpening**: Further enhanced image texture and reduced noise.
- **Adaptive Gamma Correction**: Customized contrast tuning based on the brightness of each image.

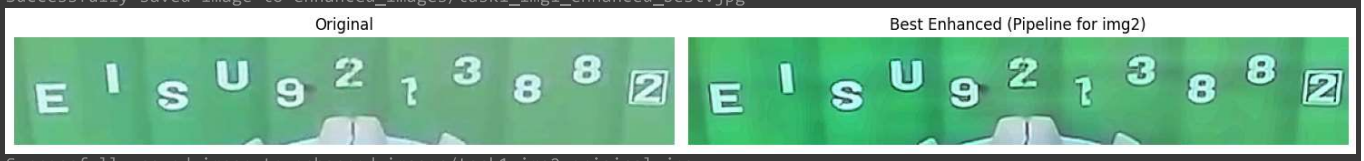
Observations:

- Image 2 was very dark and noisy, so stronger contrast and denoising were applied.
  - Best SSIM Score for task1_img2: Adaptive Gamma Correction with SSIM =**0.9601**
- Image 1 was relatively clear, so milder enhancement was enough.
  - Best SSIM Score for task1_img1: Bilateral + Sharpen with SSIM = **0.8667**
- We applied both pipelines to each image regardless of its type, and computed the Structural Similarity Index (SSIM) to evaluate the quality of enhancement compared to the original.
  - Only the best result (with the highest SSIM score) was selected, displayed, and saved for future use in OCR.
  - This approach ensured that the enhancement process was adaptive, data-driven, and optimized per image rather than relying on fixed assumptions.

```
def process_image_with_combined_pipelines(image_path, is_img1=True, save_dir="enhanced_images"):
    """
    Process the image with both combined pipelines (img1 and img2), compute SSIM for comparison,
    display only the best result, and save it.
```

```
    """
    # Load the image
    img = cv2.imread(image_path)
    if img is None:
        print(f"Error: Could not load image {image_path}. Check the file path.")
        return

    if not os.path.exists(save_dir):
        os.makedirs(save_dir)

    base_filename = os.path.splitext(os.path.basename(image_path))[0]
    original_save_path = os.path.join(save_dir, f"{base_filename}_original.jpg")
    save_image(img, original_save_path)

    # Apply both pipelines
    enhanced_img1 = enhance_methods_combined_img1(img)
    enhanced_img2 = enhance_methods_combined_img2(img)

    # Compute SSIM
    ssim_img1 = compute_ssim(img, enhanced_img1)
    ssim_img2 = compute_ssim(img, enhanced_img2)

    # Determine best
    if ssim_img1 > ssim_img2:
        best_img = enhanced_img1
        best_pipeline = "Pipeline for img1"
        best_ssim = ssim_img1
    else:
        best_img = enhanced_img2
        best_pipeline = "Pipeline for img2"
        best_ssim = ssim_img2

    print(f"\nBest Pipeline for {base_filename}: {best_pipeline} with SSIM = {best_ssim:.4f}\n")

    # Save and show only best enhanced image
    enhanced_save_path = os.path.join(save_dir, f"{base_filename}_enhanced_best.jpg")
    save_image(best_img, enhanced_save_path)
    show_images([img, best_img], ["Original", f"Best Enhanced ({best_pipeline})"])

if __name__ == "__main__":
    # Process both images
    process_image_with_combined_pipelines("task1_img1.jpg", is_img1=True, save_dir="enhanced_images")
    process_image_with_combined_pipelines("task1_img2.jpg", is_img1=False, save_dir="enhanced_images")
```

```
Successfully saved image to enhanced_images/task1_img1_original.jpg

Best Pipeline for task1_img1: Pipeline for img2 with SSIM = 0.9182

Successfully saved image to enhanced_images/task1_img1_enhanced_best.jpg
```




```
Successfully saved image to enhanced_images/task1_img2_original.jpg

Best Pipeline for task1_img2: Pipeline for img2 with SSIM = 0.8840

Successfully saved image to enhanced_images/task1_img2_enhanced_best.jpg
```

3.2 Task 2: Text Extraction (OCR)

Objective:

Extract highly accurate text from the enhanced images from Task 1.

📝 Task 2 Report – Text Extraction (OCR)

📌 Objective:

Extract text from the enhanced images using different OCR tools and evaluate their accuracy.

🛠 Tools Tested:

- **Tesseract**
- **EasyOCR**
- **Paddleocr**
- **EAST**
- **docTR (transformer-based)**
- **CRAFT (attempted, failed due to OpenCV issues)**
- **DBnet + CRNN**

📊 Results Summary:

Tool	task1_img1 Result	task1_img2 Result
Tesseract	i) WU wc Coe all	See
EasyOCR	E " 8 U 9 2 ? 8 8 8	|
PaddleOCR	E (conf: 1.00), 2 (conf: 0.98)	— (to be tested)

🧠 Observations:

- **Tesseract** produced more readable output but with artifacts.
- EasyOCR returned cleaner, less meaningful characters.
- **docTR**, although powerful, requires minimum input size and didn't perform well on small images without resizing/padding.
- Attempting to use **CRAFT** failed due to package version compatibility (OpenCV 4.5.4.60 couldn't be built).
- **PaddleOCR** successfully detected and recognized text with high confidence, but it did not extract the full content from the image.
- **EAST Model**: The model could not be loaded either using OpenCV or TensorFlow. Errors occurred related to file compatibility or format issues with the frozen_east_text_detection.pb file.
- **DBNet** failed to execute due to installation issues. Despite multiple attempts to install and use it, the tool did not work as expected. The installation encountered compatibility issues, and after following the setup instructions, the tool was not functional.

🧪 Conclusion:

No OCR tool gave fully reliable results out-of-the-box. The images—despite enhancement—still require advanced preprocessing or a full DL-based pipeline.

💡 Proposed DL-based Pipeline:

1. **Text Detection**: Use CRAFT or DBNet for region detection
2. **Cropping Text Areas**
3. **Text Recognition**: Use CRNN or TrOCR

```
Start coding or generate with AI.
```

Tesseract

```
def run_tesseract(image_path):
    """
    Run Tesseract OCR on a given image and return extracted text.
    """
    img = cv2.imread(image_path)
    img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    text = pytesseract.image_to_string(img_rgb)
    print("Extracted Text:\n")
    print(text)
    return text

# Example usage
text1 = run_tesseract("enhanced_images/task1_img1_enhanced_best.jpg")
text2 = run_tesseract("enhanced_images/task1_img2_enhanced_best.jpg")
```

```
Extracted Text:
```

EasyOCR

```
def run_easyocr(image_path):
    """
    Run EasyOCR on a given image and return extracted text.
    """
    result = reader.readtext(image_path, detail=0)
    text = "\n".join(result)
    print("Extracted Text with EasyOCR:\n")
    print(text)
    return text

# Example usage:
easy_text1 = run_easyocr("enhanced_images/task1_img1_enhanced_best.jpg")
easy_text2 = run_easyocr("enhanced_images/task1_img2_enhanced_best.jpg")
```

```
Extracted Text with EasyOCR:

8
U
9
2
1
```