workingmodel

November 7, 2024

[1]:
```python
# Step 1: Mount Google Drive
from google.colab import drive
drive.mount('/content/drive')

# Step 2: Set up Dataset Paths in Google Drive
train_path = "/content/drive/My Drive/dataset noise/train"
val_path = "/content/drive/My Drive/dataset noise/valid"
test_path = "/content/drive/My Drive/dataset noise/test"

# Step 3: Import Libraries
from sklearn.metrics import precision_score, recall_score, accuracy_score
import torch
import torch.nn as nn
import torch.optim as optim
import torch.nn.functional as F
from torchvision import transforms, datasets
from torch.utils.data import DataLoader, Dataset
from tqdm import tqdm
import matplotlib.pyplot as plt
import numpy as np
from skimage import color, measure, img_as_float
from PIL import Image
import os
```

```
Mounted at /content/drive
```

[2]:
```python
# Data transformations
transform = transforms.Compose([
    transforms.Resize((64, 64)),
    transforms.ToTensor(),
    transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))
])

# Custom Dataset Class
class ImageLabelDataset(Dataset):
    def __init__(self, image_folder, label_folder, transform=None):
        self.image_folder = image_folder
        self.label_folder = label_folder
```

```
        self.transform = transform
        self.image_files = [f for f in os.listdir(image_folder) if f.endswith('.
 ↪jpg')]

    def __len__(self):
        return len(self.image_files)

    def __getitem__(self, idx):
        image_path = os.path.join(self.image_folder, self.image_files[idx])
        image = Image.open(image_path).convert("RGB")

        # Load corresponding label file
        label_name = f"{os.path.splitext(self.image_files[idx])[0]}.txt"
        label_path = os.path.join(self.label_folder, label_name)

        # Read label contents
        with open(label_path, 'r') as label_file:
            label_data = label_file.read()  # Adjust this if your label file␣
 ↪has a specific structure

        if self.transform:
            image = self.transform(image)

        return image, label_data

# Define paths for train, val, and test data
train_image_path = os.path.join(train_path, "images")
train_label_path = os.path.join(train_path, "labels")
val_image_path = os.path.join(val_path, "images")
val_label_path = os.path.join(val_path, "labels")
test_image_path = os.path.join(test_path, "images")
test_label_path = os.path.join(test_path, "labels")

# Create datasets and dataloaders
train_data = ImageLabelDataset(train_image_path, train_label_path,␣
 ↪transform=transform)
val_data = ImageLabelDataset(val_image_path, val_label_path,␣
 ↪transform=transform)
test_data = ImageLabelDataset(test_image_path, test_label_path,␣
 ↪transform=transform)

train_loader = DataLoader(train_data, batch_size=8, shuffle=True, num_workers=2)
val_loader = DataLoader(val_data, batch_size=8, shuffle=False, num_workers=2)
test_loader = DataLoader(test_data, batch_size=8, shuffle=False, num_workers=2)
```

[3]:
```python
# High-Low Frequency Separation
class HighLowFrequencySeparation:
    def __init__(self, kernel_size=5, sigma=1.5):
        self.gaussian_blur = transforms.GaussianBlur(kernel_size=kernel_size,␣
 ↪sigma=sigma)

    def separate(self, image):
        low_freq = self.gaussian_blur(image)
        high_freq = torch.abs(image - low_freq)
        return high_freq, low_freq

# ConvGroup Module
class ConvGroup(nn.Module):
    def __init__(self, in_channels):
        super(ConvGroup, self).__init__()
        self.conv = nn.Conv2d(in_channels, in_channels, kernel_size=3,␣
 ↪padding=1, groups=in_channels)
        self.bn = nn.BatchNorm2d(in_channels)
        self.dwconv = nn.Conv2d(in_channels, in_channels, kernel_size=3,␣
 ↪padding=1, groups=in_channels)

    def forward(self, x):
        out = F.relu(self.bn(self.conv(x)))
        out = F.relu(self.bn(self.dwconv(out)))
        return out + x

# CrossAttention Module
class CrossAttention(nn.Module):
    def __init__(self, in_channels, r=4):
        super(CrossAttention, self).__init__()
        self.pool_avg = nn.AdaptiveAvgPool2d(1)
        self.pool_max = nn.AdaptiveMaxPool2d(1)
        intermediate_channels = max(1, in_channels // r)
        self.fc1 = nn.Conv2d(in_channels, intermediate_channels, 1)
        self.fc2 = nn.Conv2d(intermediate_channels, in_channels, 1)

    def forward(self, x):
        avg = self.pool_avg(x)
        maxp = self.pool_max(x)
        avg_out = self.fc2(F.relu(self.fc1(avg)))
        max_out = self.fc2(F.relu(self.fc1(maxp)))
        return avg_out * max_out + x

# GCE Module
class GCE(nn.Module):
    def __init__(self, in_channels):
        super(GCE, self).__init__()
```

```
        self.conv_group = ConvGroup(in_channels)
        self.cross_attention = CrossAttention(in_channels)

    def forward(self, x):
        x = self.conv_group(x)
        x = self.cross_attention(x)
        return x

# ResidualBlock Module
class ResidualBlock(nn.Module):
    def __init__(self, in_channels):
        super(ResidualBlock, self).__init__()
        self.conv1 = nn.Conv2d(in_channels, in_channels, kernel_size=3,␣
 ↪padding=1)
        self.inorm = nn.InstanceNorm2d(in_channels)
        self.prelu = nn.PReLU()

    def forward(self, x):
        out = self.prelu(self.inorm(self.conv1(x)))
        out = self.conv1(out)
        return out + x

# HHDNet Model
class HHDNet(nn.Module):
    def __init__(self, in_channels=3):
        super(HHDNet, self).__init__()
        self.high_low_sep = HighLowFrequencySeparation()
        self.high_freq_branch = nn.Sequential(
            *[GCE(in_channels) for _ in range(8)]
        )
        self.low_freq_branch = nn.Sequential(
            *[ResidualBlock(in_channels) for _ in range(4)]
        )
        self.fusion = nn.Conv2d(in_channels * 2, in_channels, kernel_size=3,␣
 ↪padding=1)

    def forward(self, x):
        high_freq, low_freq = self.high_low_sep.separate(x)
        high_freq_out = self.high_freq_branch(high_freq)
        low_freq_out = self.low_freq_branch(low_freq)
        concat_out = torch.cat([high_freq_out, low_freq_out], dim=1)
        return self.fusion(concat_out) + x
```

[4]:
```
import torch.nn.functional as F

def calculate_psnr(target, output):
    # Ensure MSE is calculated appropriately for better range control
```

```
    mse = F.mse_loss(output, target)
    if mse == 0:
        return 100
    return 20 * torch.log10(1.0 / torch.sqrt(mse))

# Optionally, introduce a perceptual loss or SSIM for enhanced quality
```

[5]:
```
def calculate_pixel_metrics(target, output, threshold=0.1):
    """
    Calculate pixel-wise precision, recall, and accuracy
    using a threshold for the pixel error.
    """
    # Binarize images using the threshold
    target_bin = (target > threshold).cpu().numpy().flatten()
    output_bin = (output > threshold).cpu().numpy().flatten()

    # Calculate precision, recall, accuracy
    precision = precision_score(target_bin, output_bin)
    recall = recall_score(target_bin, output_bin)
    accuracy = accuracy_score(target_bin, output_bin)

    return precision, recall, accuracy
```

[6]:
```
import numpy as np
from skimage import color

def calculate_uciqe(output):
    output = unnormalize(output)  # Unnormalize to bring values back between 0␣
↪and 1
    output = output.permute(1, 2, 0).cpu().numpy()  # Convert to numpy
    contrast = np.std(output)
    mean_saturation = np.mean(color.rgb2hsv(output)[:, :, 1])
    std_hue = np.std(color.rgb2hsv(output)[:, :, 0])
    return 0.4680 * contrast + 0.2745 * mean_saturation + 0.2576 * std_hue

def calculate_uiqm(output):
    output = unnormalize(output)
    output = output.permute(1, 2, 0).cpu().numpy()
    uicm = np.mean(color.rgb2lab(output)[:, :, 1:3])  # Colorfulness
    uism = np.std(output)  # Sharpness
    uiconm = np.mean(output)  # Contrast
    return 0.0282 * uicm + 0.2953 * uism + 3.5753 * uiconm
```

[7]:
```
import torch.nn as nn

class Discriminator(nn.Module):
    def __init__(self):
```

```
        super(Discriminator, self).__init__()
        self.main = nn.Sequential(
            # input is (nc) x 64 x 64
            nn.Conv2d(3, 64, 4, 2, 1, bias=False),
            nn.LeakyReLU(0.2, inplace=True),
            # state size. (ndf) x 32 x 32
            nn.Conv2d(64, 128, 4, 2, 1, bias=False),
            nn.BatchNorm2d(128),
            nn.LeakyReLU(0.2, inplace=True),
            # state size. (ndf*2) x 16 x 16
            nn.Conv2d(128, 256, 4, 2, 1, bias=False),
            nn.BatchNorm2d(256),
            nn.LeakyReLU(0.2, inplace=True),
            # state size. (ndf*4) x 8 x 8
            nn.Conv2d(256, 512, 4, 2, 1, bias=False),  # Reduced kernel size to
↪3x3
            nn.BatchNorm2d(512),
            nn.LeakyReLU(0.2, inplace=True),
            # state size. (ndf*8) x 4 x 4
            nn.Conv2d(512, 1, 3, 1, 1, bias=False), # Reduced kernel size to
↪3x3, Adjusted padding to 1
            # state size. 1 x 4 x 4 (or smaller based on other changes)
        )
        self.pool = nn.AdaptiveAvgPool2d((1, 1)) #changed here
        self.flatten = nn.Flatten()

    def forward(self, x):
        x = self.main(x)
        x = self.pool(x) #changed here
        x = self.flatten(x)
        return x
```

[8]:
```
class Generator(nn.Module):
    def __init__(self):
        super(Generator, self).__init__()
        self.encoder = nn.Sequential(
            nn.Conv2d(3, 64, kernel_size=4, stride=2, padding=1),
            nn.ReLU(True),
            nn.Conv2d(64, 128, kernel_size=4, stride=2, padding=1),
            nn.ReLU(True),
            nn.Conv2d(128, 256, kernel_size=4, stride=2, padding=1),
            nn.ReLU(True),
        )

        self.middle = nn.Sequential(
            nn.Conv2d(256, 512, kernel_size=4, stride=2, padding=1),
            nn.ReLU(True),
```

```
            nn.Conv2d(512, 1024, kernel_size=4, stride=2, padding=1),
            nn.ReLU(True),
        )

        self.decoder = nn.Sequential(
            nn.ConvTranspose2d(1024, 512, kernel_size=4, stride=2, padding=1),
            nn.ReLU(True),
            nn.ConvTranspose2d(512, 256, kernel_size=4, stride=2, padding=1),
            nn.ReLU(True),
            nn.ConvTranspose2d(256, 128, kernel_size=4, stride=2, padding=1),
            nn.ReLU(True),
            nn.ConvTranspose2d(128, 3, kernel_size=4, stride=2, padding=1),
            nn.Tanh()  # Output in range [-1, 1]
        )

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

[9]:
```
# Training Function
def train_model(generator, discriminator, train_loader, optimizer_G,
 ↪optimizer_D, criterion, num_epochs=10, device='cuda'):
    generator.train()
    discriminator.train()

    for epoch in range(num_epochs):
        total_g_loss = 0
        total_d_loss = 0
        for images, _ in tqdm(train_loader):
            images = images.to(device)

            # Train Discriminator
            optimizer_D.zero_grad()
            real_labels = torch.ones((images.size(0), 1), device=device)
            fake_labels = torch.zeros((images.size(0), 1), device=device)

            # Discriminator Loss on Real Images
            outputs = discriminator(images)
            d_loss_real = criterion(outputs, real_labels)

            # Discriminator Loss on Fake Images
            fake_images = generator(images)
            outputs = discriminator(fake_images.detach())
            d_loss_fake = criterion(outputs, fake_labels)
```

```
            # Total Discriminator Loss
            d_loss = d_loss_real + d_loss_fake
            d_loss.backward()
            optimizer_D.step()

            # Train Generator
            optimizer_G.zero_grad()
            outputs = discriminator(fake_images)
            g_loss = criterion(outputs, real_labels)
            g_loss.backward()
            optimizer_G.step()

            total_g_loss += g_loss.item()
            total_d_loss += d_loss.item()

        print(f"Epoch [{epoch+1}/{num_epochs}], Generator Loss: {total_g_loss/
 ↪len(train_loader):.4f}, Discriminator Loss: {total_d_loss/len(train_loader):.
 ↪4f}")
```

[10]:
```
def calc_gradient_penalty(D, real_samples, fake_samples, lambda_gp=10):
    ...
    # Experiment with a lower lambda_gp
    lambda_gp = 5  # Adjust this value
    return gradient_penalty
```

[11]:
```
# Validation Function
def validate_model(generator, val_loader, device='cuda'):
    generator.eval()
    avg_psnr = 0
    avg_precision = 0
    avg_recall = 0
    avg_accuracy = 0
    avg_uciqe = 0
    num_samples = 0

    with torch.no_grad():
        for images, _ in tqdm(val_loader):
            images = images.to(device)
            outputs = generator(images)

            # Calculate PSNR
            psnr = calculate_psnr(images, outputs)
            avg_psnr += psnr

            # Calculate pixel-based metrics
            precision, recall, accuracy = calculate_pixel_metrics(images,␣
 ↪outputs)
```

```
            avg_precision += precision
            avg_recall += recall
            avg_accuracy += accuracy

            # Calculate UCIQE
            uciqe = calculate_uciqe(outputs[0])
            avg_uciqe += uciqe
            num_samples += 1

    avg_psnr /= num_samples
    avg_precision /= num_samples
    avg_recall /= num_samples
    avg_accuracy /= num_samples
    avg_uciqe /= num_samples

    print(f"Validation PSNR: {avg_psnr:.4f}, Precision: {avg_precision:.4f},␣
↪Recall: {avg_recall:.4f}, Accuracy: {avg_accuracy:.4f}, UCIQE: {avg_uciqe:.
↪4f}")
```

[12]:
```
# Initialize Models
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
generator = HHDNet().to(device)
discriminator = Discriminator().to(device)

# Optimizers
optimizer_G = optim.Adam(generator.parameters(), lr=0.0001, betas=(0.5, 0.999))
optimizer_D = optim.Adam(discriminator.parameters(), lr=0.0001, betas=(0.5, 0.
↪999))

# Loss Function
criterion = nn.BCEWithLogitsLoss()
```

[14]:
```
# Train the Model# Reuse the previous train_model function without changes
num_epochs = 30  # Adjust based on your needs
train_model(generator, discriminator, train_loader, optimizer_G, optimizer_D,␣
↪criterion, num_epochs, device)
```

```
100%|      | 471/471 [00:45<00:00, 10.28it/s]

Epoch [1/30], Generator Loss: 5.5466, Discriminator Loss: 0.0548

100%|      | 471/471 [00:44<00:00, 10.48it/s]

Epoch [2/30], Generator Loss: 5.8618, Discriminator Loss: 0.0812

100%|      | 471/471 [00:44<00:00, 10.48it/s]

Epoch [3/30], Generator Loss: 6.9920, Discriminator Loss: 0.0076

100%|      | 471/471 [00:45<00:00, 10.43it/s]
```

```
Epoch [4/30], Generator Loss: 6.4974, Discriminator Loss: 0.0737
100%|      | 471/471 [00:43<00:00, 10.85it/s]
Epoch [5/30], Generator Loss: 6.2151, Discriminator Loss: 0.1043
100%|      | 471/471 [00:44<00:00, 10.69it/s]
Epoch [6/30], Generator Loss: 6.0358, Discriminator Loss: 0.0708
100%|      | 471/471 [00:45<00:00, 10.40it/s]
Epoch [7/30], Generator Loss: 6.5207, Discriminator Loss: 0.0806
100%|      | 471/471 [00:45<00:00, 10.46it/s]
Epoch [8/30], Generator Loss: 6.3874, Discriminator Loss: 0.0620
100%|      | 471/471 [00:44<00:00, 10.63it/s]
Epoch [9/30], Generator Loss: 6.3038, Discriminator Loss: 0.0983
100%|      | 471/471 [00:44<00:00, 10.69it/s]
Epoch [10/30], Generator Loss: 6.4891, Discriminator Loss: 0.0848
100%|      | 471/471 [00:43<00:00, 10.79it/s]
Epoch [11/30], Generator Loss: 5.8032, Discriminator Loss: 0.1259
100%|      | 471/471 [00:43<00:00, 10.94it/s]
Epoch [12/30], Generator Loss: 6.1932, Discriminator Loss: 0.0832
100%|      | 471/471 [00:43<00:00, 10.75it/s]
Epoch [13/30], Generator Loss: 5.7267, Discriminator Loss: 0.1662
100%|      | 471/471 [00:44<00:00, 10.50it/s]
Epoch [14/30], Generator Loss: 5.2556, Discriminator Loss: 0.1975
100%|      | 471/471 [00:44<00:00, 10.60it/s]
Epoch [15/30], Generator Loss: 5.1214, Discriminator Loss: 0.1614
100%|      | 471/471 [00:44<00:00, 10.69it/s]
Epoch [16/30], Generator Loss: 4.4290, Discriminator Loss: 0.3107
100%|      | 471/471 [00:44<00:00, 10.62it/s]
Epoch [17/30], Generator Loss: 3.7789, Discriminator Loss: 0.3659
100%|      | 471/471 [00:43<00:00, 10.85it/s]
Epoch [18/30], Generator Loss: 3.4550, Discriminator Loss: 0.4290
100%|      | 471/471 [00:42<00:00, 11.05it/s]
Epoch [19/30], Generator Loss: 3.1589, Discriminator Loss: 0.4940
100%|      | 471/471 [00:43<00:00, 10.73it/s]
```

```
Epoch [20/30], Generator Loss: 2.8948, Discriminator Loss: 0.5422
100%|          | 471/471 [00:45<00:00, 10.38it/s]
Epoch [21/30], Generator Loss: 2.1157, Discriminator Loss: 0.7782
100%|          | 471/471 [00:45<00:00, 10.36it/s]
Epoch [22/30], Generator Loss: 1.6909, Discriminator Loss: 0.8988
100%|          | 471/471 [00:44<00:00, 10.57it/s]
Epoch [23/30], Generator Loss: 1.4601, Discriminator Loss: 1.0153
100%|          | 471/471 [00:44<00:00, 10.56it/s]
Epoch [24/30], Generator Loss: 1.3447, Discriminator Loss: 1.0350
100%|          | 471/471 [00:44<00:00, 10.69it/s]
Epoch [25/30], Generator Loss: 1.2364, Discriminator Loss: 1.0897
100%|          | 471/471 [00:43<00:00, 10.83it/s]
Epoch [26/30], Generator Loss: 1.1036, Discriminator Loss: 1.1678
100%|          | 471/471 [00:43<00:00, 10.93it/s]
Epoch [27/30], Generator Loss: 1.0223, Discriminator Loss: 1.1950
100%|          | 471/471 [00:43<00:00, 10.77it/s]
Epoch [28/30], Generator Loss: 0.9187, Discriminator Loss: 1.2799
100%|          | 471/471 [00:44<00:00, 10.66it/s]
Epoch [29/30], Generator Loss: 0.9058, Discriminator Loss: 1.2630
100%|          | 471/471 [00:43<00:00, 10.72it/s]
Epoch [30/30], Generator Loss: 0.8623, Discriminator Loss: 1.3020
```

[15]:
```python
def unnormalize(image):
    # Ensure mean and std are on the same device as the image
    mean = torch.tensor([0.5, 0.5, 0.5], device=image.device).view(3, 1, 1)
    std = torch.tensor([0.5, 0.5, 0.5], device=image.device).view(3, 1, 1)
    return image * std + mean
```

[16]:
```python
# Validate the Model
validate_model(generator, val_loader, device)
```

```
100%|          | 87/87 [04:40<00:00,  3.23s/it]
Validation PSNR: 35.3715, Precision: 0.9991, Recall: 0.9503, Accuracy: 0.9916,
UCIQE: 0.2389
```

[ ]:
```
# Save the trained models
torch.save(generator.state_dict(), "/content/drive/My Drive/generator.pth")
torch.save(discriminator.state_dict(), "/content/drive/My Drive/discriminator.
↪pth")

# To load the models later:
# generator.load_state_dict(torch.load("/content/drive/My Drive/generator.pth"))
# discriminator.load_state_dict(torch.load("/content/drive/My Drive/
↪discriminator.pth"))
```

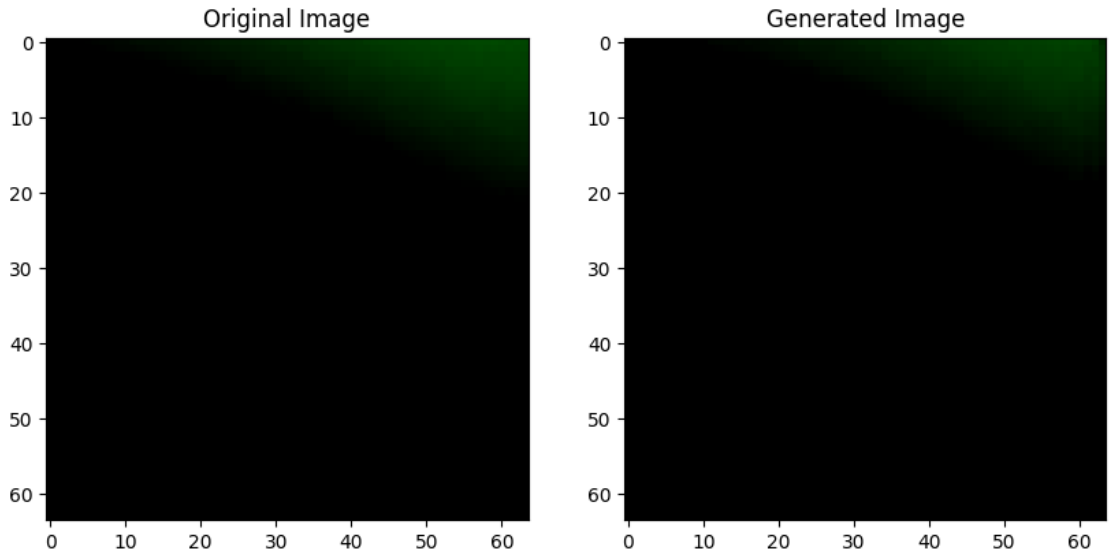
[17]:
```
# Unnormalize helper function
def unnormalize(image):
    # Assuming the mean and std from the data normalization transform
    mean = torch.tensor([0.5, 0.5, 0.5]).view(3, 1, 1).to(image.device) # Move␣
↪mean to image's device
    std = torch.tensor([0.5, 0.5, 0.5]).view(3, 1, 1).to(image.device)  # Move␣
↪std to image's device
    return image * std + mean
```

[18]:
```
import torch.nn.functional as F
from sklearn.metrics import precision_score, recall_score, accuracy_score
import matplotlib.pyplot as plt
import numpy as np
from skimage import color

# Testing function with PSNR, Pixel-wise metrics, UCIQE, and UIQM
def test_model(generator, test_loader, device='cuda'):
    generator.eval()
    total_psnr = 0
    total_precision = 0
    total_recall = 0
    total_accuracy = 0
    total_uciqe = 0
    total_uiqm = 0
    num_images = 0

    with torch.no_grad():
        for images, _ in test_loader:
            images = images.to(device)
            outputs = generator(images)

            for i in range(len(images)):
                original_img = images[i]
                generated_img = outputs[i]

                # Calculate metrics
                psnr_value = calculate_psnr(original_img, generated_img)
```

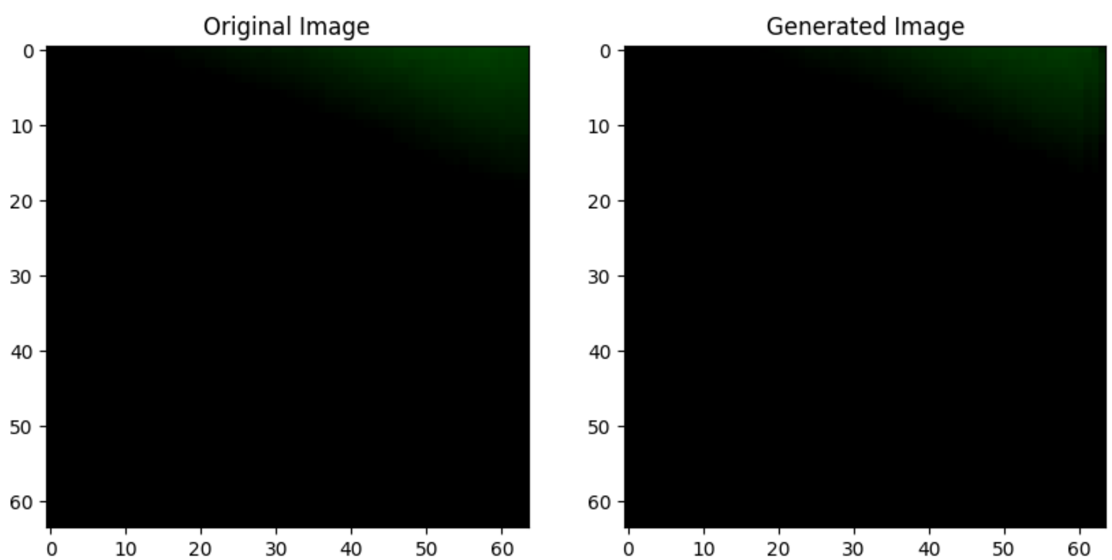
```
                precision, recall, accuracy =␣
↪calculate_pixel_metrics(original_img, generated_img)
                ucige = calculate_uciqe(generated_img)
                uiqm = calculate_uiqm(generated_img)

