# Image Denoising Report

This report outlines the process of image denoising using various techniques and provides visualizations of the denoised images. The code snippet provided demonstrates the denoising process for a specific category of images, specifically benign cases. Here are the key steps involved:

### Directory Definitions

Directories are defined to store the original images, noisy images, and denoised images for each category: benign, malignant, and normal. Separate directories are created to store the noisy and denoised images.

```
# Define the directories containing the images for each category
benign = "./Bengin cases"
malignant = "./Malignant cases"
normal = "./Normal cases"

# Define the directories to store the noisy images for each category
benign_noisy = "benign_noisy"
malignant_noisy = "malignant_noisy"
normal_noisy = "normal_noisy"

# Define the directories to store the denoised images for each category
benign_denoised = "benign_denoised"
malignant_denoised = "malignant_denoised"
normal_denoised = "normal_denoised"

# Create the new directories to store the noisy and denoised images
os.makedirs(benign_noisy, exist_ok=True)
os.makedirs(malignant_noisy, exist_ok=True)
os.makedirs(normal_noisy, exist_ok=True)
os.makedirs(benign_denoised, exist_ok=True)
os.makedirs(malignant_denoised, exist_ok=True)
os.makedirs(normal_denoised, exist_ok=True)
```

**Find the number of noisy images per directory**

The mean squared error  and the peak signal noise ratio are used to determine if an image is noisy. The following screenshot shows the total number of noisy images per directory

```
    # Print the results
    print("Noisy Images in Benign Cases:", benign_noisy_count)
    print("Noisy Images in Malignant Cases:", malignant_noisy_count)
    print("Noisy Images in Normal Cases:", normal_noisy_count)

[2]  ✓ 2m 7.9s

Noisy Images in Benign Cases: 120
Noisy Images in Malignant Cases: 561
Noisy Images in Normal Cases: 416
```

**Generating Noisy Image**

A noisy version of the image is generated by adding Gaussian noise using the skimage.util.random_noise() function.

```
# Generate a noisy version of the image (e.g. by adding Gaussian noise)
noisy_image = skimage.util.random_noise(image, mode='gaussian')
```

**Calculating MSE and SNR**

The Mean Squared Error (MSE) and Signal-to-Noise Ratio (SNR) are calculated to evaluate the noise level in the image. These metrics are computed using the skimage.metrics.mean_squared_error() and skimage.metrics.peak_signal_noise_ratio() functions, respectively.

```
# Calculate the MSE
mse = mean_squared_error(image, noisy_image)

# Calculate the SNR
snr = peak_signal_noise_ratio(image, noisy_image)
```

### Checking Noise Level

The MSE and SNR values are used to determine whether the image is considered noisy. If the MSE is above the defined threshold and the SNR is below the defined threshold, the image is considered noisy.

```
# Check if the image is noisy based on the MSE and SNR values
if mse > mse_threshold and snr < snr_threshold:
    # Move the noisy image to the new directory
    shutil.move(image_path, os.path.join(benign_noisy, file_name))
```

### Handling Noisy Images

For noisy images, the following steps are performed:

### Moving the Noisy Image

The noisy image is moved to the corresponding directory for storing noisy images using the shutil.copy() function.

```
    # Check if the image is noisy based on the MSE and SNR values
    if mse > mse_threshold and snr < snr_threshold:
        # Move the noisy image to the new directory
        shutil.copy(image_path, os.path.join(benign_noisy, file_name))
```

### Denoising the Image with DWT

The noisy image is denoised using the Discrete Wavelet Transform (DWT) technique. The DWT coefficients are calculated using the pywt.dwt2() function, and denoising is performed using the denoise_wavelet() function. The denoised image is reconstructed using the inverse DWT (pywt.idwt2()) and then clipped to the valid pixel value range. The resulting denoised image is saved as an 8-bit grayscale image using the skimage.io.imsave() function.

```
# Denoise the noisy image using DWT
coeffs = pywt.dwt2(noisy_image, 'haar')
coeffs = list(coeffs)
coeffs[0] = denoise_wavelet(coeffs[0], method='VisuShrink', mode='soft', sigma=0.1, wavelet='haar')
reconstructed_image = pywt.idwt2(coeffs, 'haar')
denoised_image = np.clip(reconstructed_image, 0, 1)
denoised_image = (denoised_image * 255).astype(np.uint8)
```

### Identifying Noise Type and Denoising Method

The type of noise removed (in this case, Gaussian noise) and the denoising method (DWT) are identified.

```
# Identify the type of noise removed
removed_noise = "Gaussian"
denoising_method = "DWT"
    # Load the original image
original_image = Image.open(image_path)
```

### Displaying Results

Various visualizations and information are displayed for the denoised image, including the noise density, the original image with noise, the denoised image, and the difference image between the original and denoised versions.

```
    # Create a figure with two subplots: original image and denoised image
    fig, axs = plt.subplots(1, 2, figsize=(5, 4))
    axs[0].imshow(original_image)
    axs[0].set_title("Noisy Image - {}".format(removed_noise))
    axs[1].imshow(denoised_image)
    axs[1].set_title("Noise removed Image by using  - {}".format(denoising_method))

    # Display the difference image
    # fig, ax = plt.subplots(figsize=(4, 4))
    axs[1].imshow(difference_image)
    axs[1].set_title("Noise removed by {}".format(denoising_method))
    plt.tight_layout()
    plt.show()

    print("--------------------")
```

**Visualization**

To provide visualizations of the denoised images and the difference between the original and denoised images, the following steps are performed:

The original image with noise and the denoised image are displayed side by side in a figure using the matplotlib.pyplot.imshow() function.

The difference image between the original and denoised images is calculated using ImageChops.difference().

The difference image and denoised image are displayed in a subplot for visual comparison.

```
C:\Users\Jack\AppData\Local\Temp\ipykernel_4572\3010624713.py:56: UserWarning: Inputs have
  snr = peak_signal_noise_ratio(image, noisy_image)
Noise density: 15088.2723
Original Image - Noise Type: Gaussian
Denoised Image - Method: DWT
```