                # Accumulate metrics for averaging
                total_psnr += psnr_value
                total_precision += precision
                total_recall += recall
                total_accuracy += accuracy
                total_uciqe += uciqe
                total_uiqm += uiqm
                num_images += 1

                # Display a few examples
                if i < 3:  # Display up to 3 images
                    plt.figure(figsize=(10, 5))

                    # Display Original Image
                    plt.subplot(1, 2, 1)
                    plt.imshow(original_img.cpu().permute(1, 2, 0).numpy())
                    plt.title("Original Image")

                    # Display Generated Image
                    plt.subplot(1, 2, 2)
                    plt.imshow(generated_img.cpu().permute(1, 2, 0).numpy())
                    plt.title("Generated Image")

                    plt.show()

        # Calculate average metrics
        avg_psnr = total_psnr / num_images
        avg_precision = total_precision / num_images
        avg_recall = total_recall / num_images
        avg_accuracy = total_accuracy / num_images
        avg_uciqe = total_uciqe / num_images
        avg_uiqm = total_uiqm / num_images

        # Print average metrics
        print(f"Average PSNR: {avg_psnr:.2f}")
        print(f"Average Precision: {avg_precision:.4f}")
        print(f"Average Recall: {avg_recall:.4f}")
        print(f"Average Accuracy: {avg_accuracy:.4f}")
        print(f"Average UCIQE: {avg_uciqe:.4f}")
        print(f"Average UIQM: {avg_uiqm:.4f}")

# Run the test model
```

```
test_model(generator, test_loader, device)
```

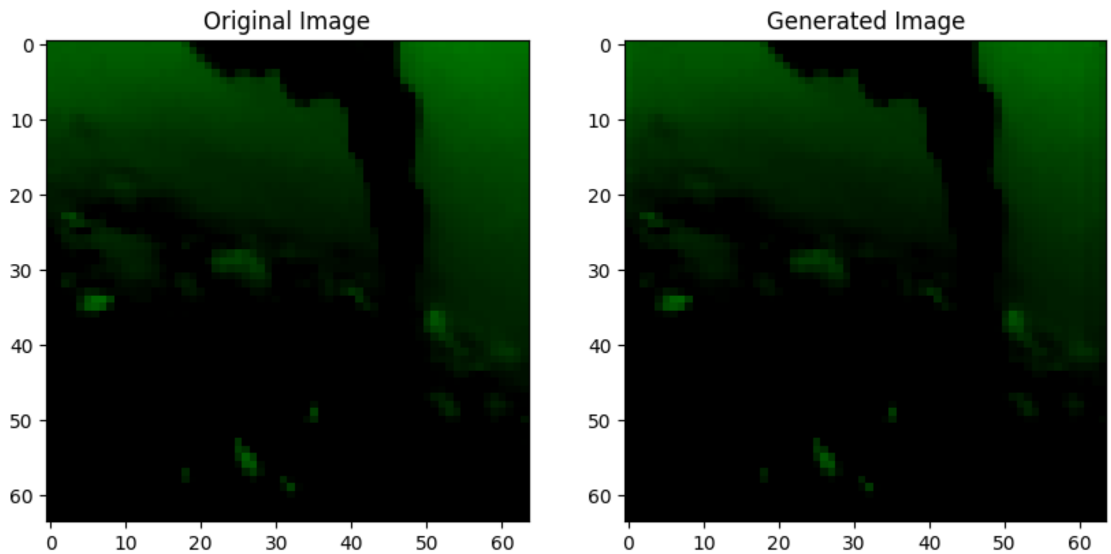
```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```



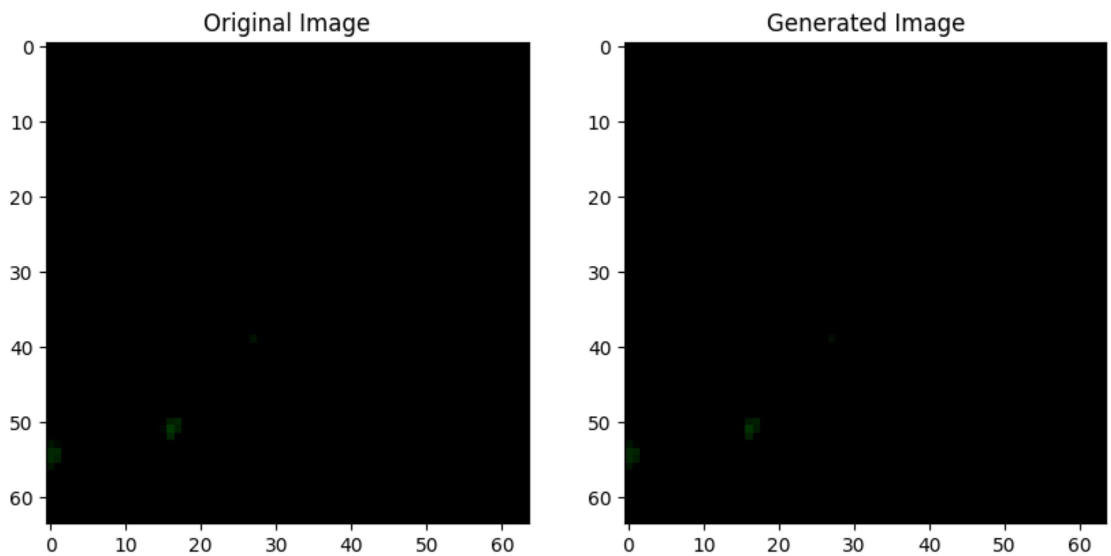

```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```



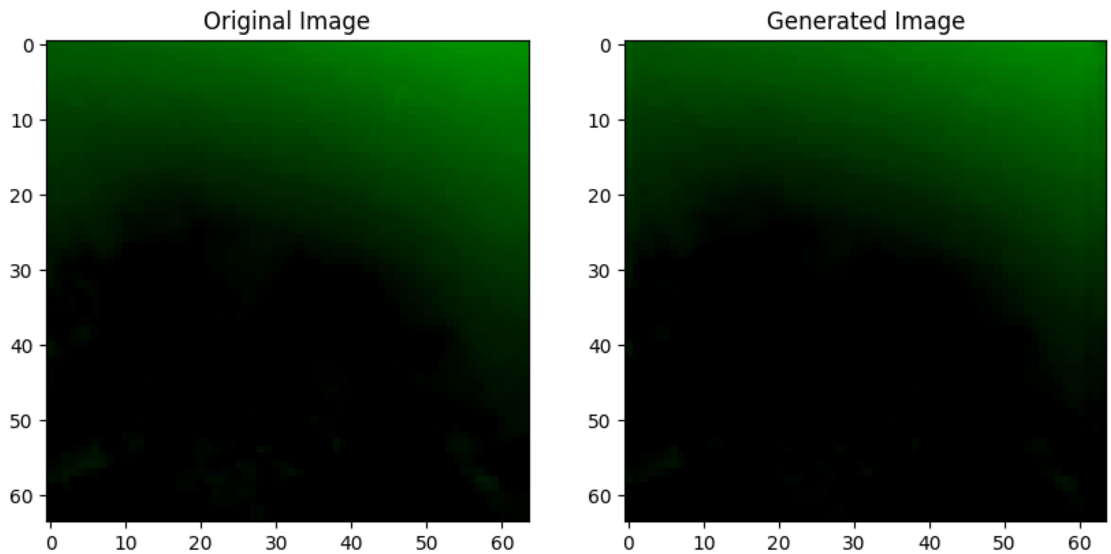
```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```



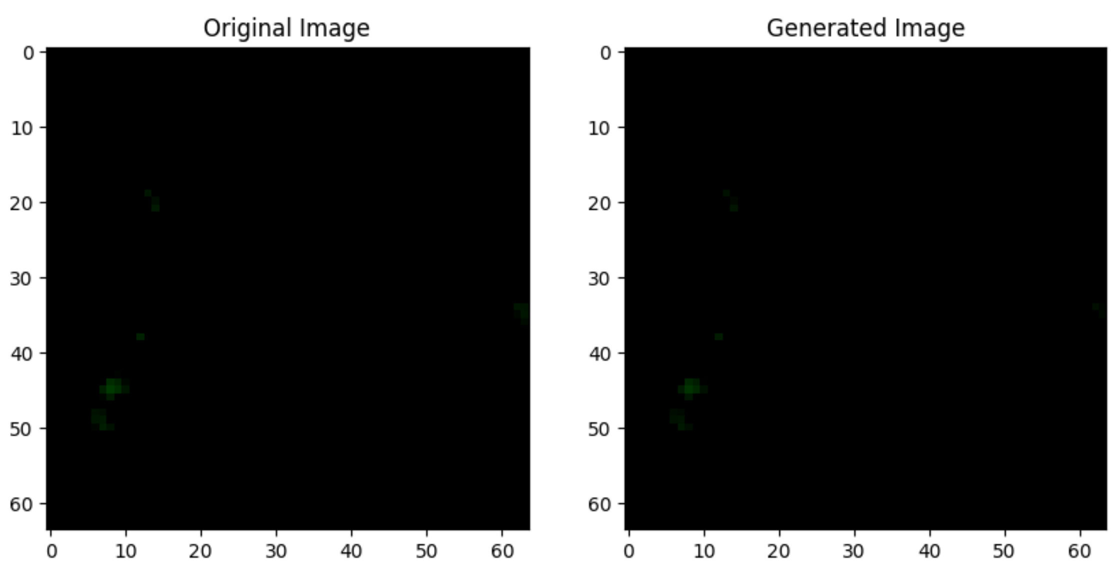

```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```



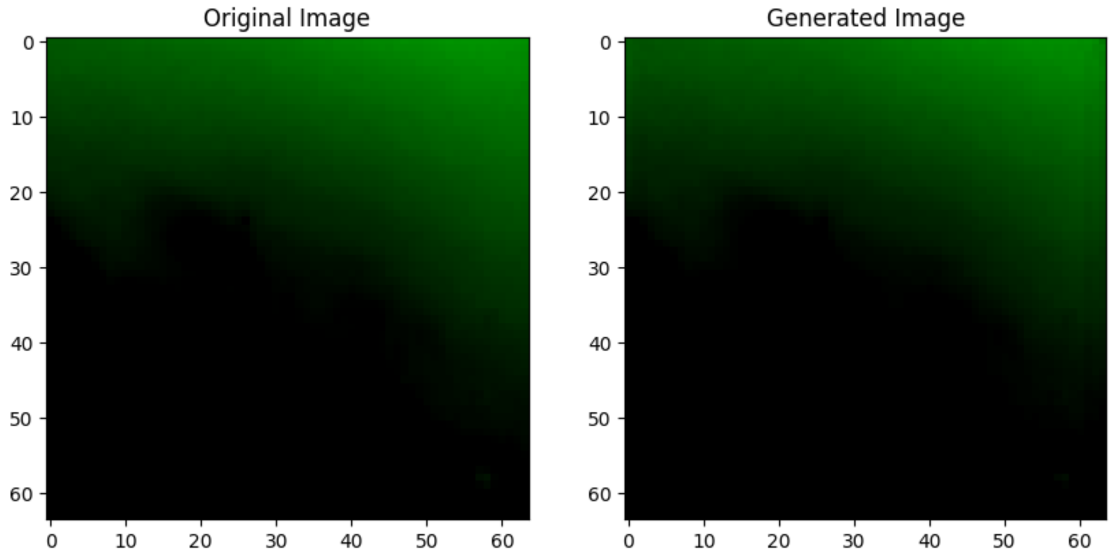
```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```



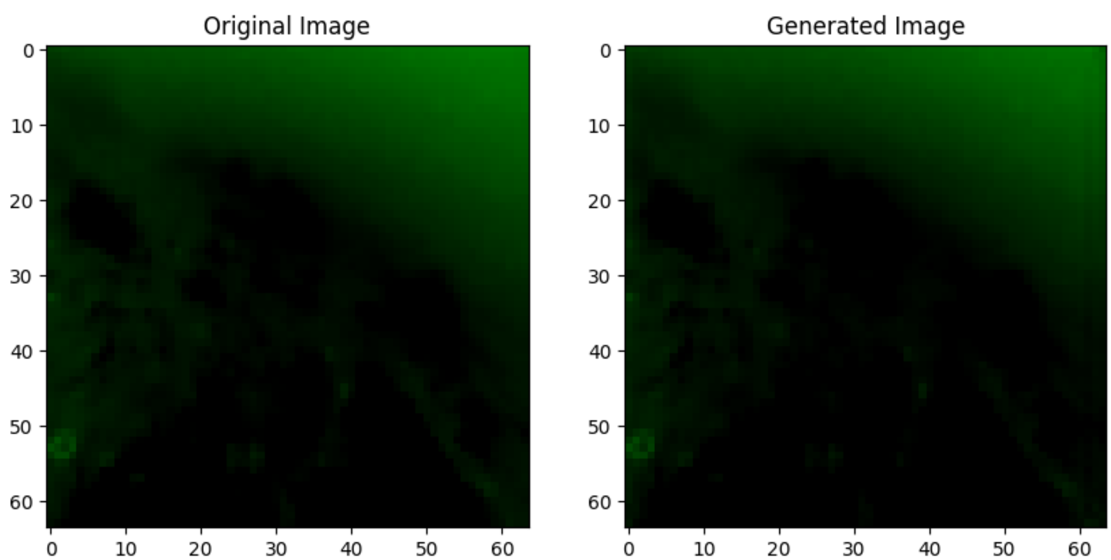

```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```



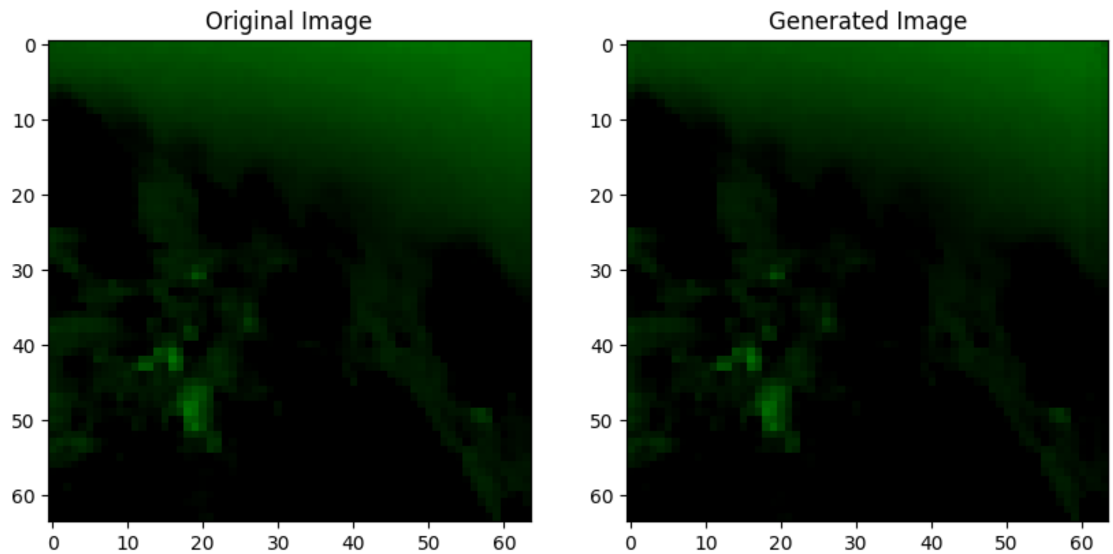
```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```



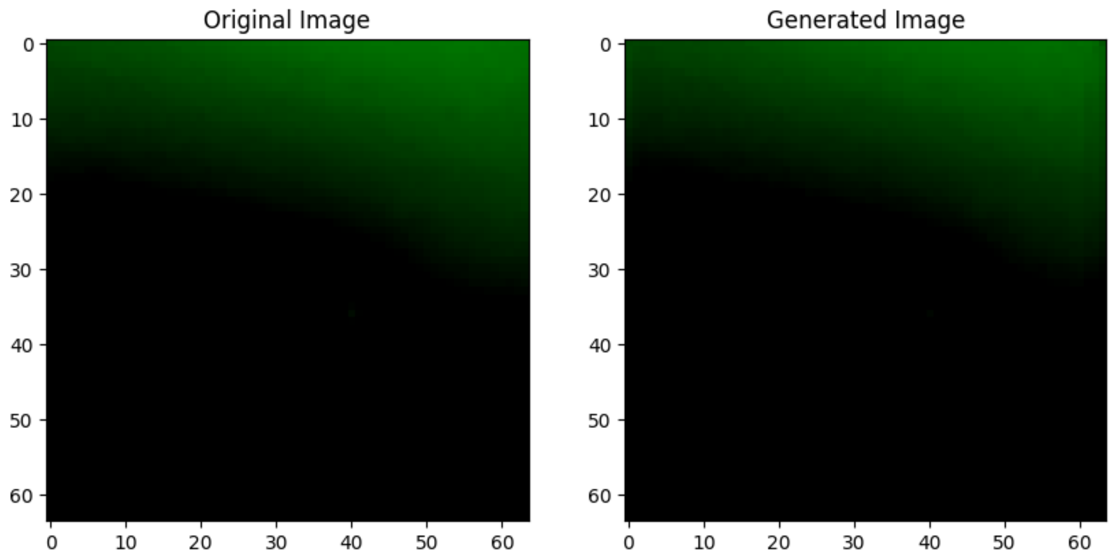

```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```



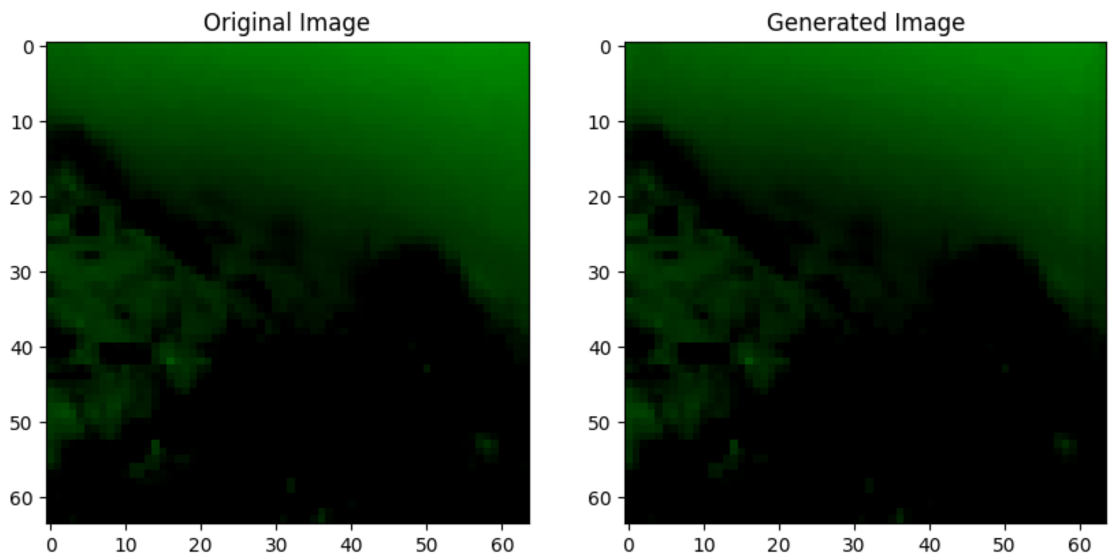
```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```



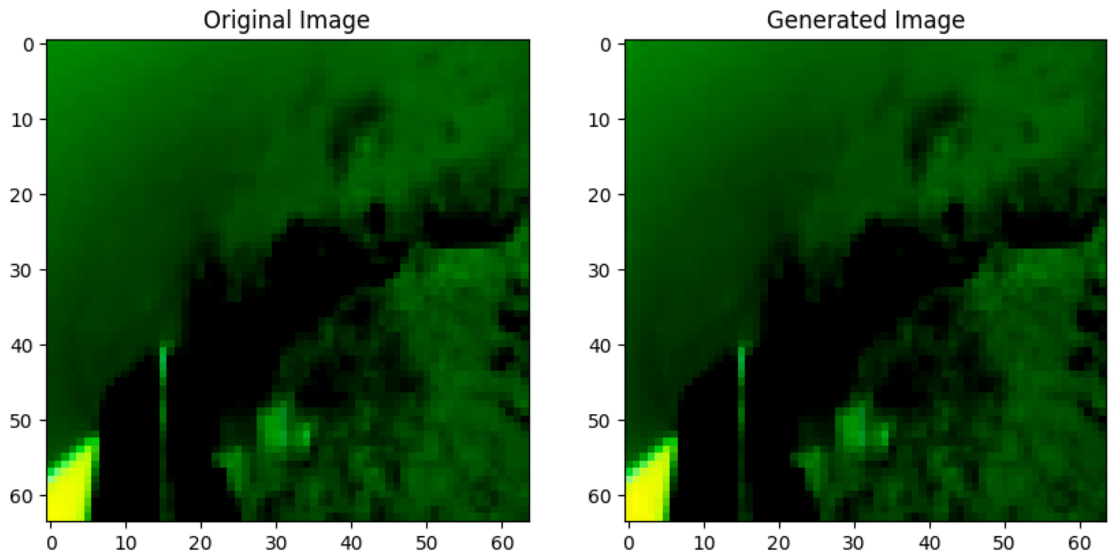

```
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1531:
UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 due to no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1531:
UndefinedMetricWarning: Recall is ill-defined and being set to 0.0 due to no
true samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```



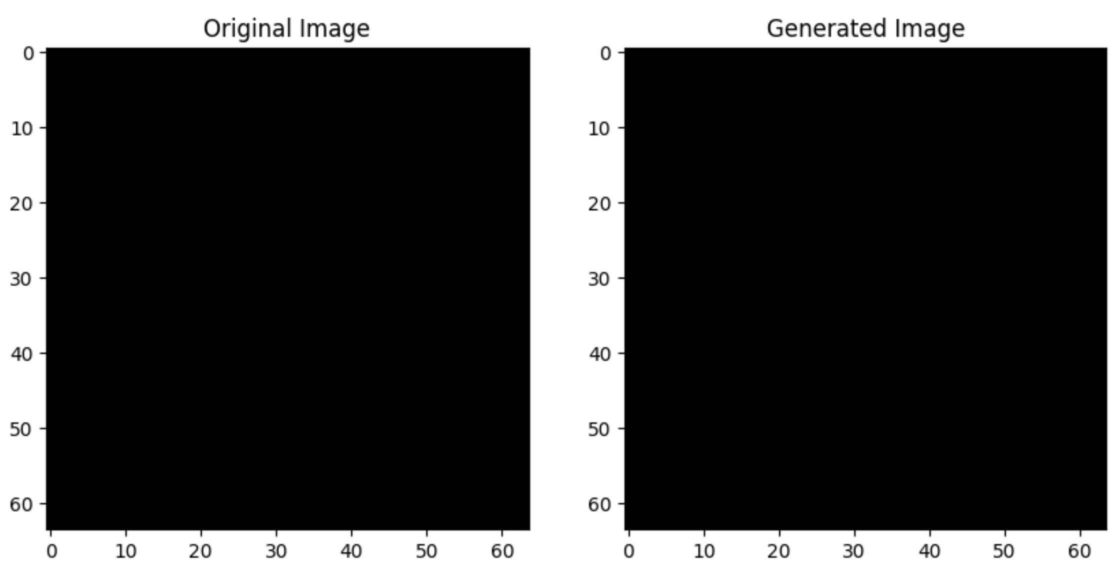

```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```



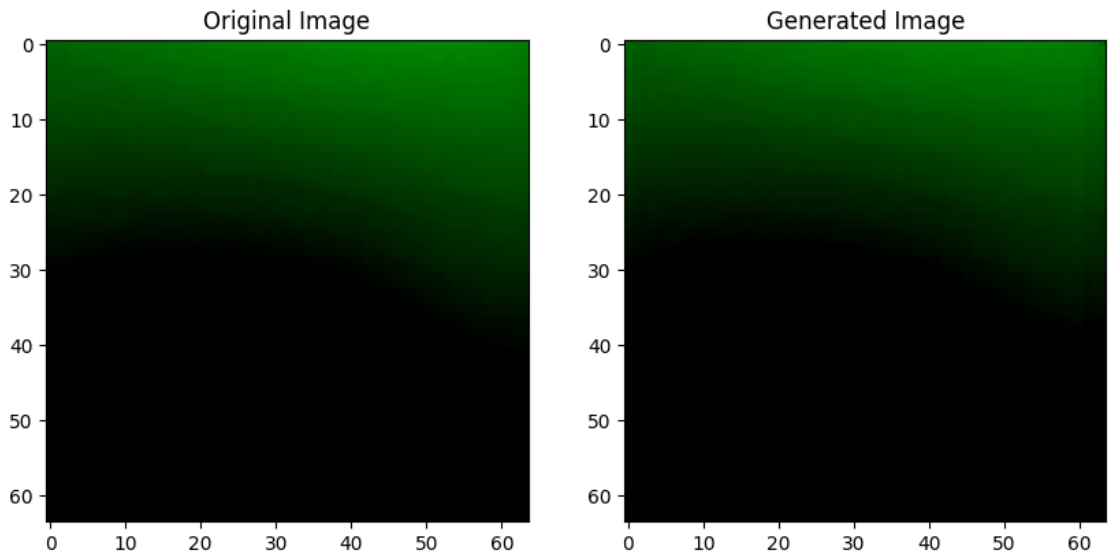

```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```



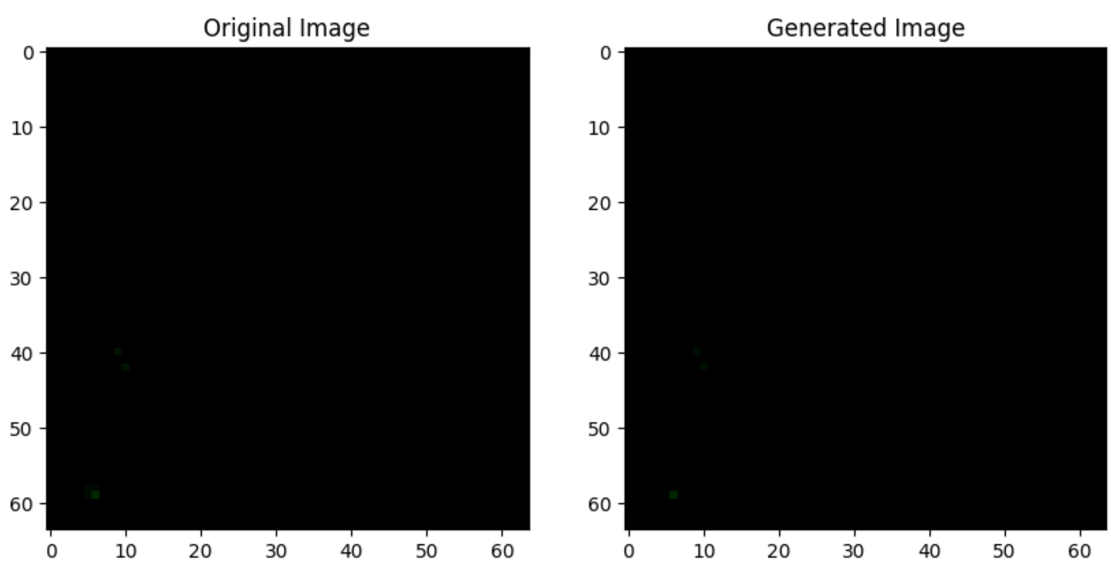

```
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1531:
UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 due to no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1531:
UndefinedMetricWarning: Recall is ill-defined and being set to 0.0 due to no
true samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```



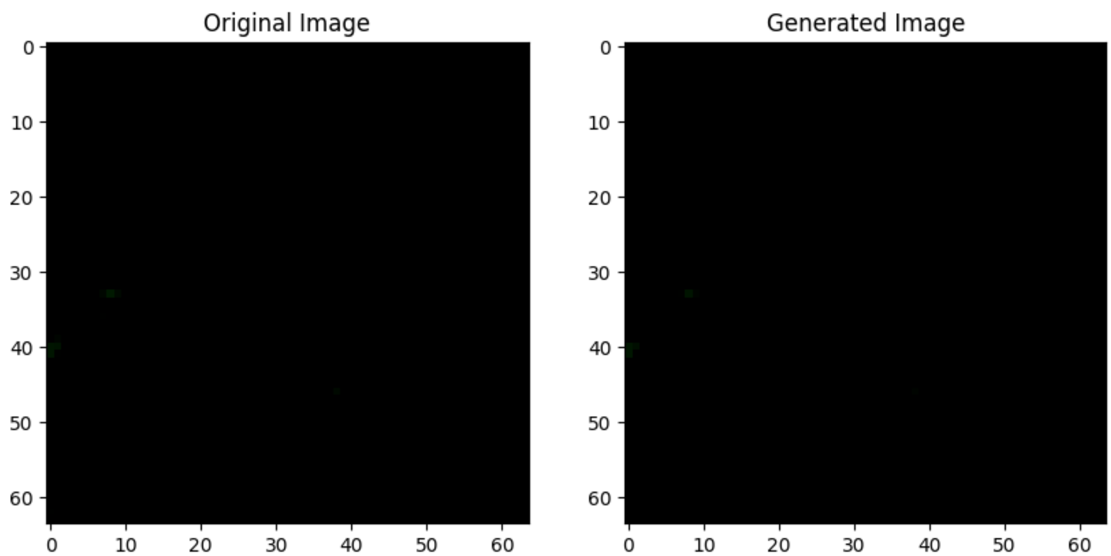
```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```



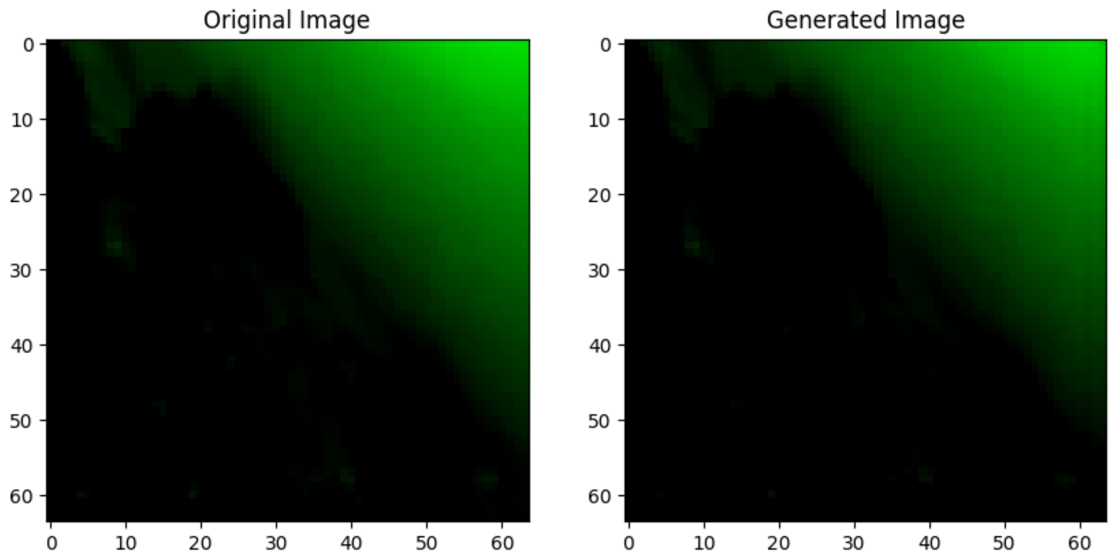

```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```



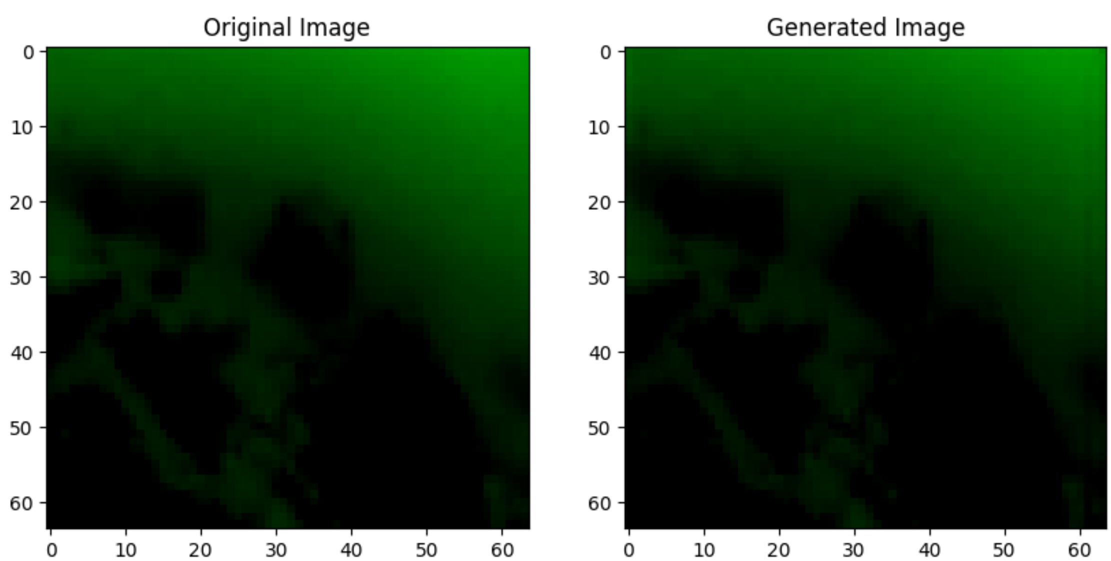
```
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1531:
UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 due to no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1531:
UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 due to no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1531:
UndefinedMetricWarning: Recall is ill-defined and being set to 0.0 due to no
true samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```




```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```

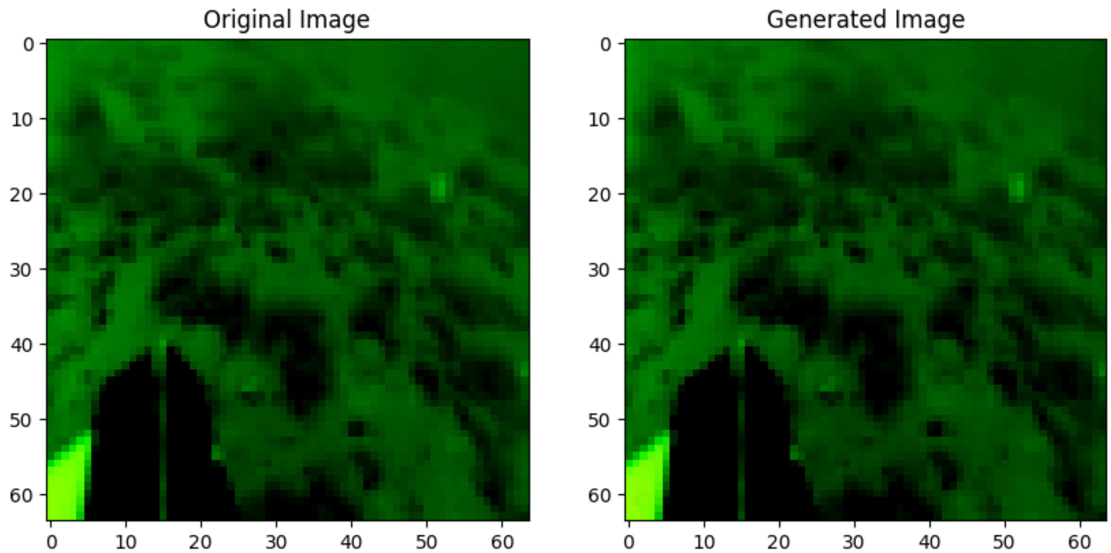



```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```

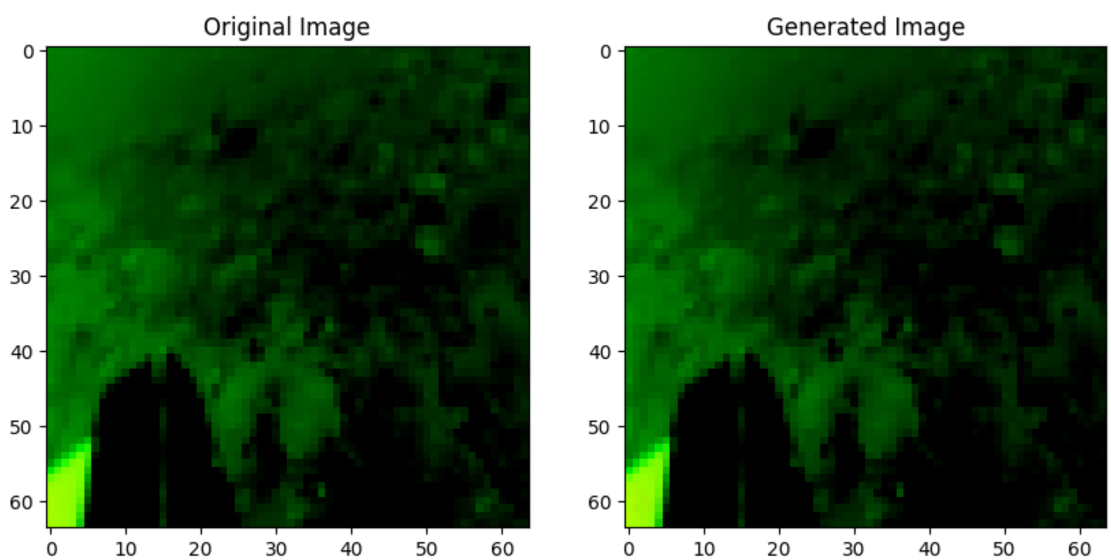



```
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1531:
UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 due to no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1531:
```

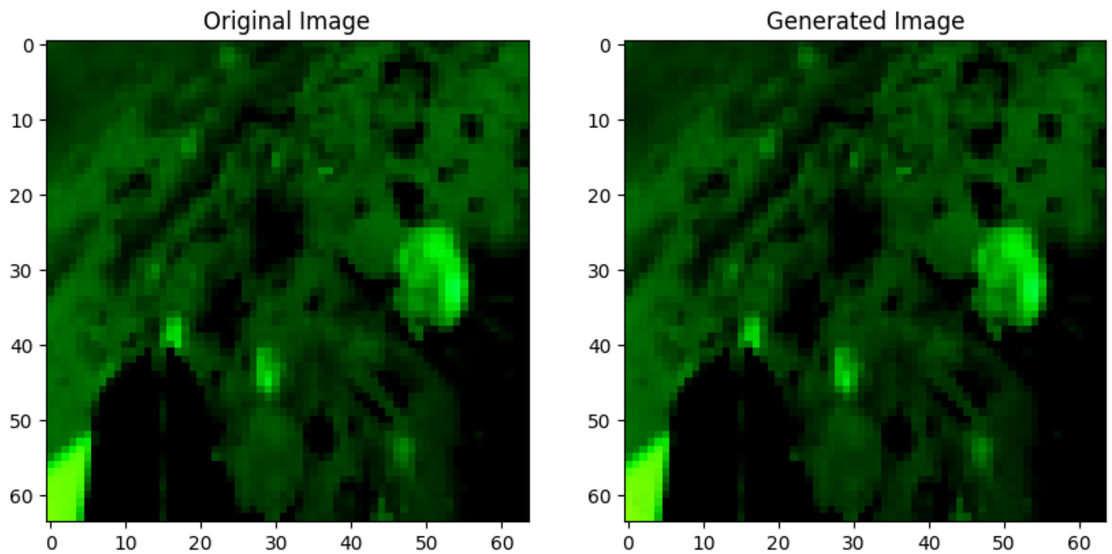
```
UndefinedMetricWarning: Recall is ill-defined and being set to 0.0 due to no
true samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```



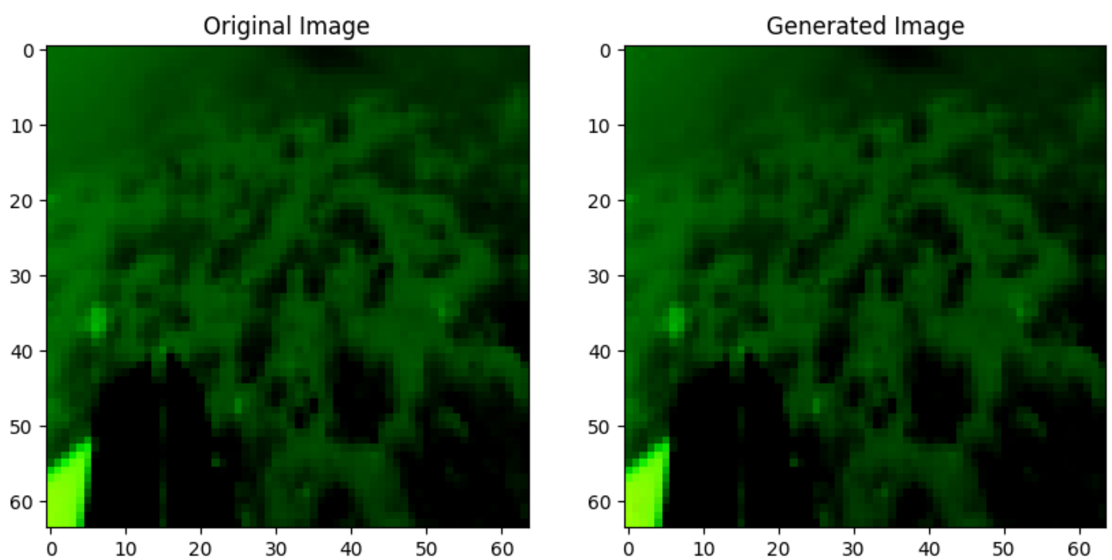

```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```



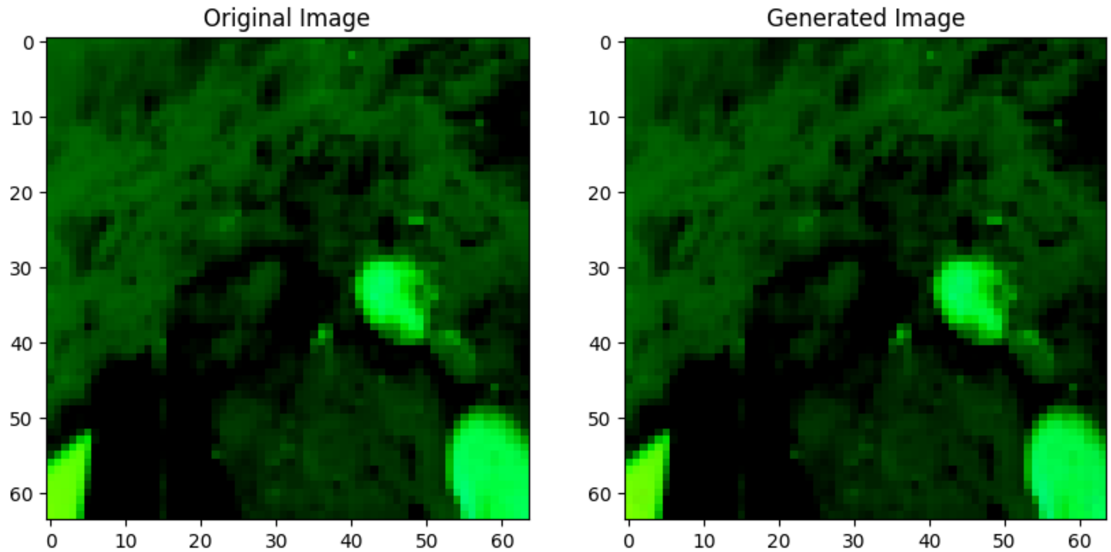
```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```



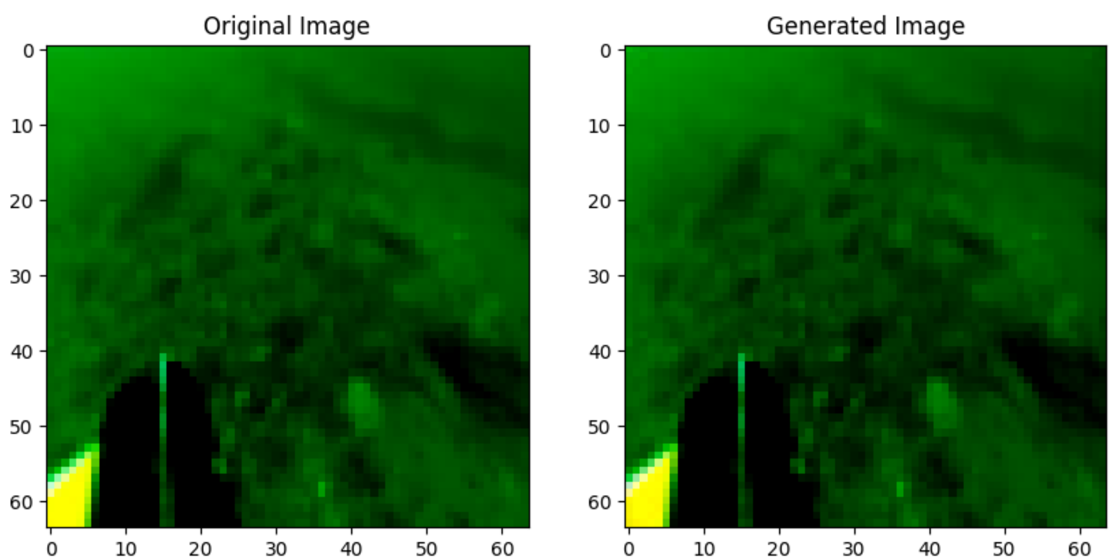

```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```



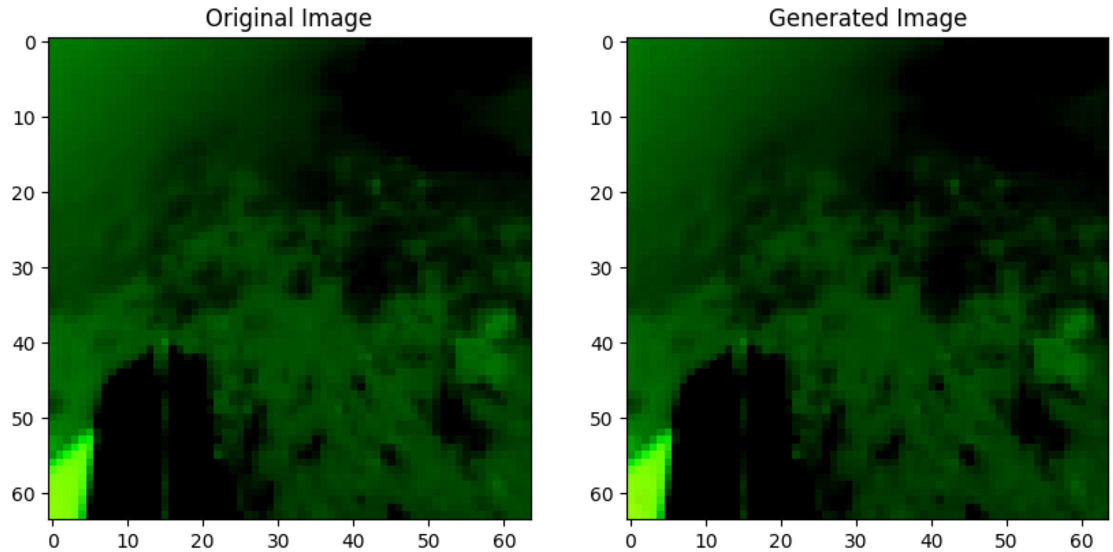
```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```



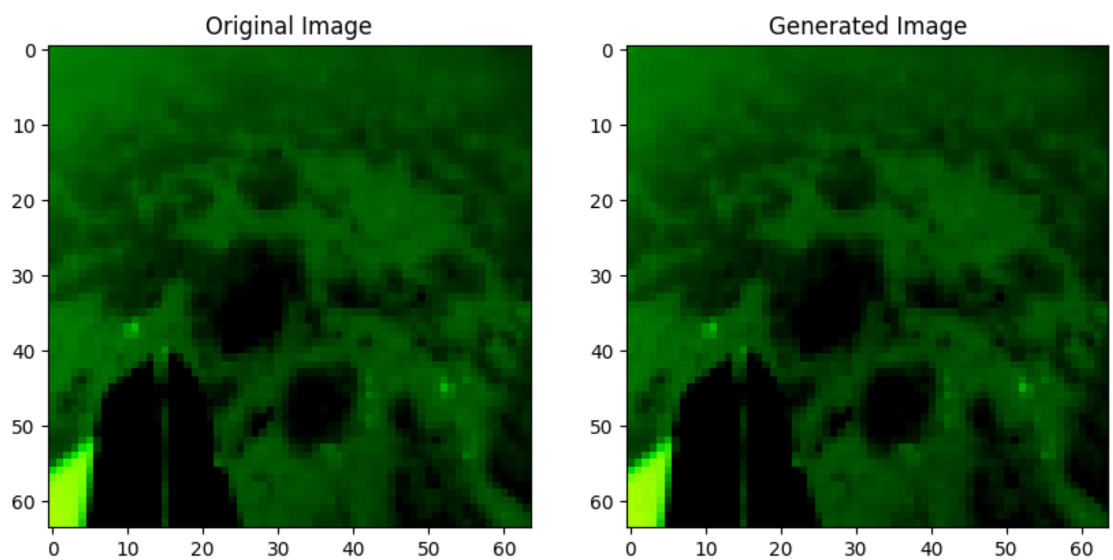

```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```



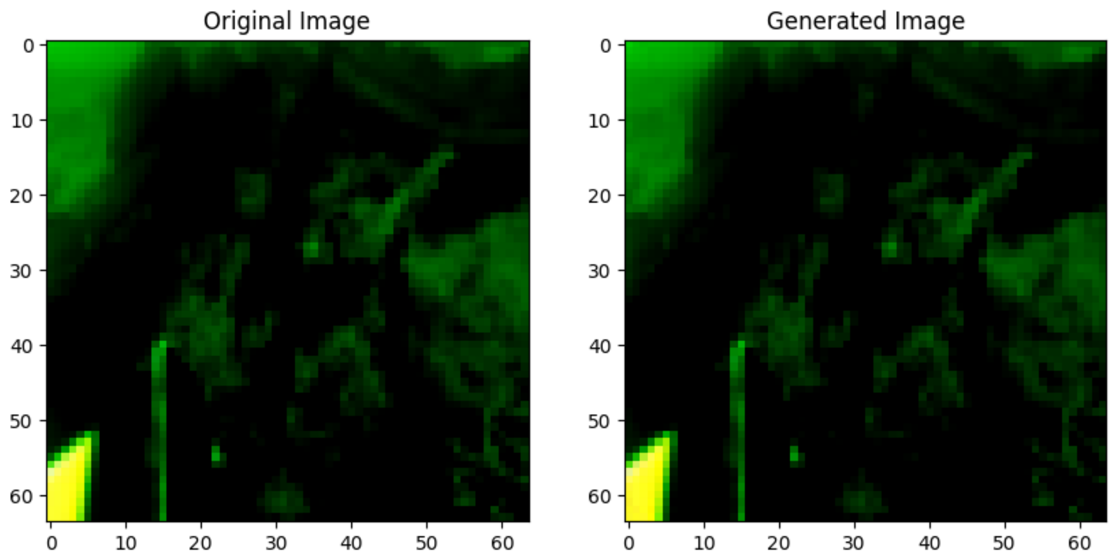
```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```



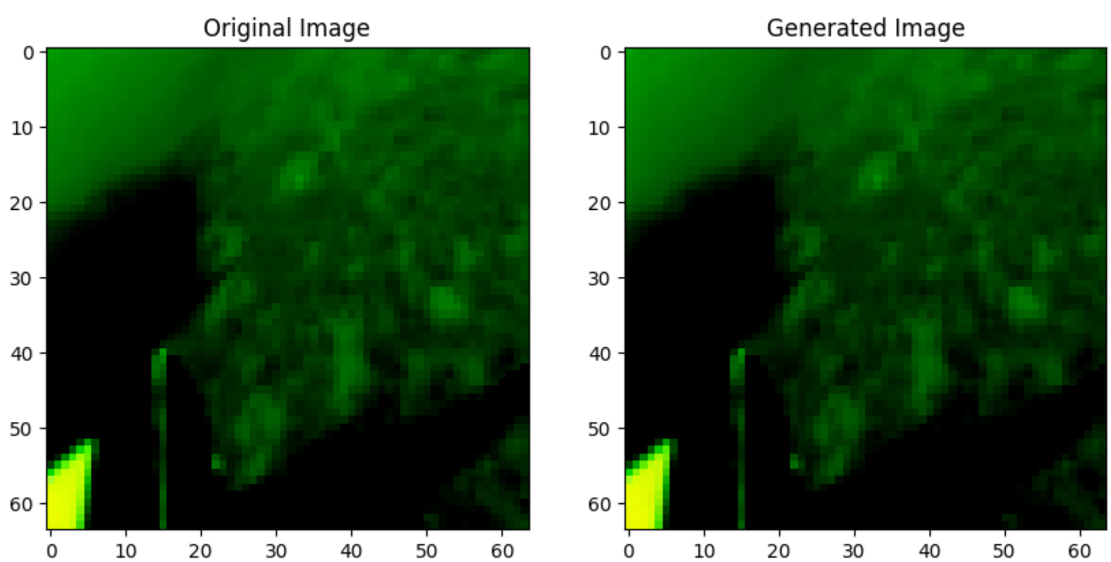

```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```



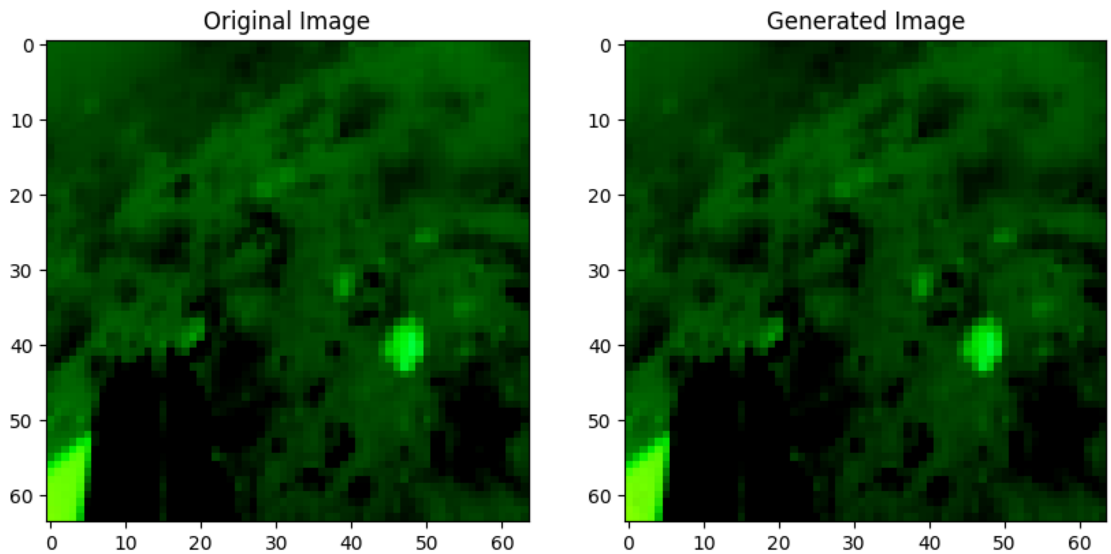
```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```



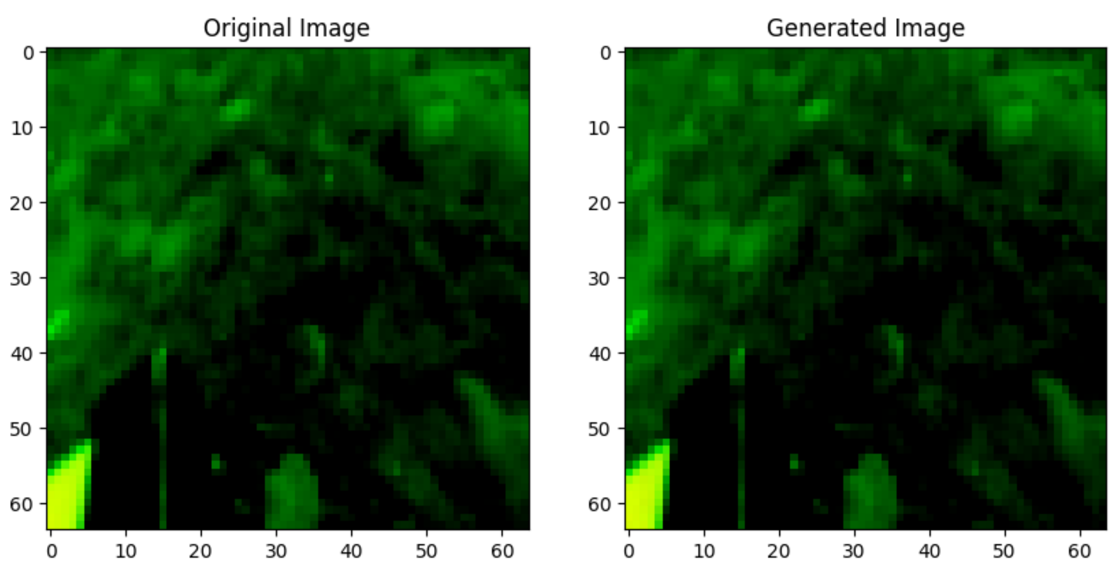

```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```



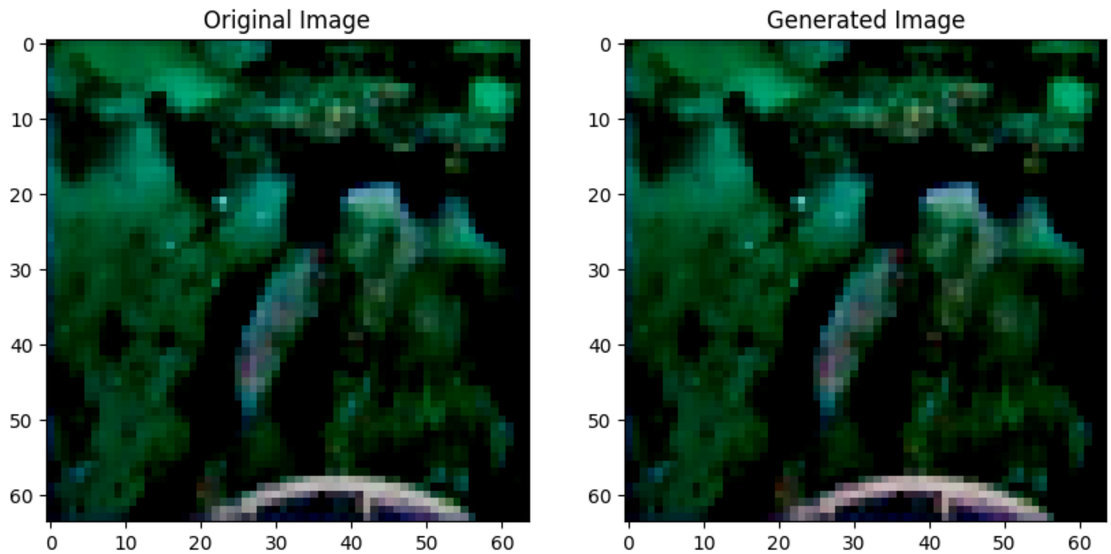
```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```



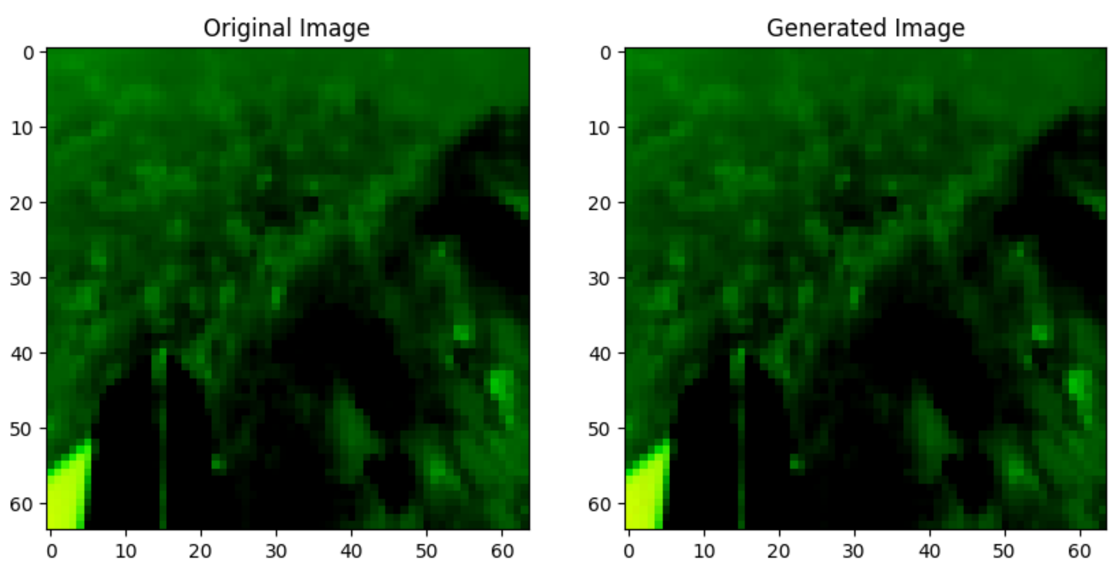

```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```



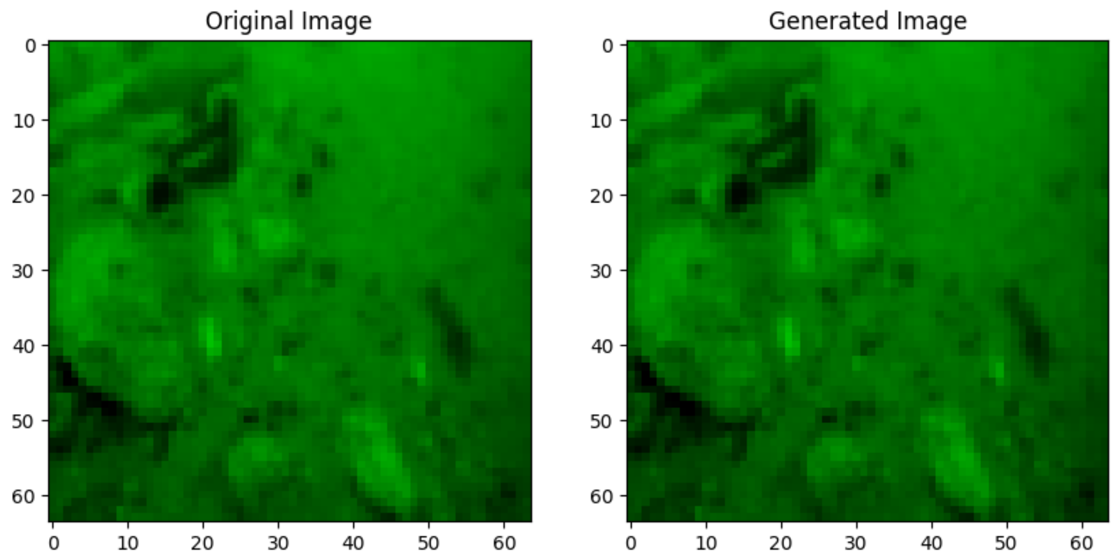
```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```



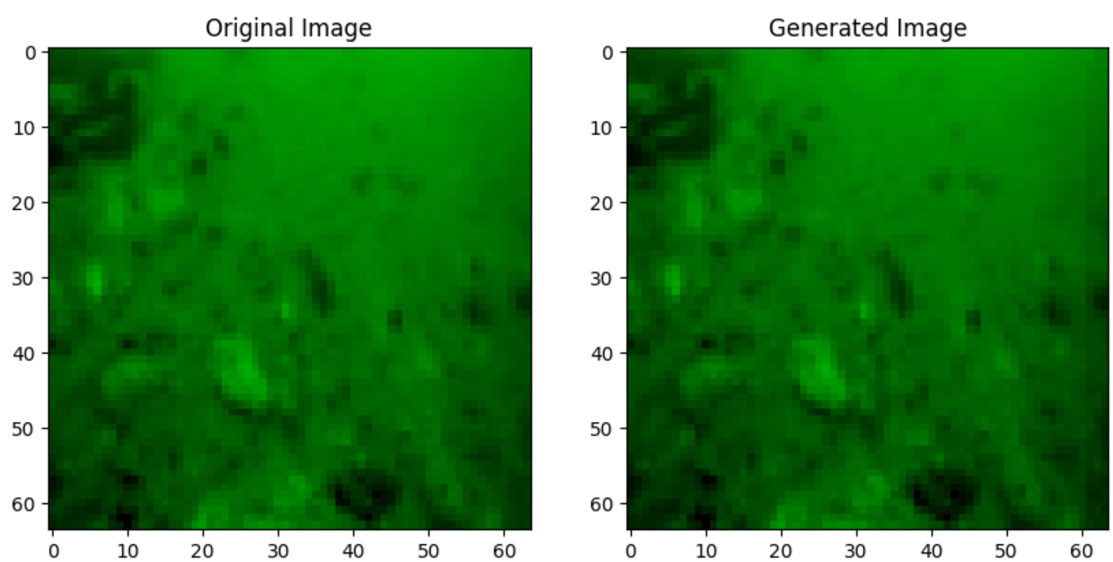

```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```



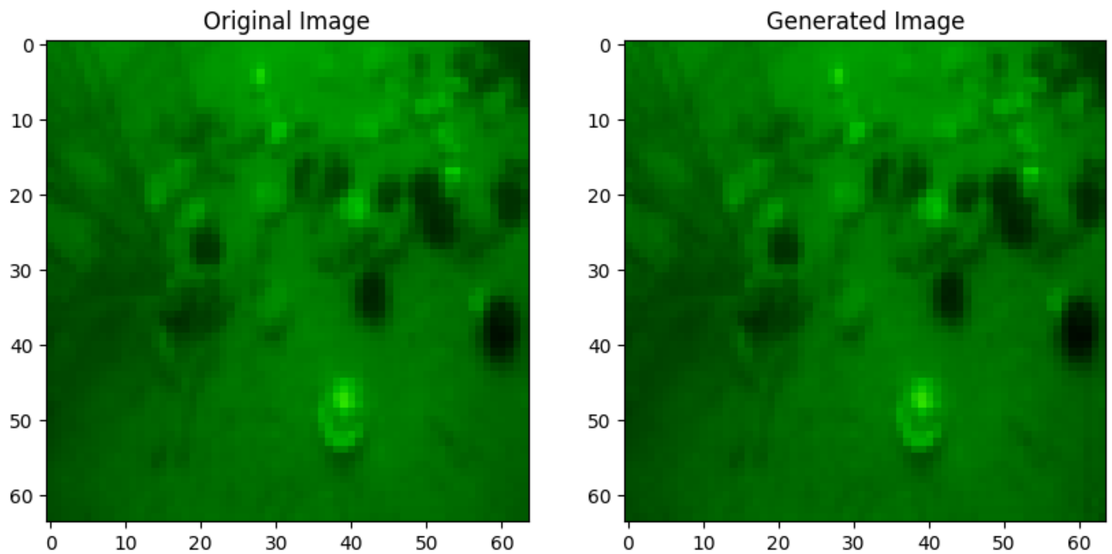
```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```



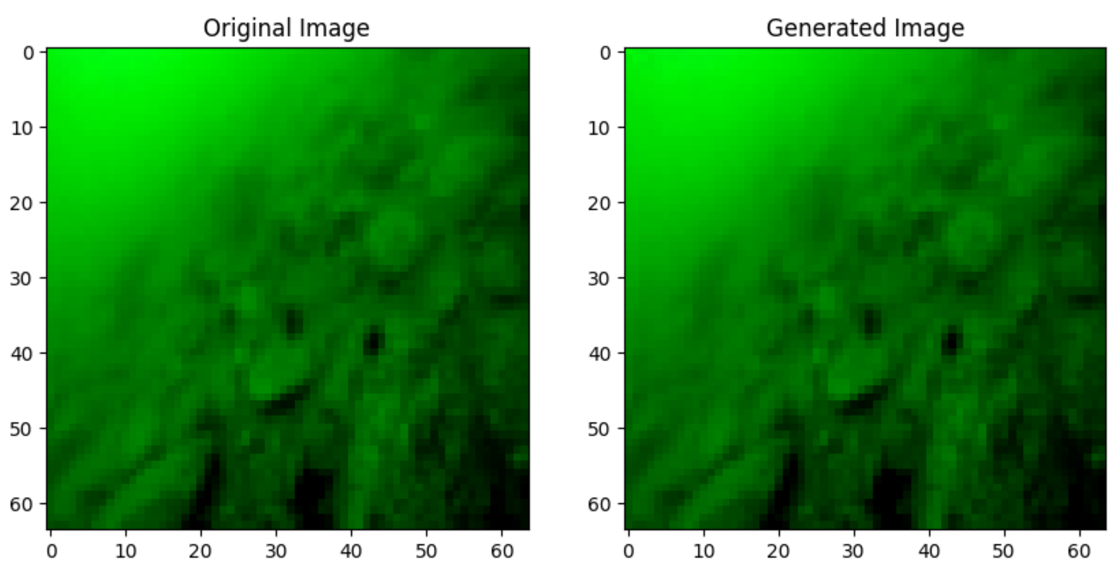

```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```



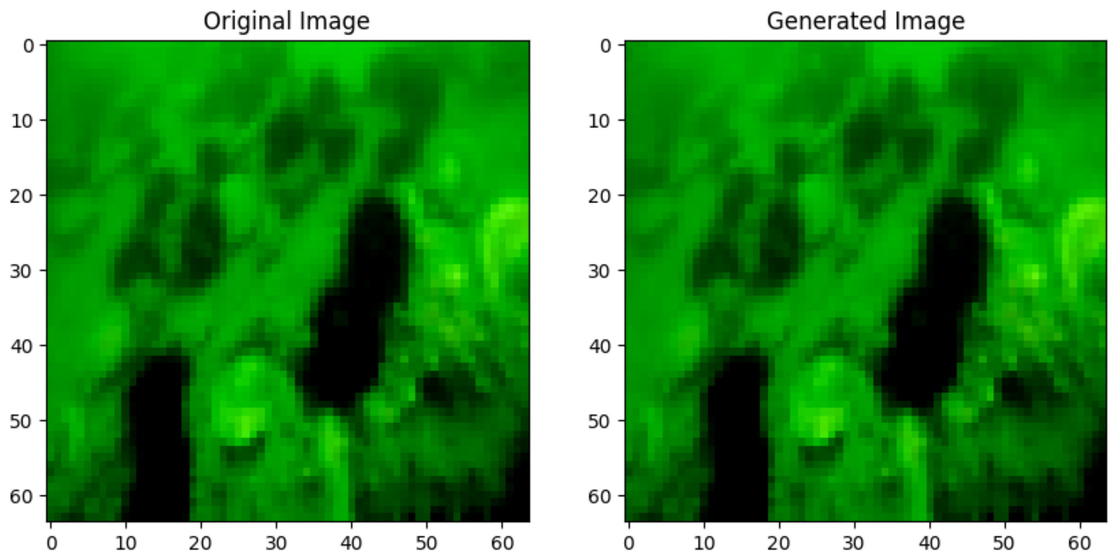
```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```



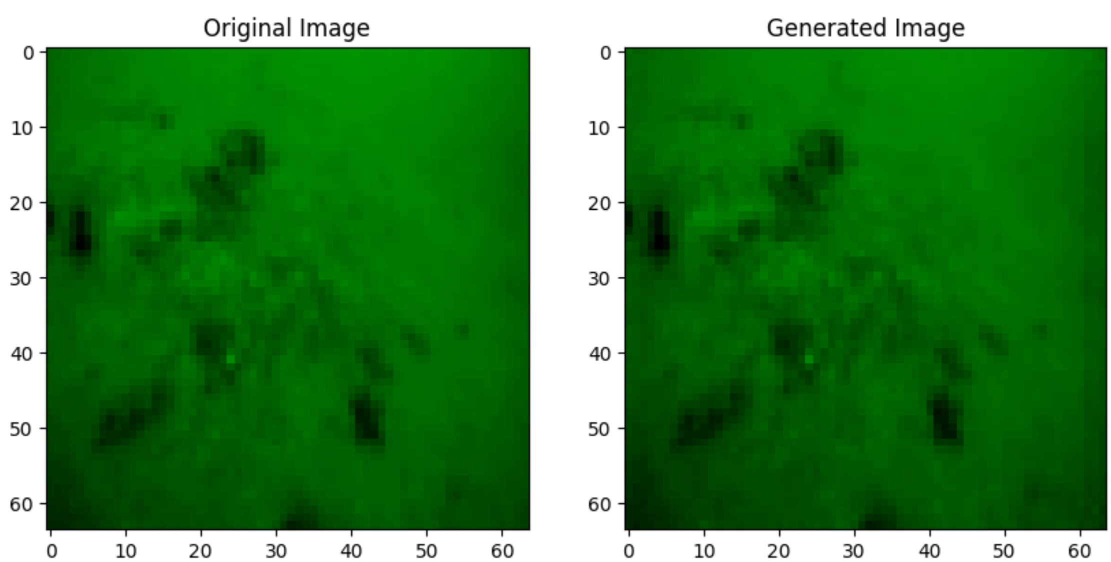

```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```



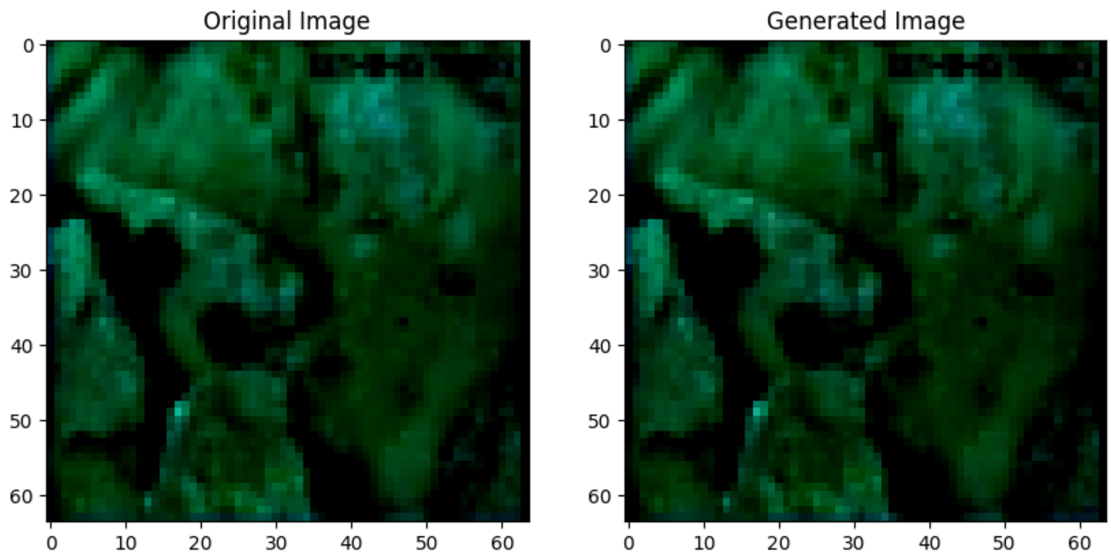
```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```



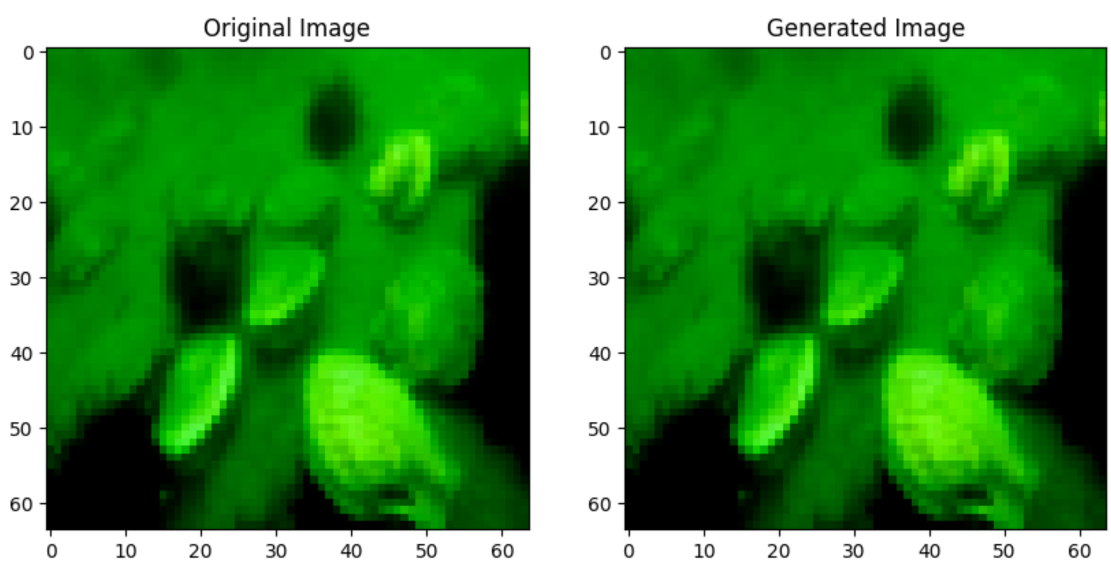

```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```



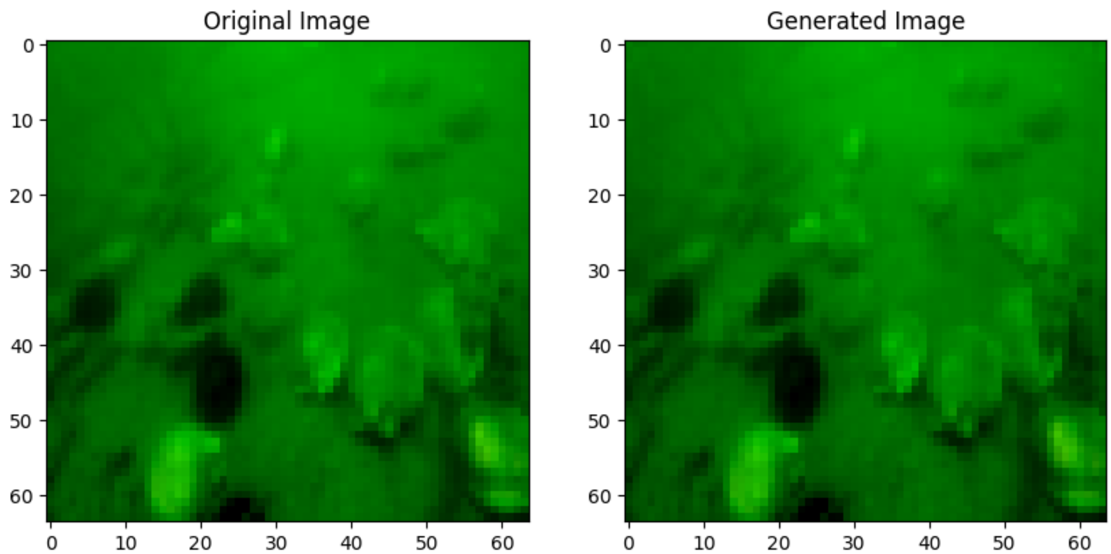
```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```



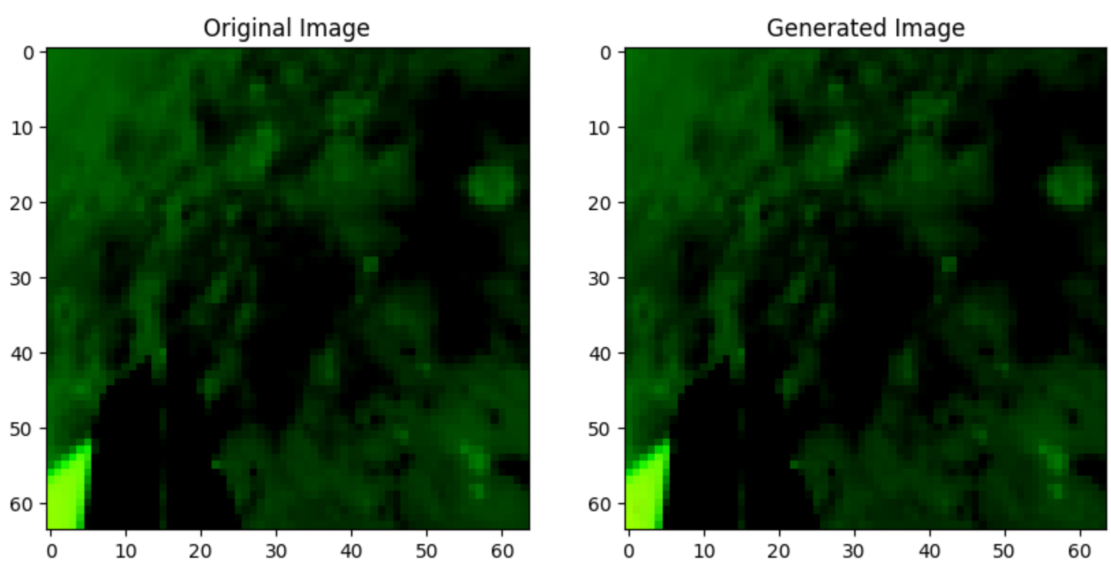

```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```



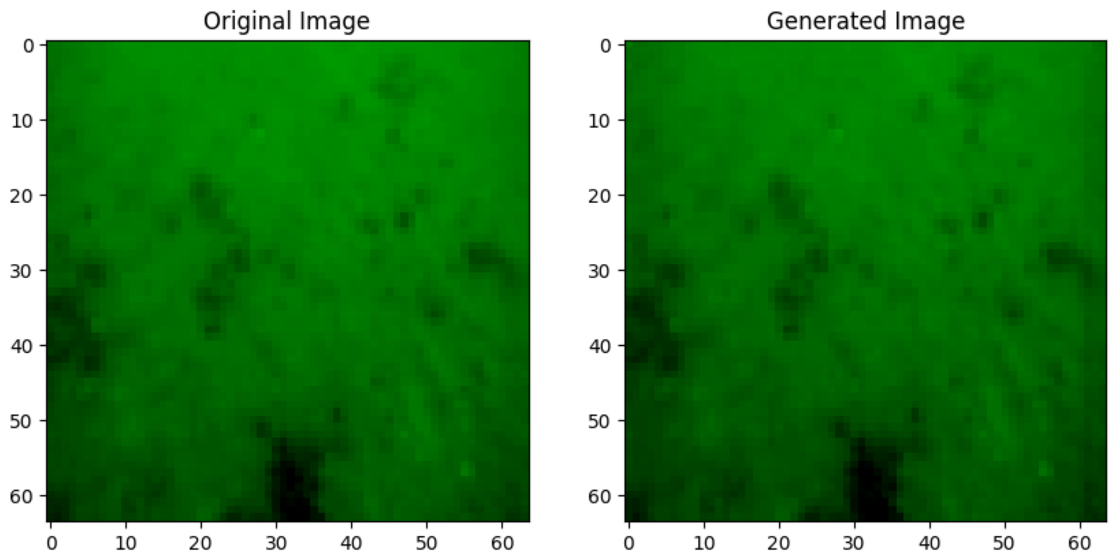
```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```



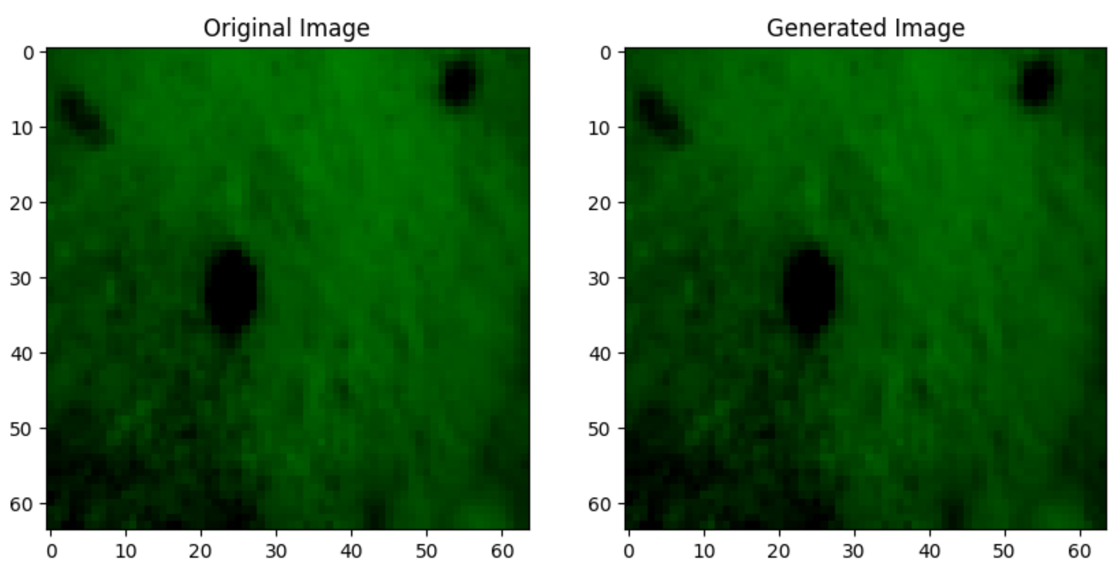

```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```



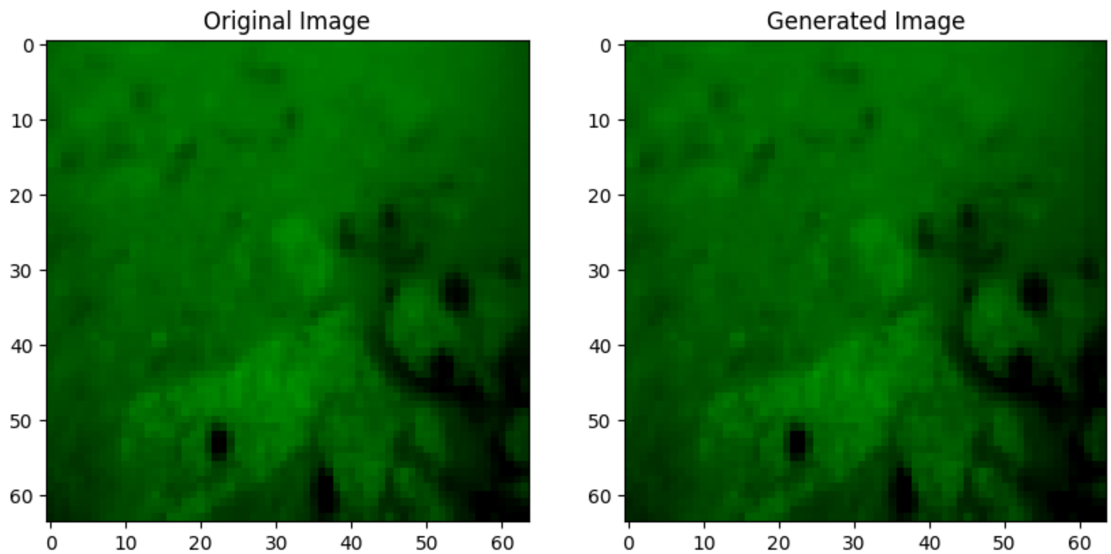
```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```



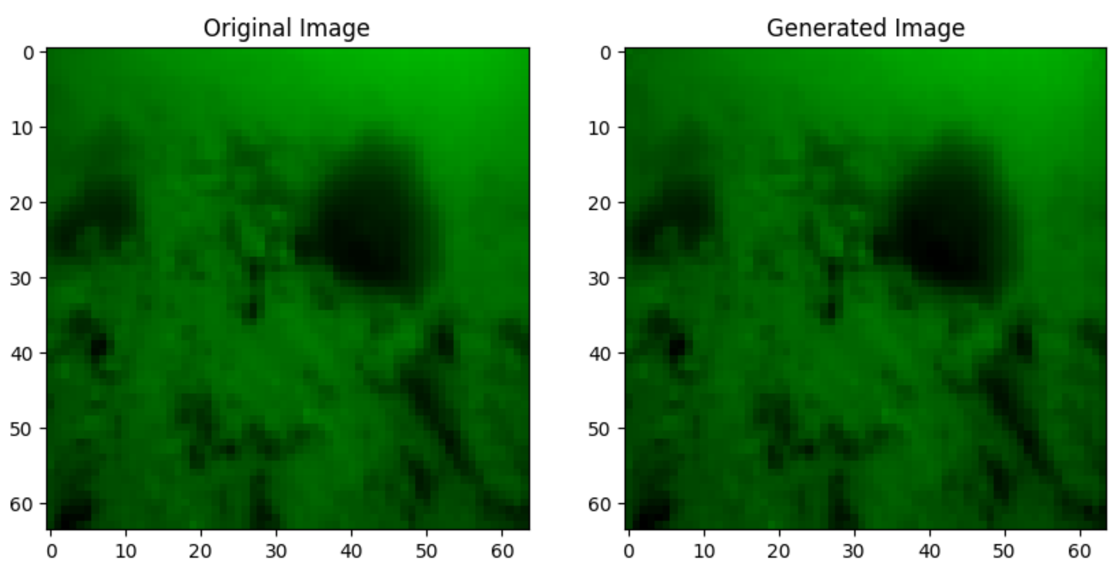

```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```



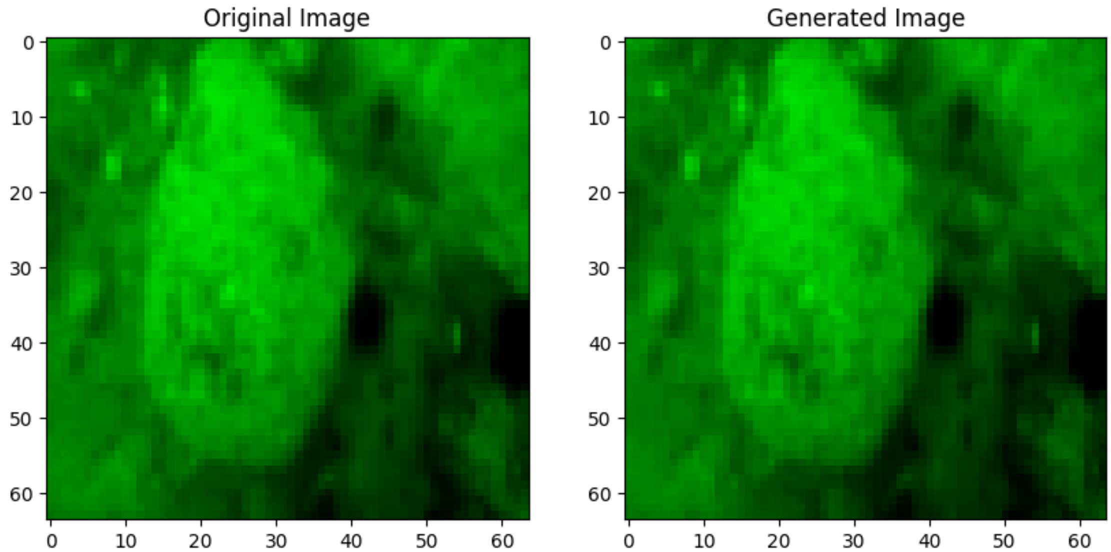
```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```



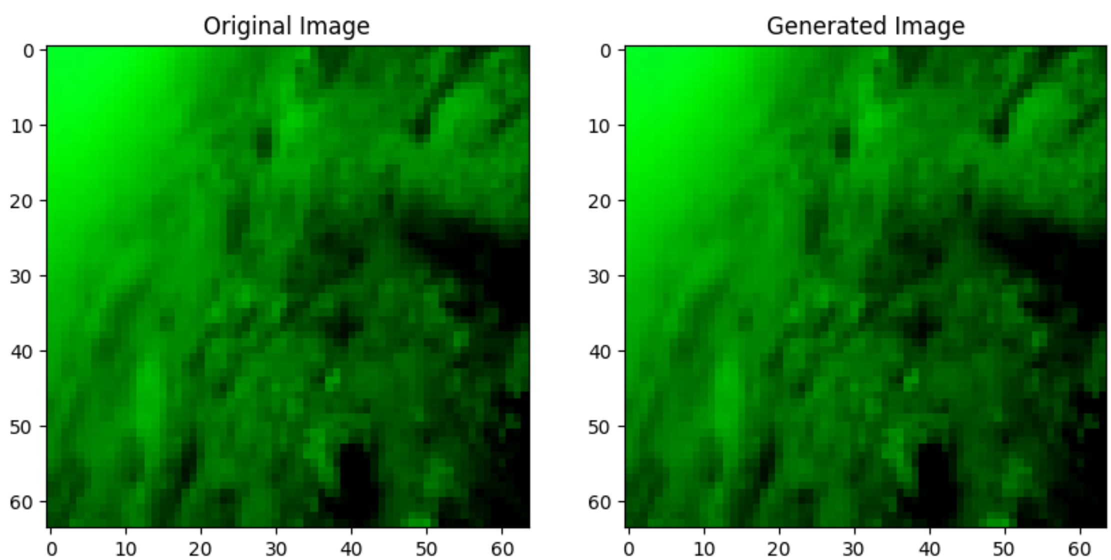

```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```



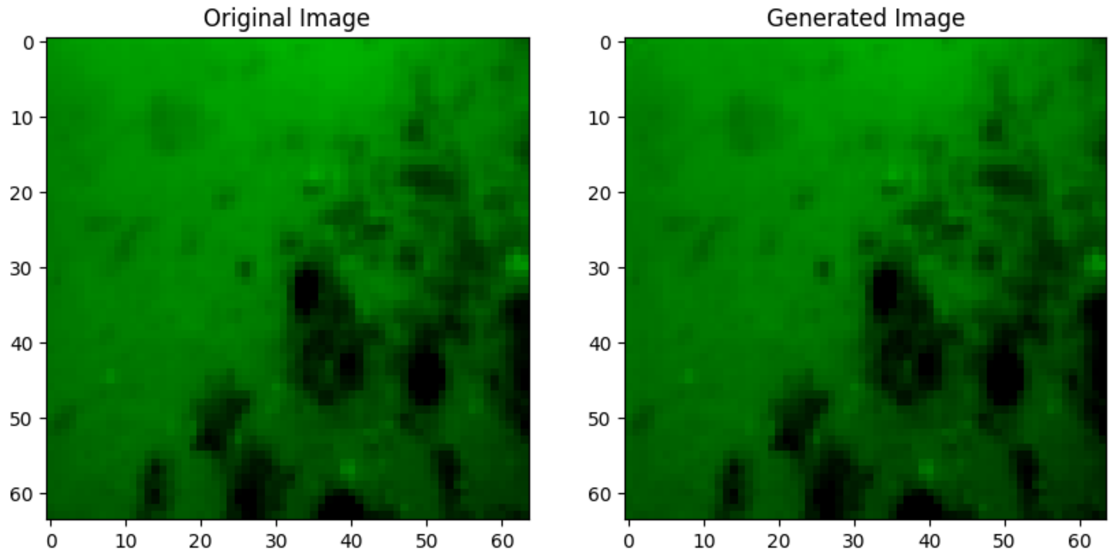
```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```



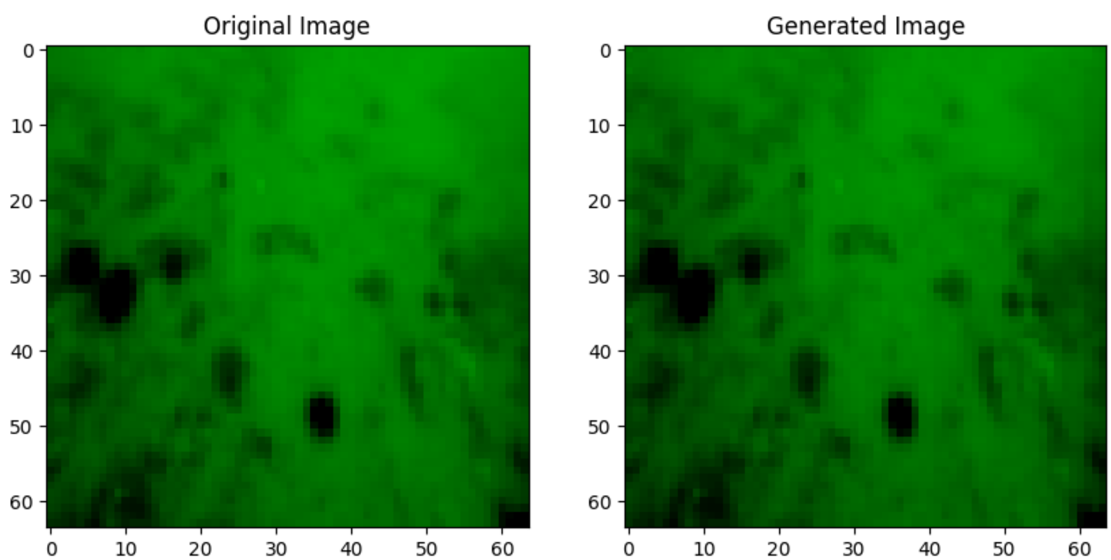

```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```



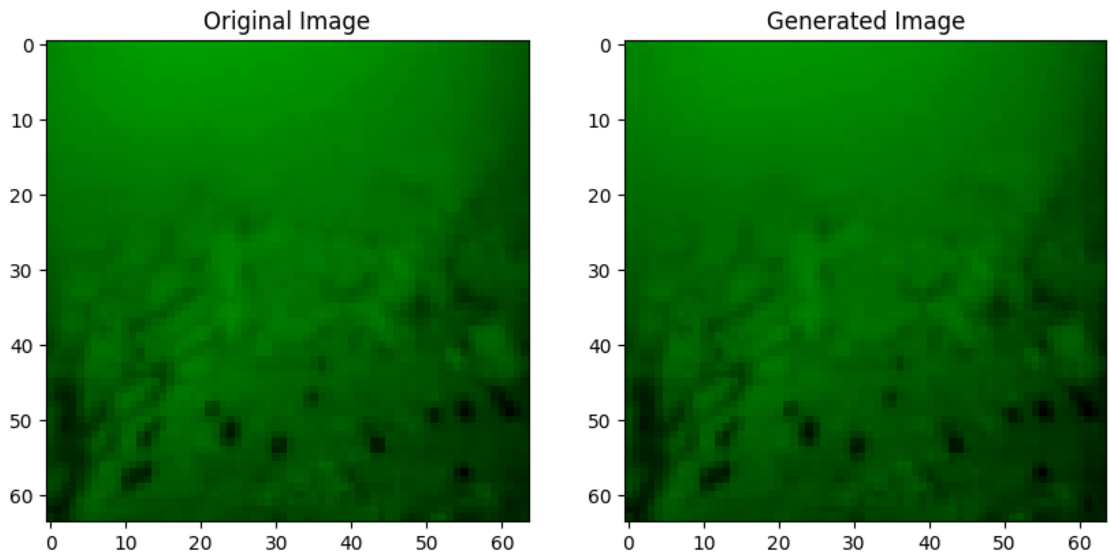
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).



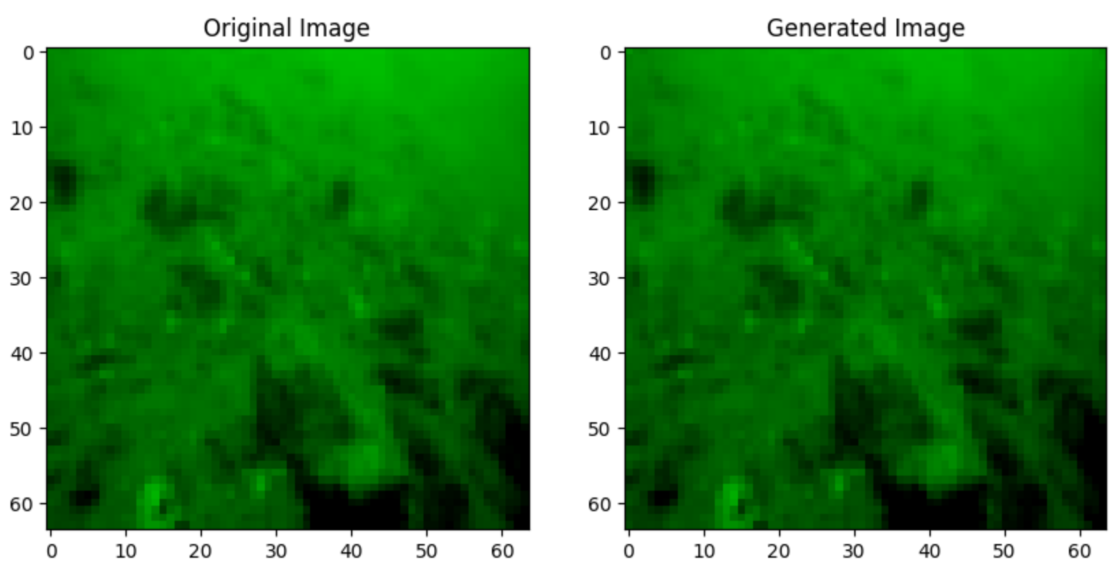
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).



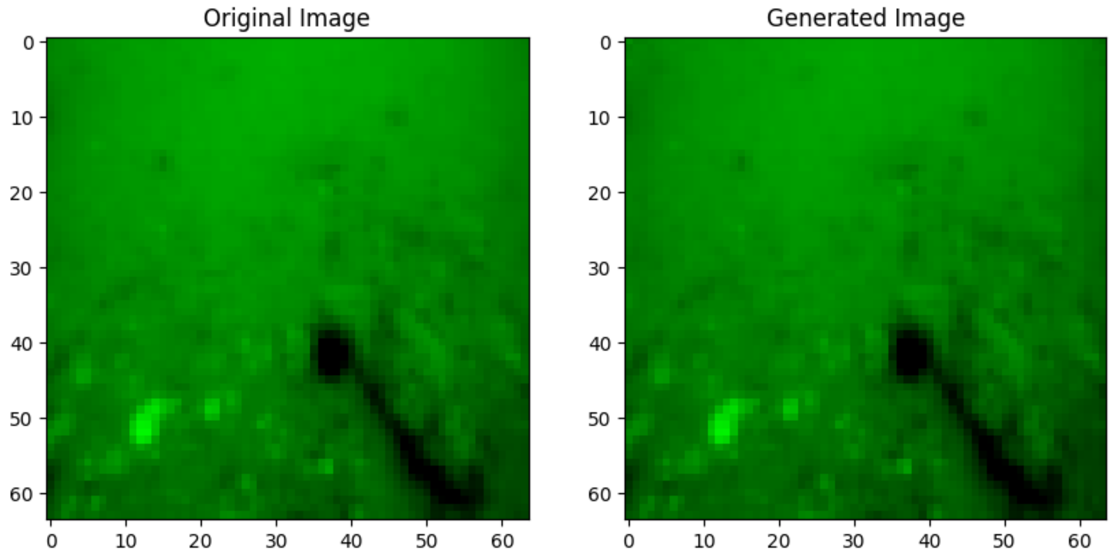
```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```



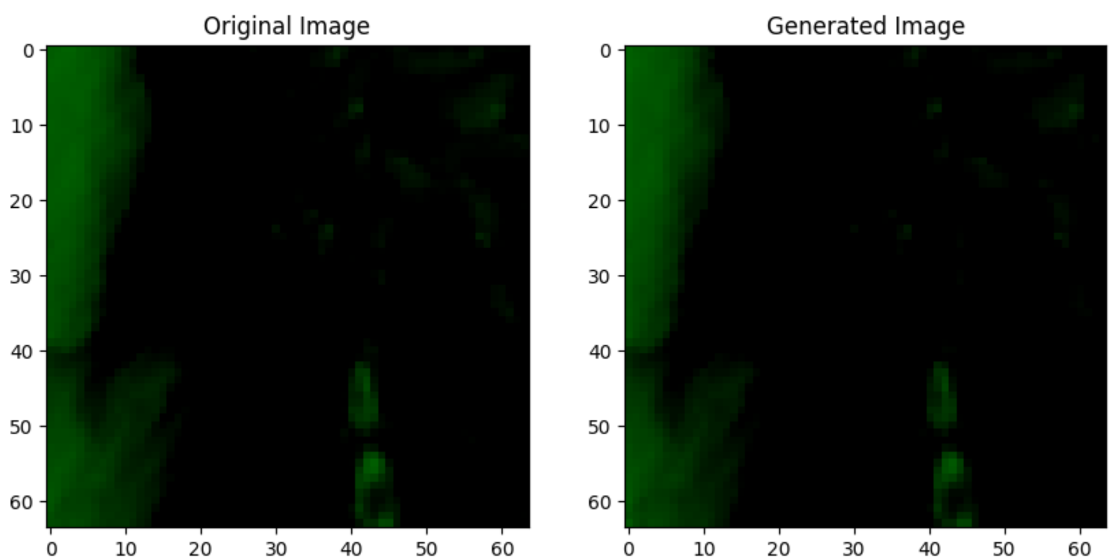

```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```



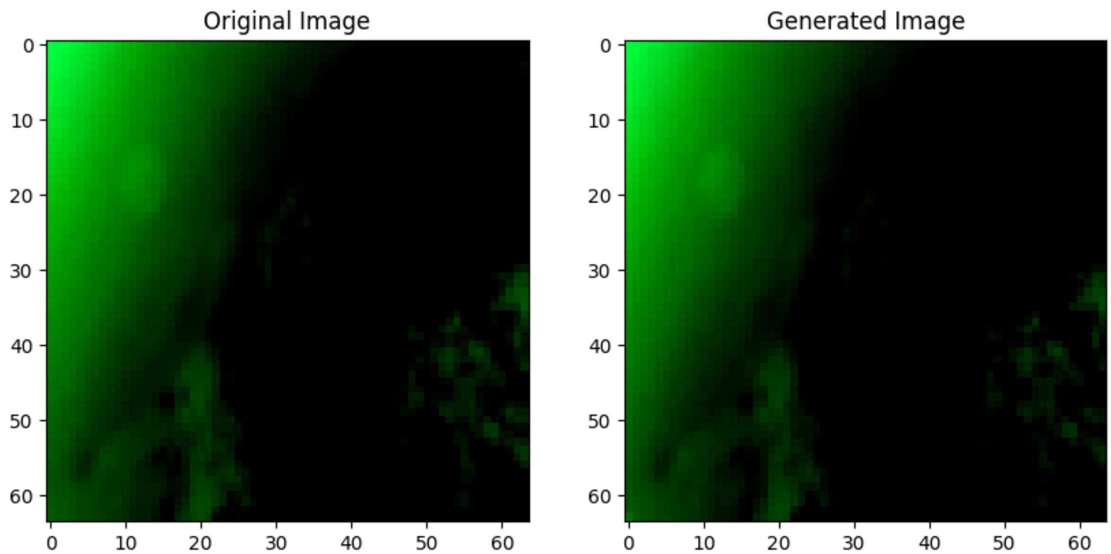
```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```



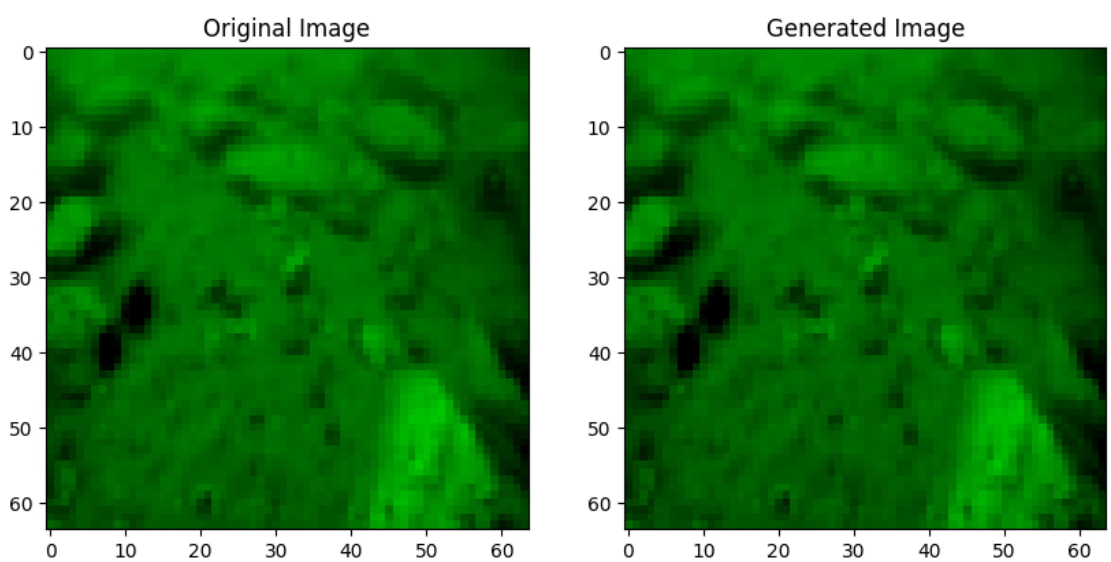

```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```



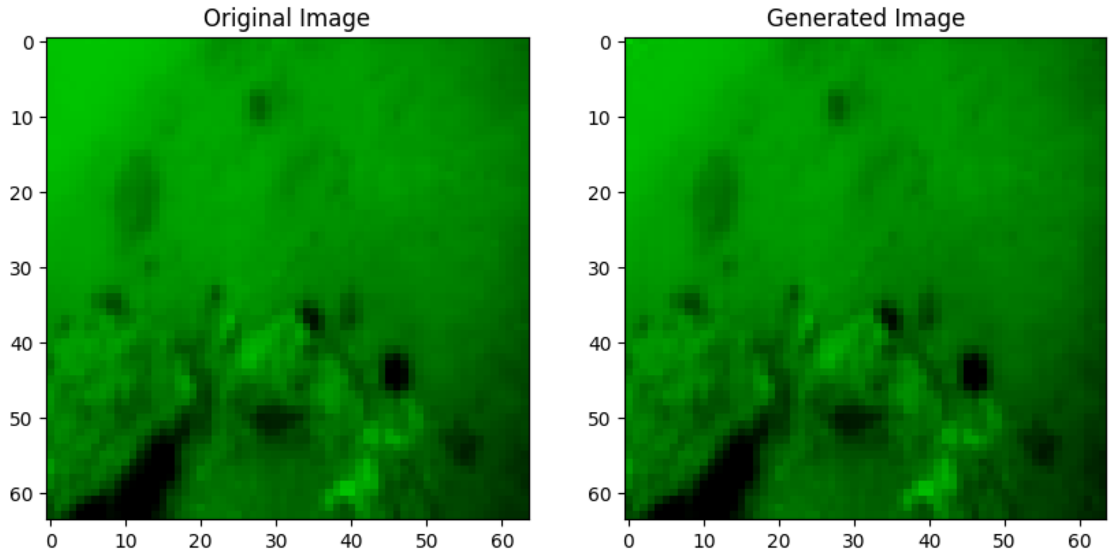
```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```



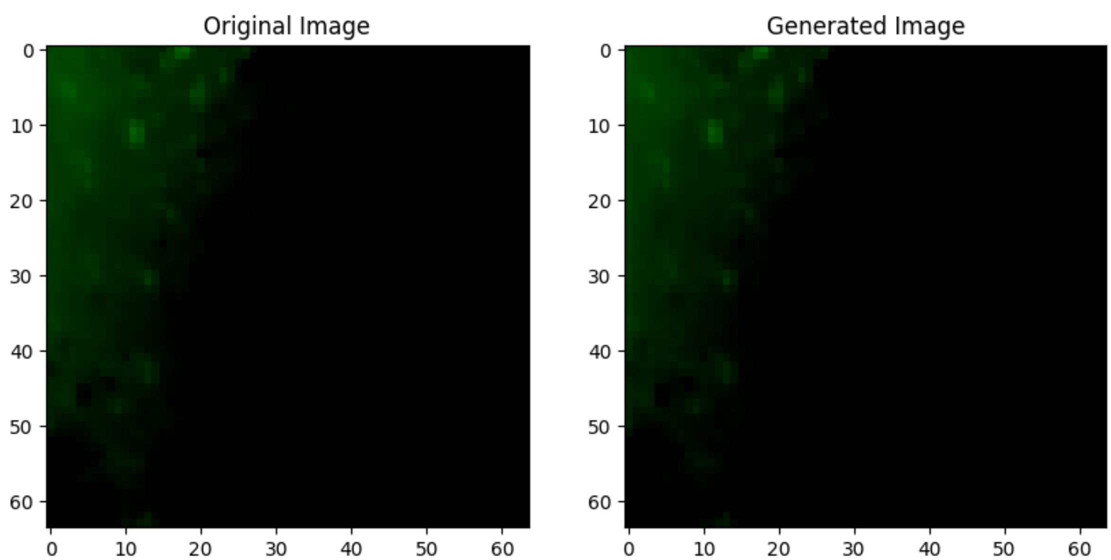

```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```



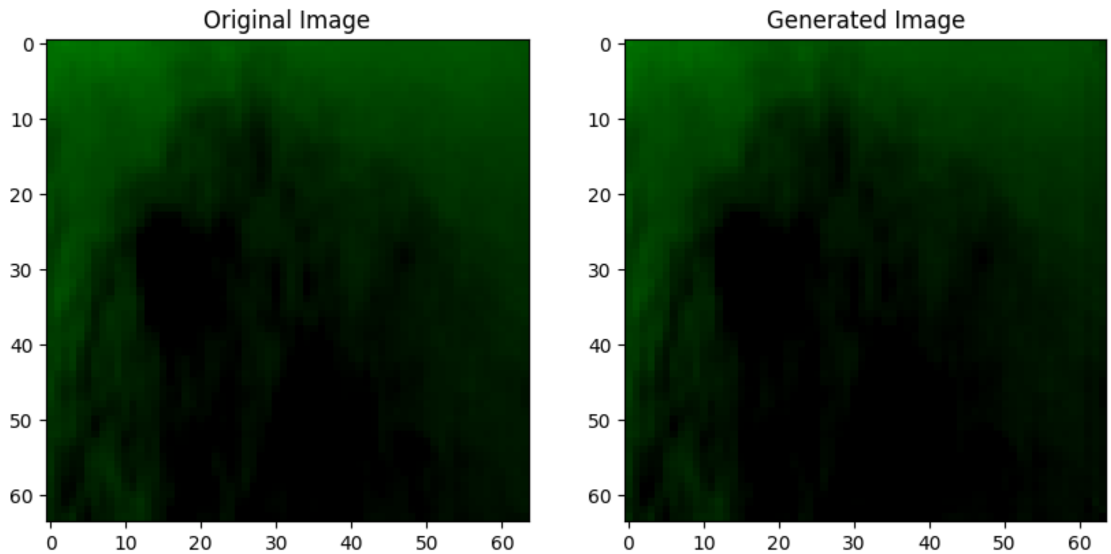
```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```



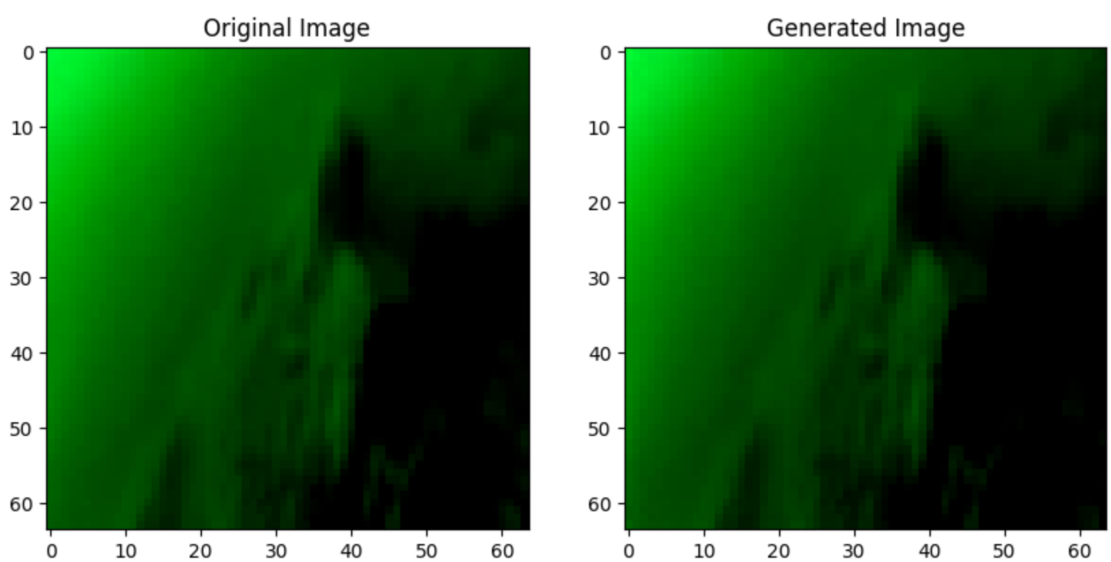

```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```



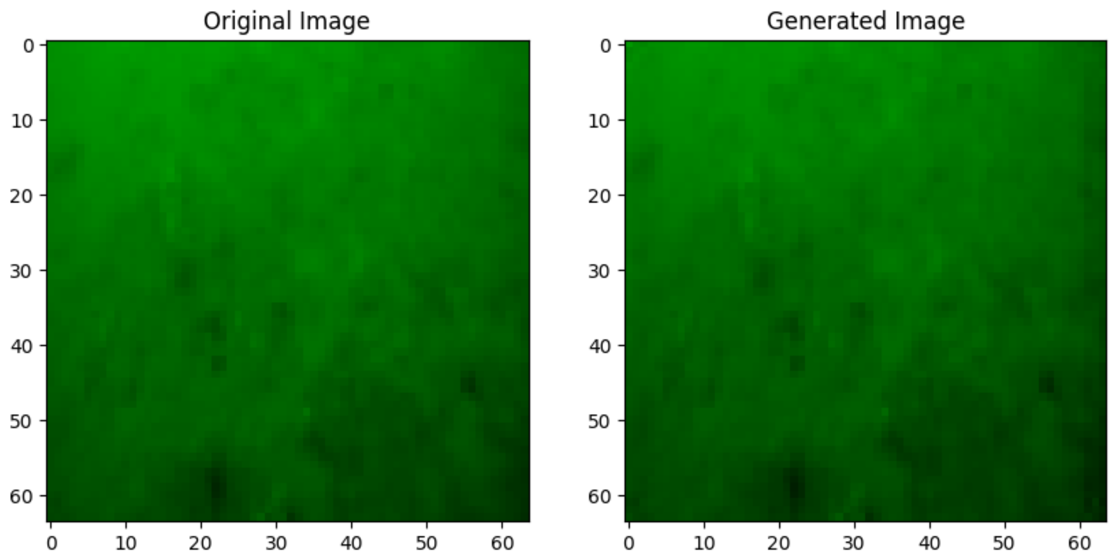
```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```



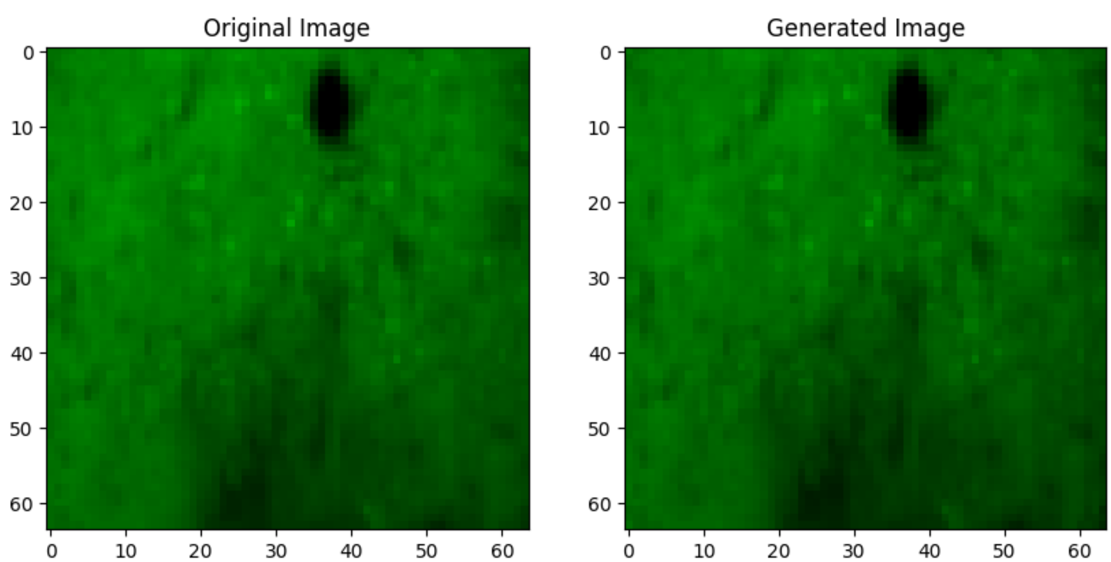

```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```



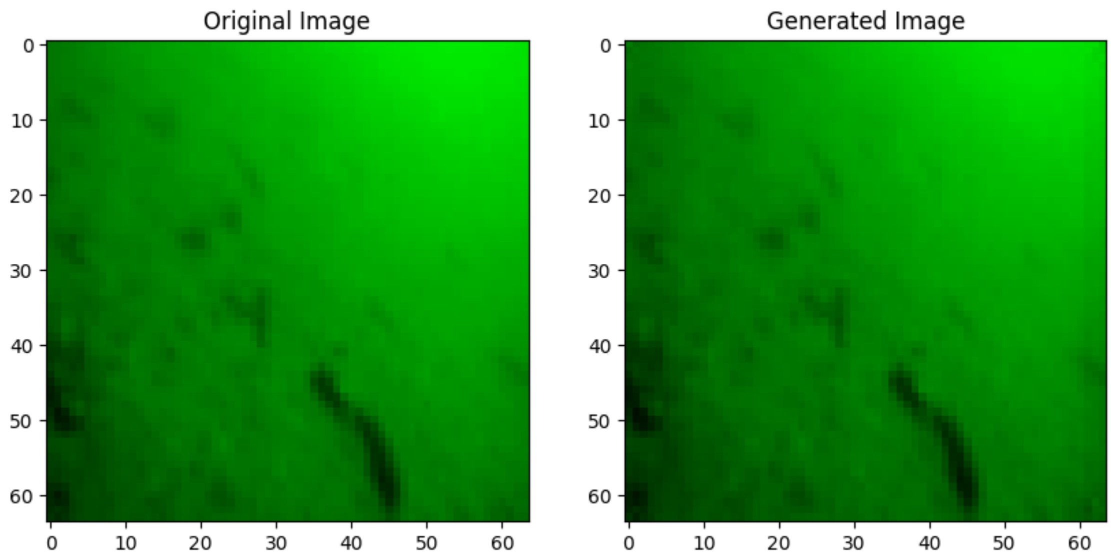
```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```



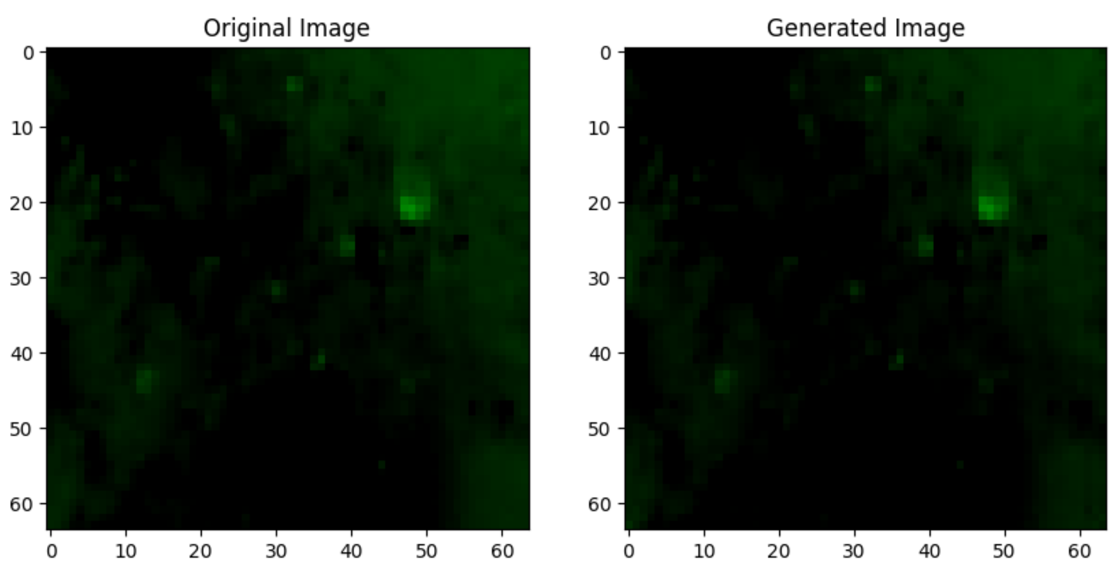

```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```



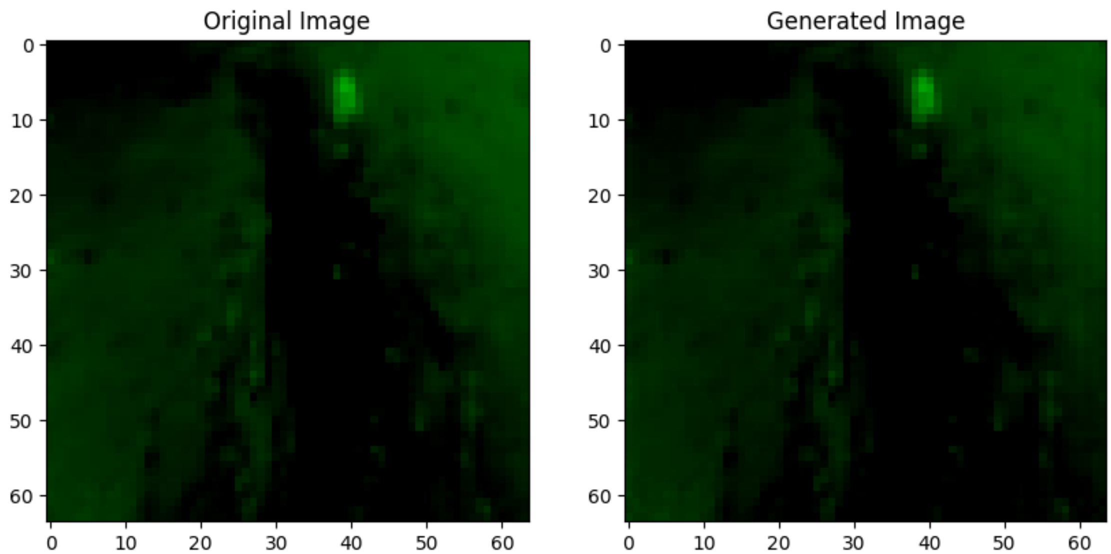
```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```



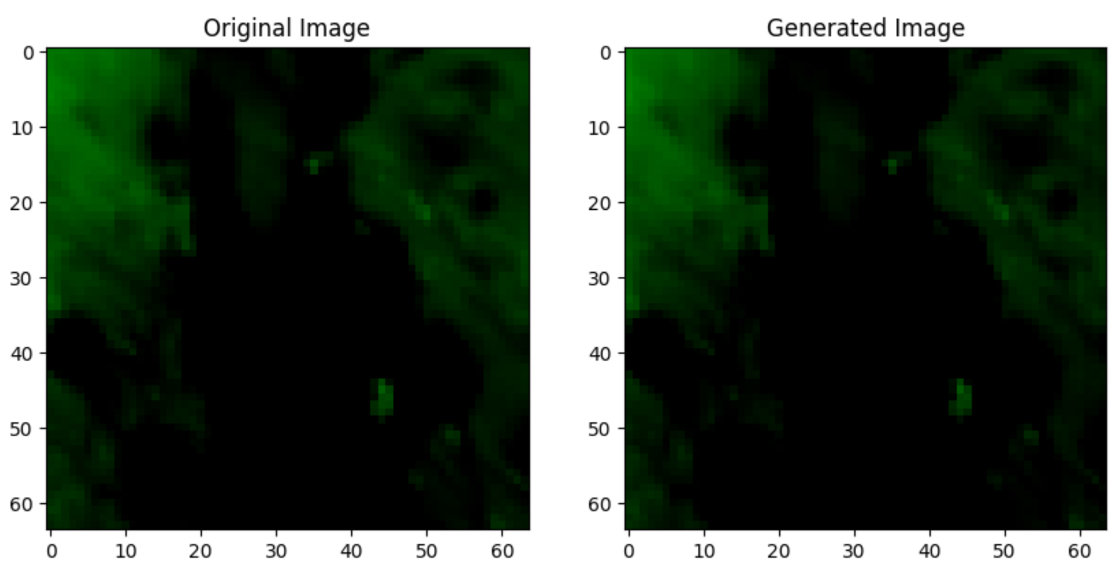

```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```



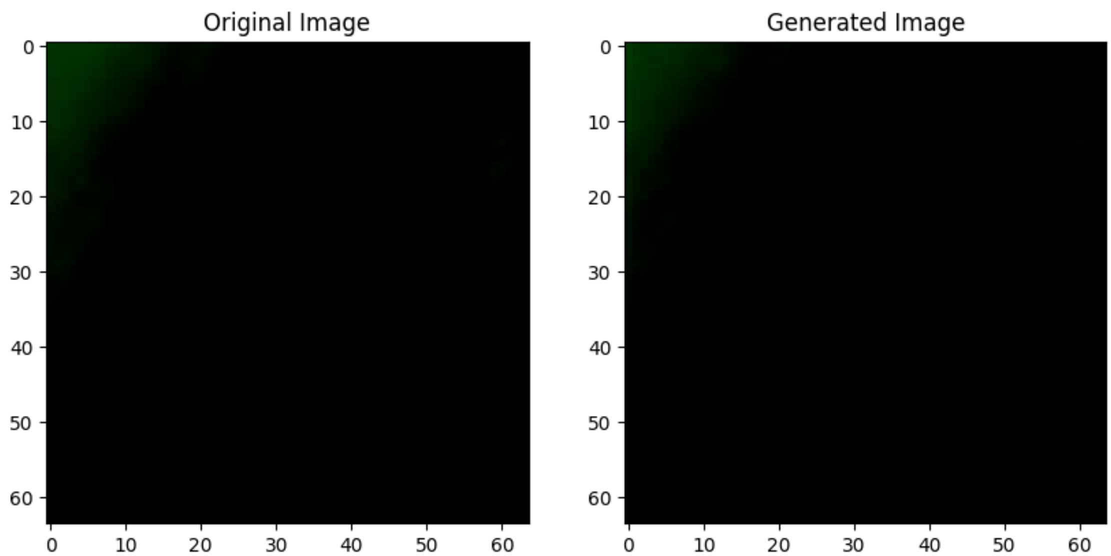
```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```



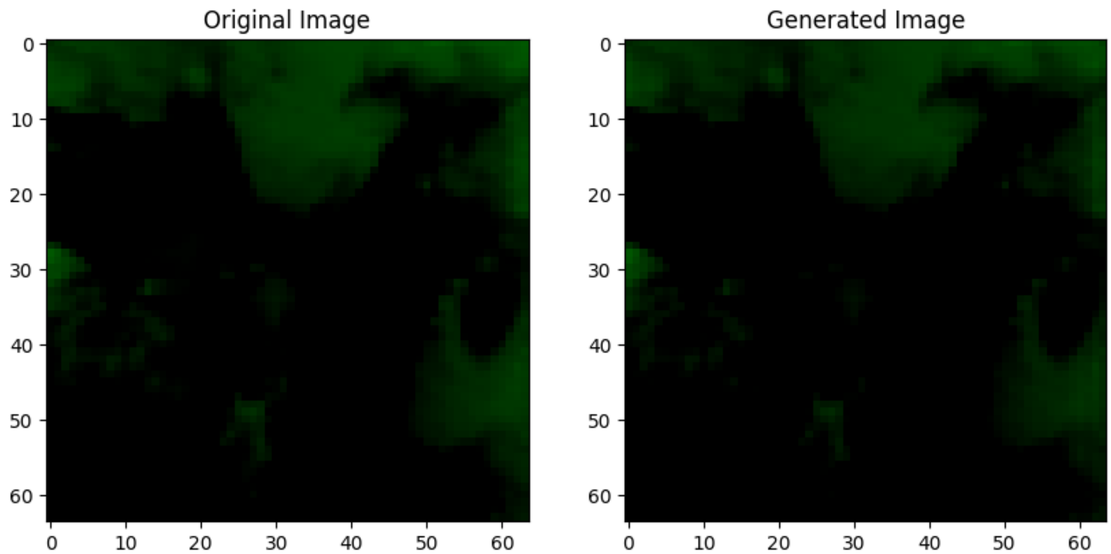

```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```



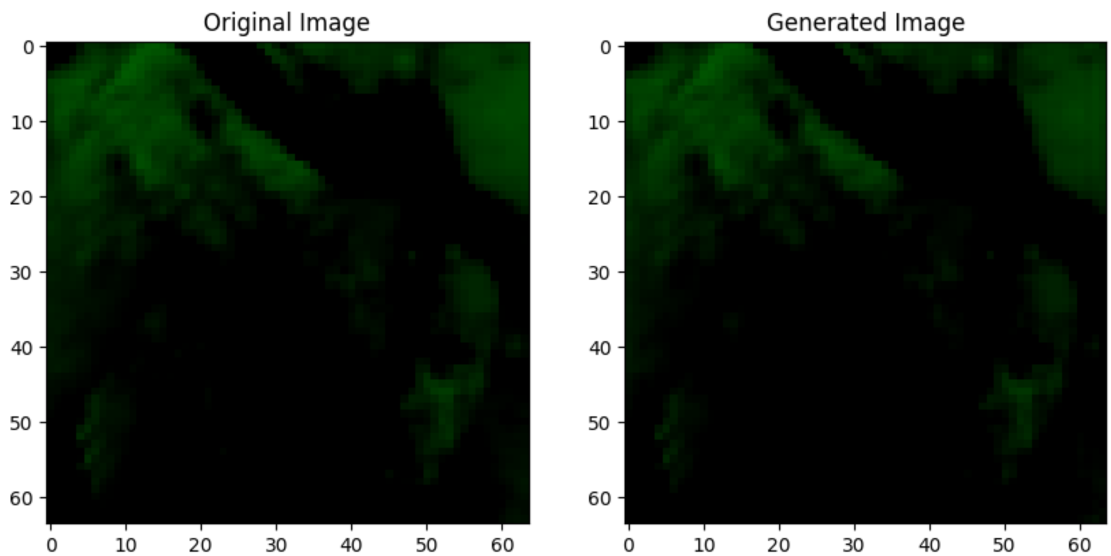
```
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1531:
UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 due to no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1531:
UndefinedMetricWarning: Recall is ill-defined and being set to 0.0 due to no
true samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```




```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```

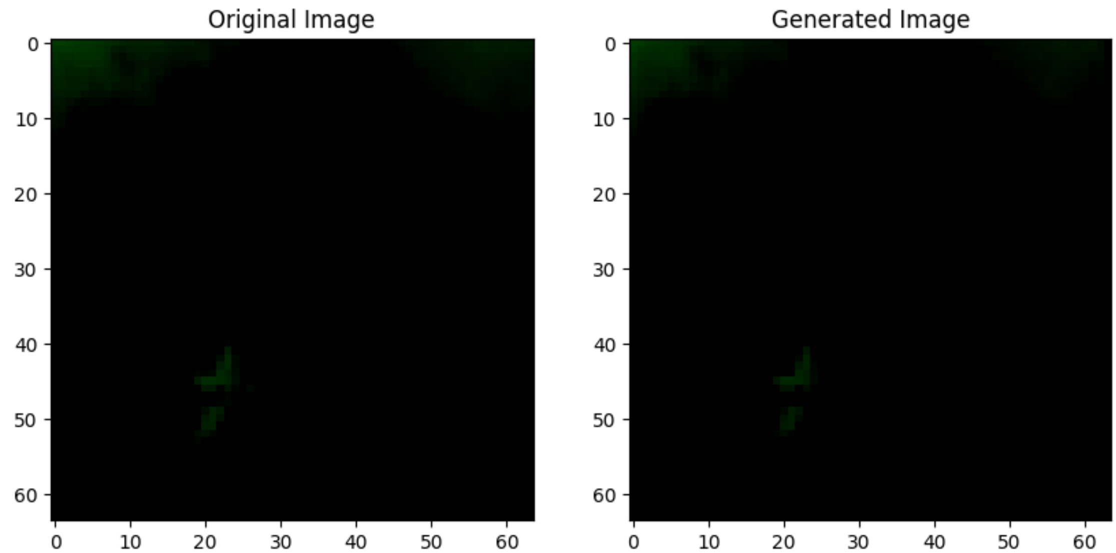



```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```

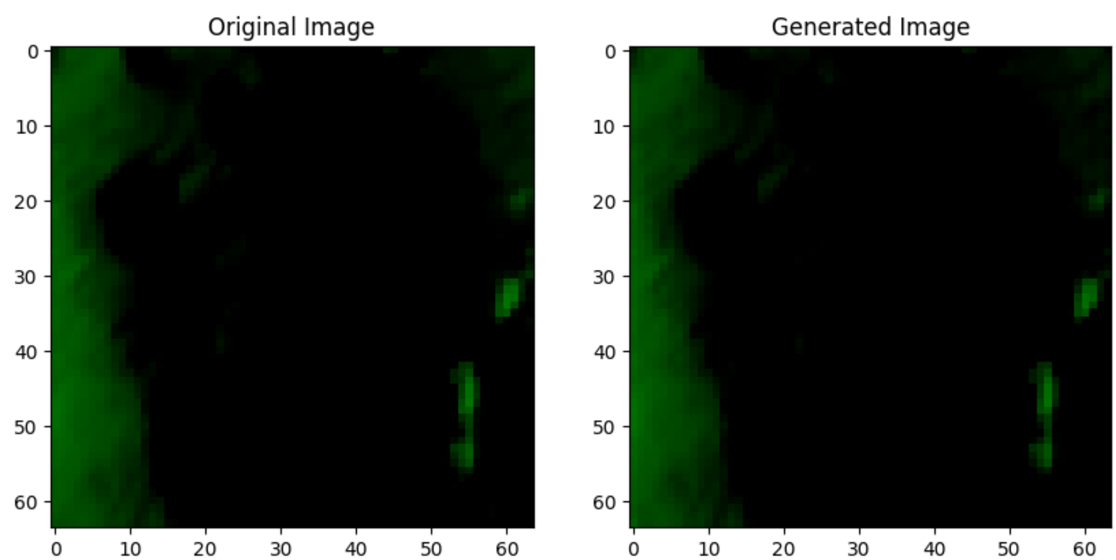



```
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1531:
UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 due to no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1531:
```

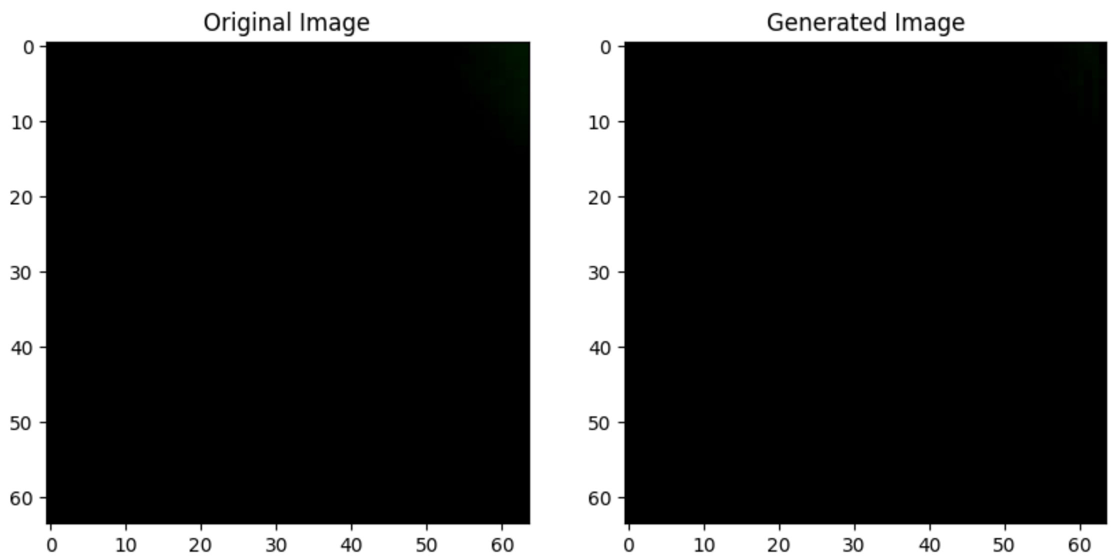
```
UndefinedMetricWarning: Recall is ill-defined and being set to 0.0 due to no
true samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```



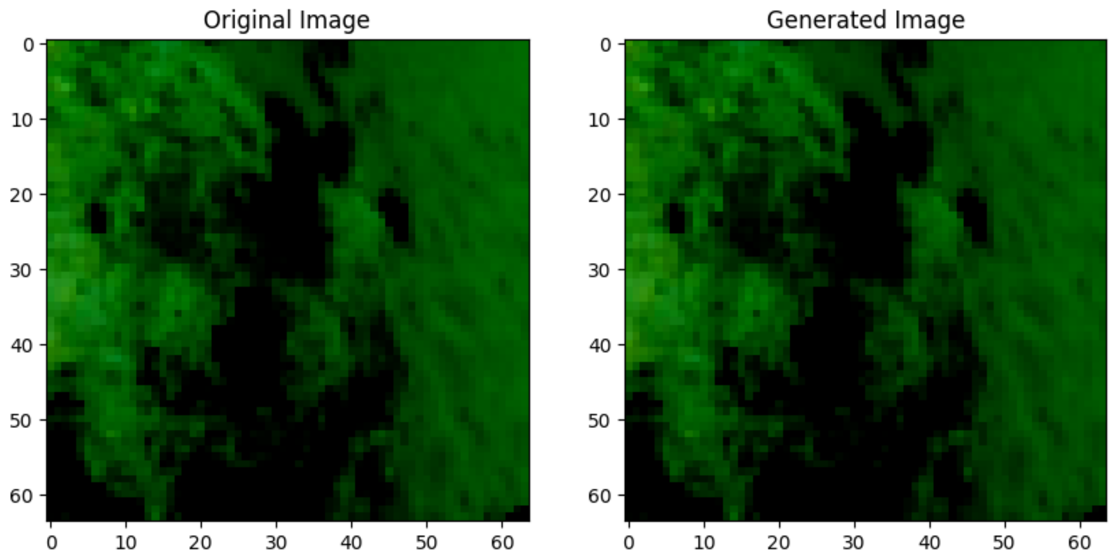

```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```



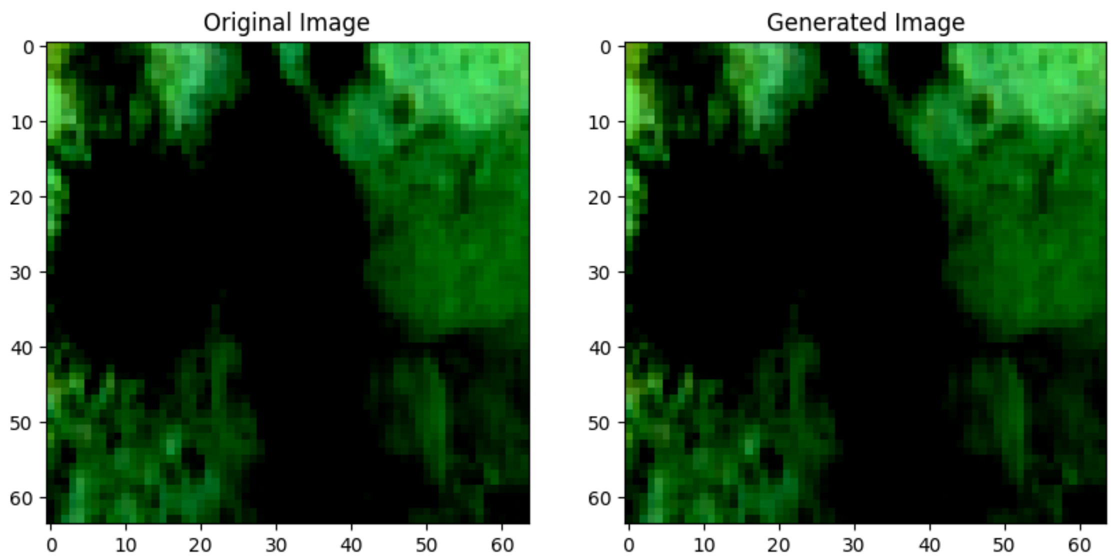
```
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1531:
UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 due to no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1531:
UndefinedMetricWarning: Recall is ill-defined and being set to 0.0 due to no
true samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```



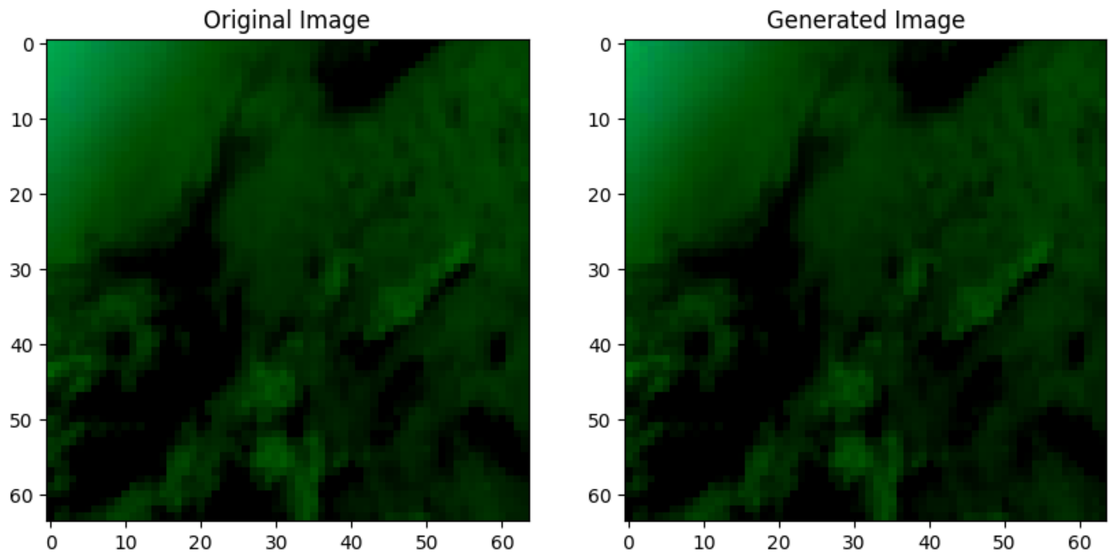

```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```



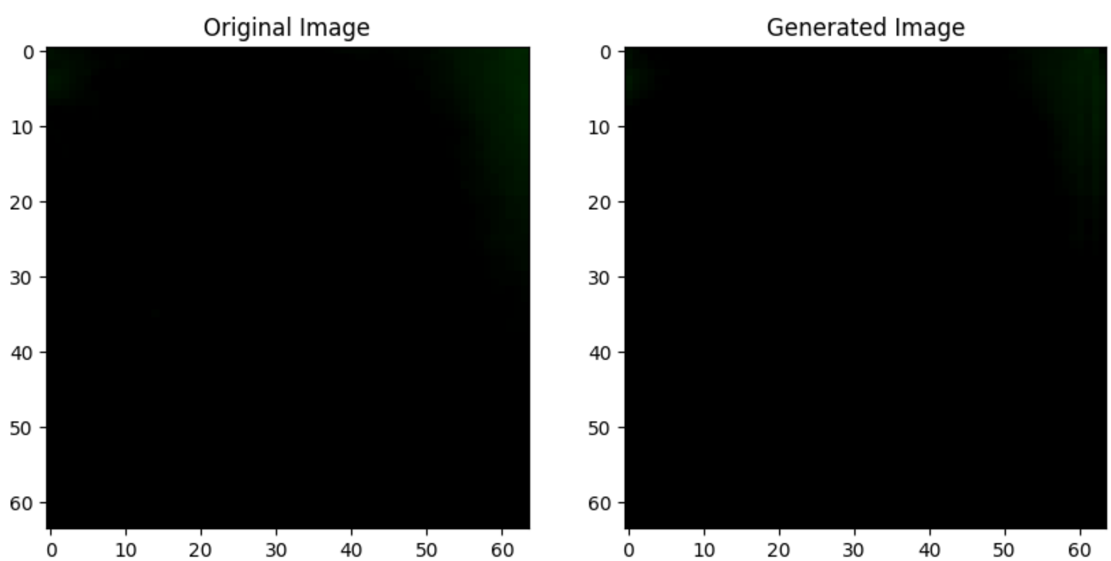
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).



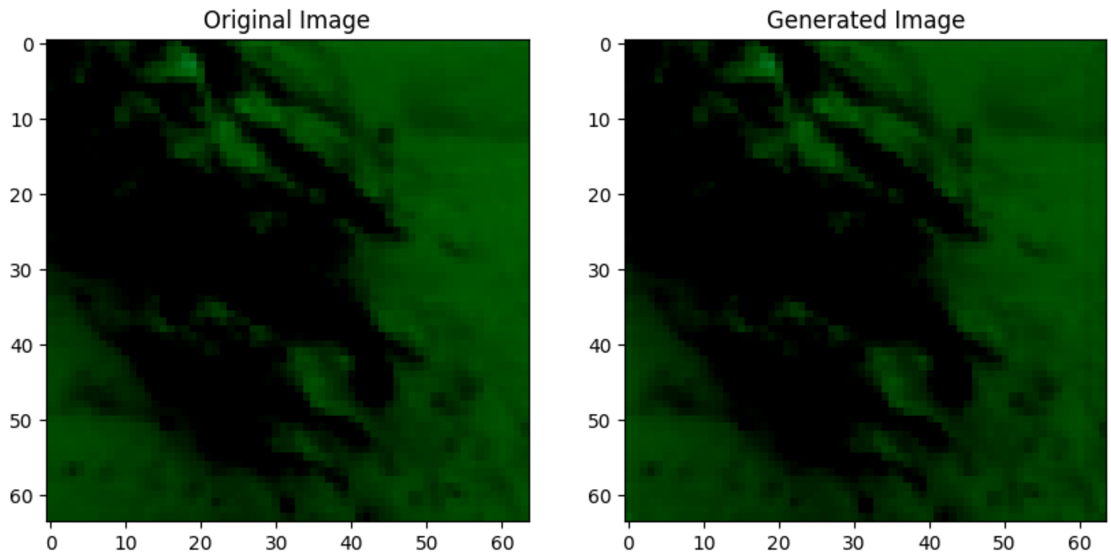
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).



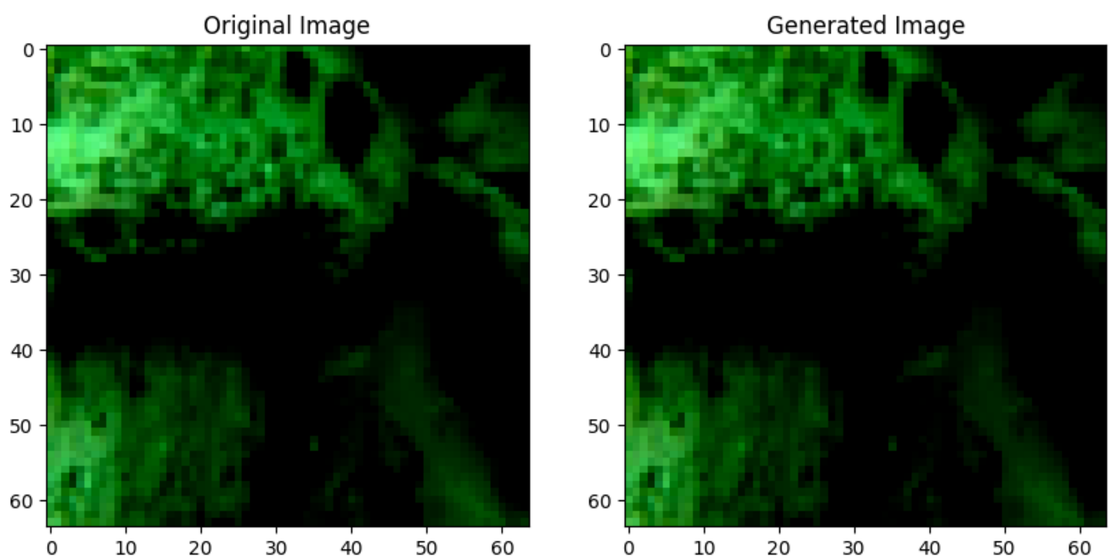
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).



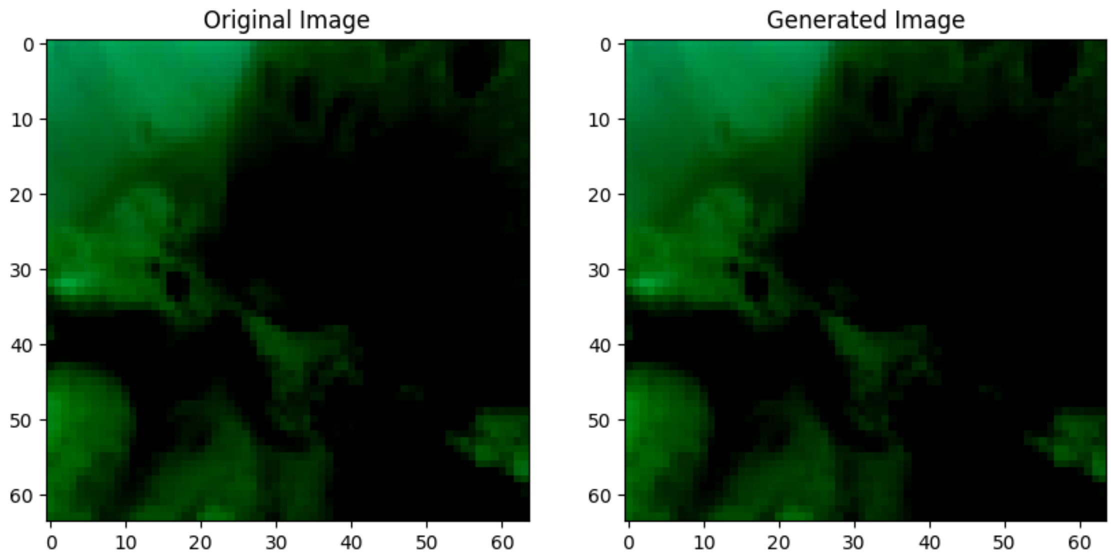
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).



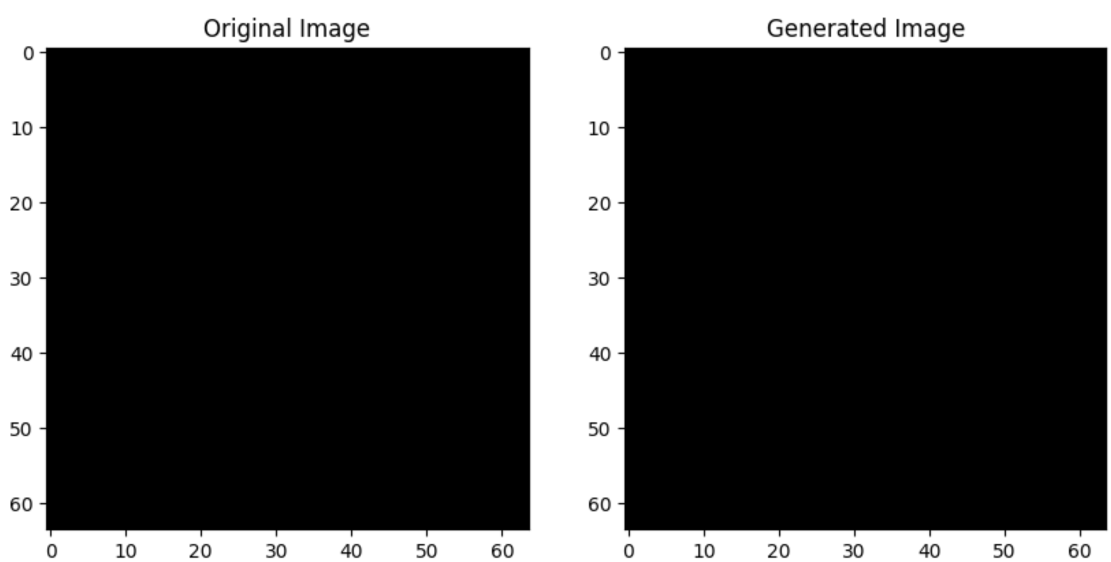

```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```



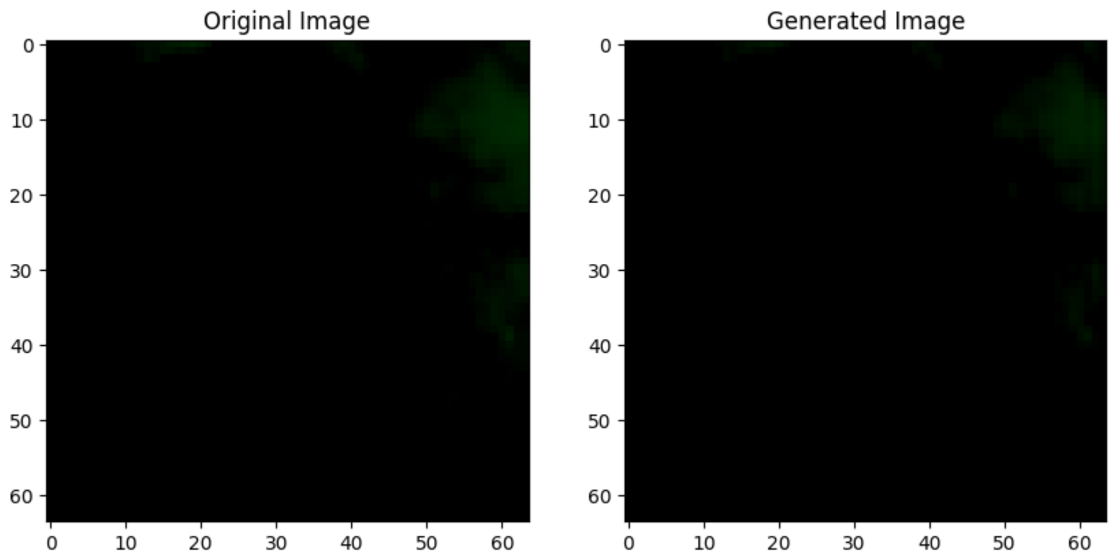

```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```



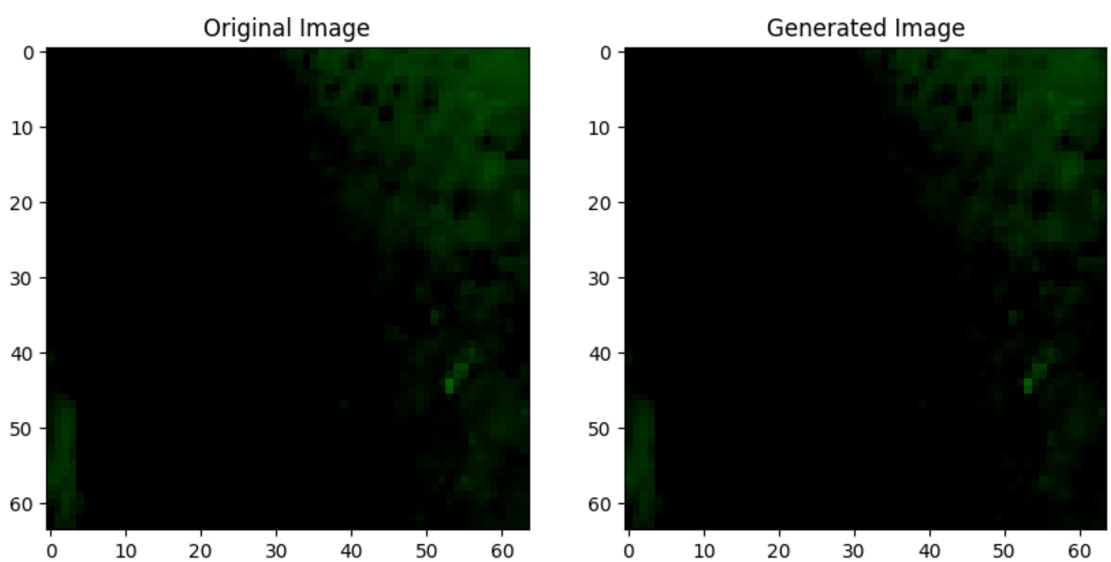

```
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1531:
UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 due to no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1531:
UndefinedMetricWarning: Recall is ill-defined and being set to 0.0 due to no
true samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```



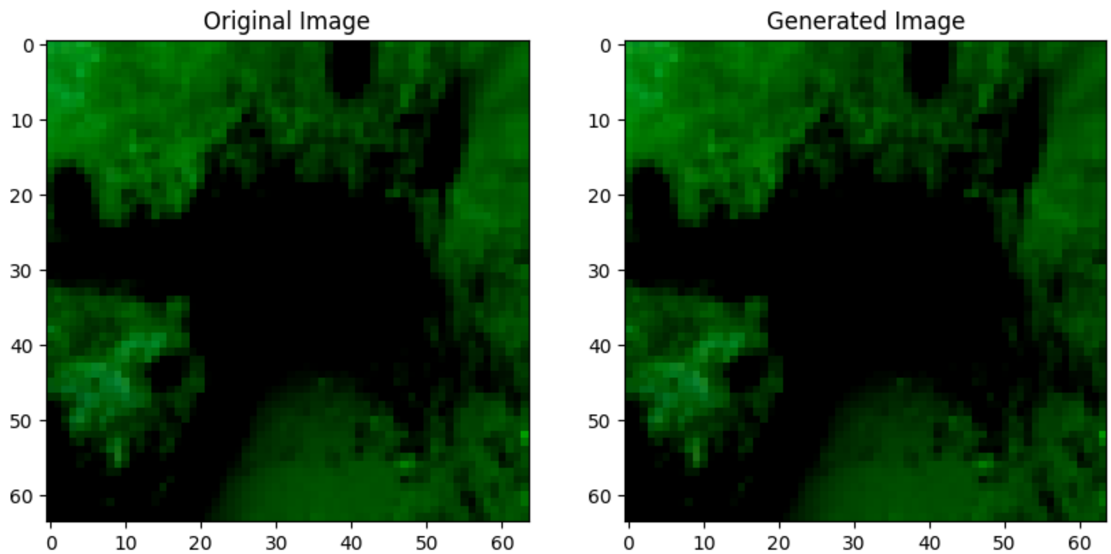
```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```



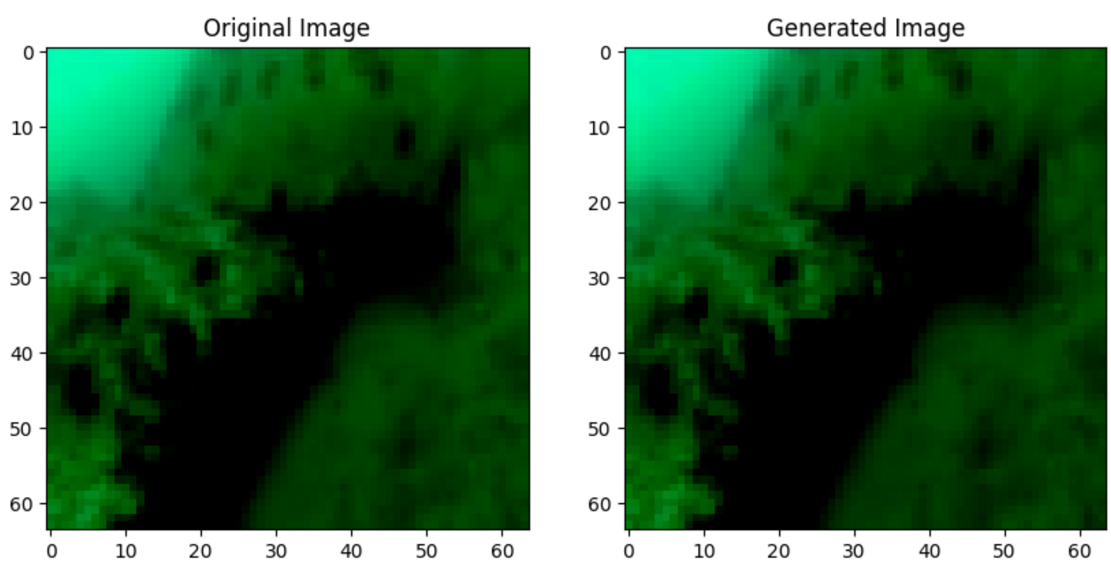

```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```



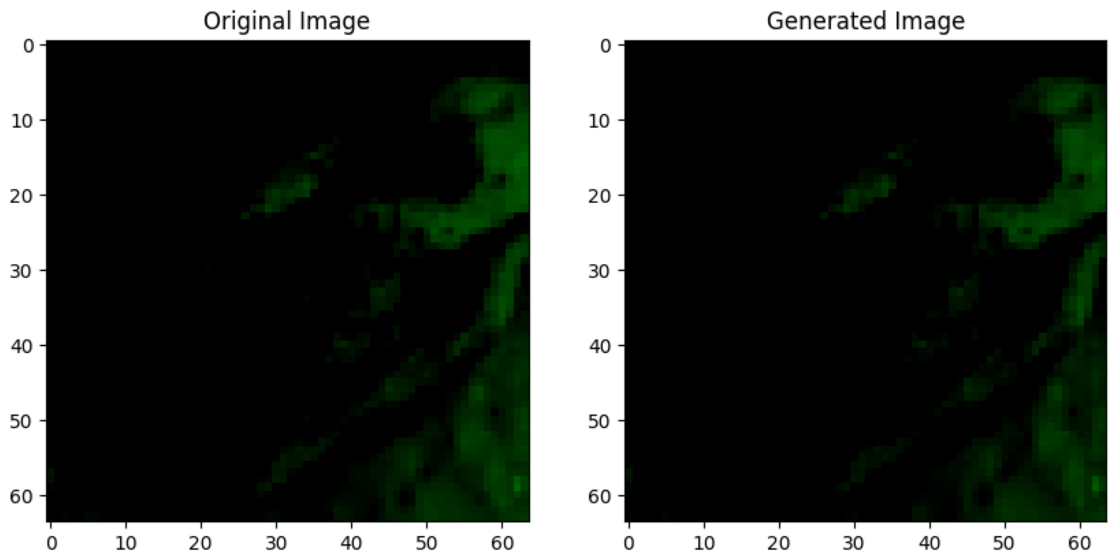
```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```



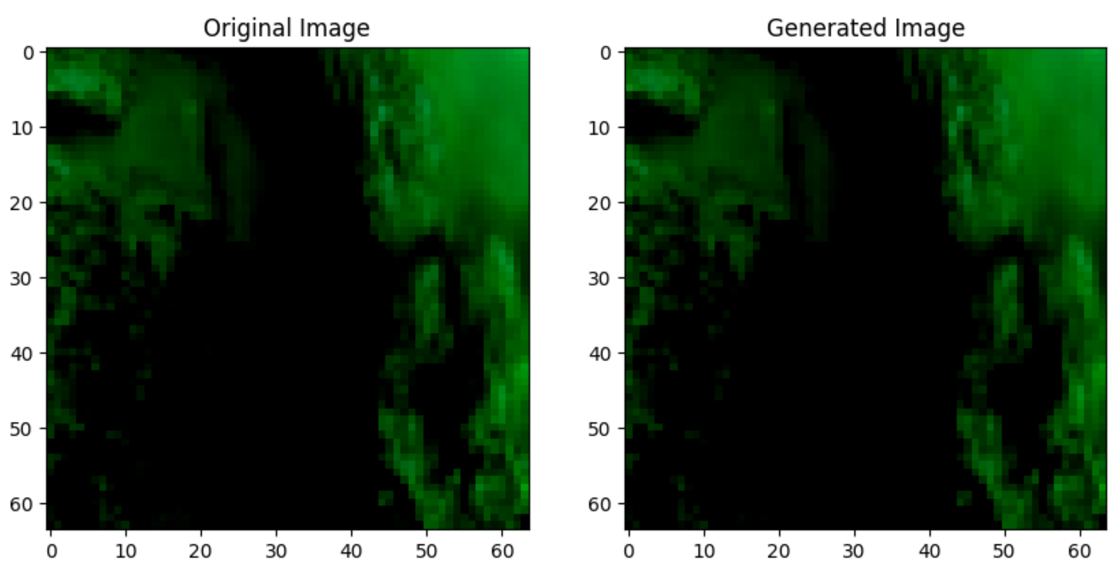

```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```



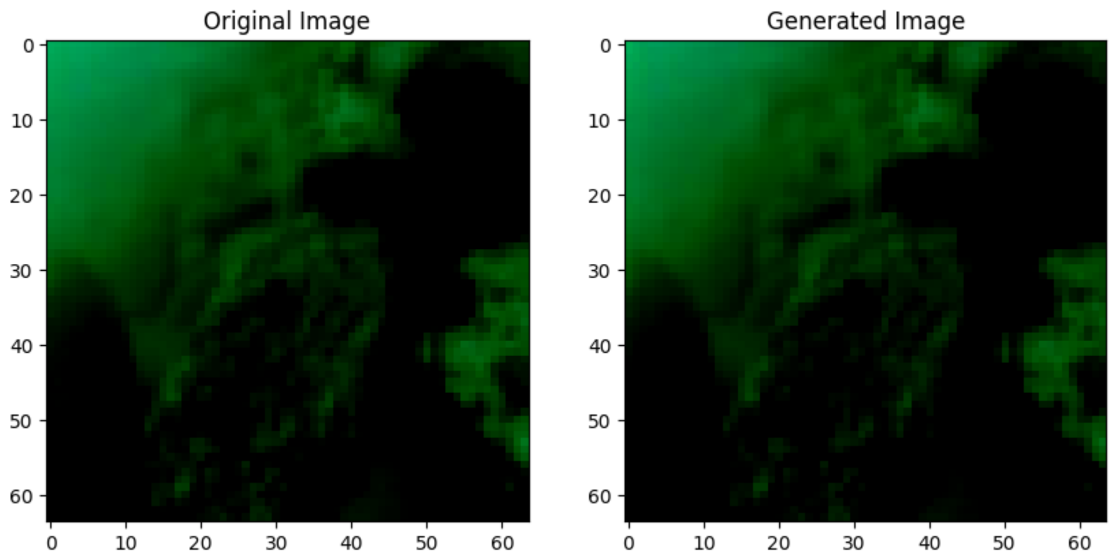
```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```



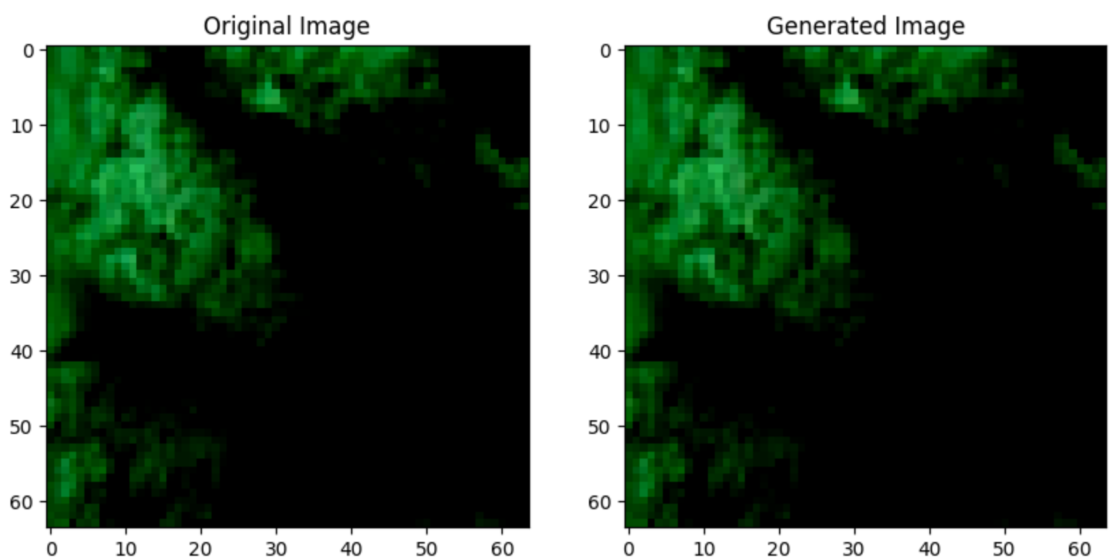

```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```



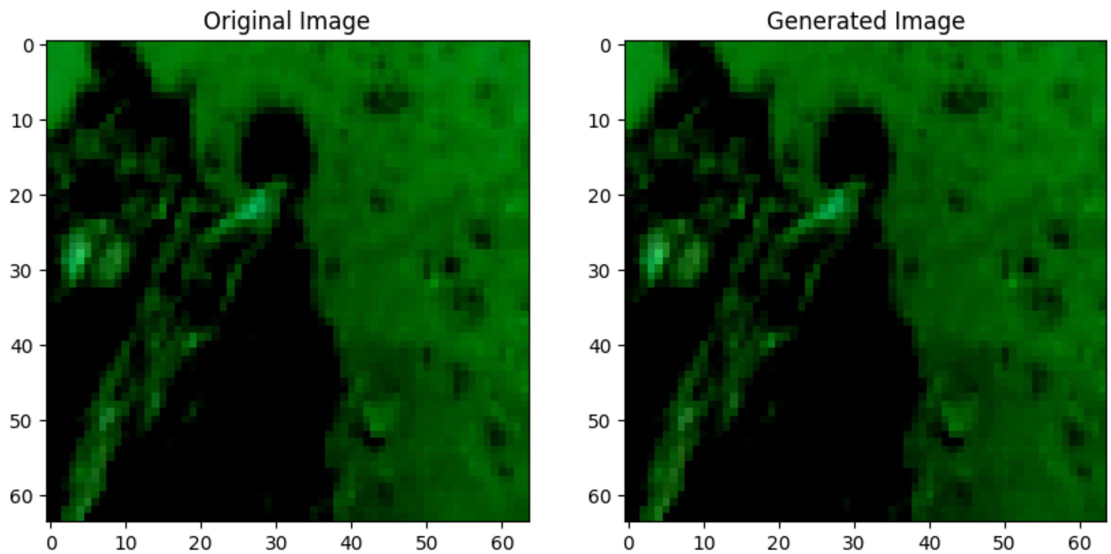
```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```



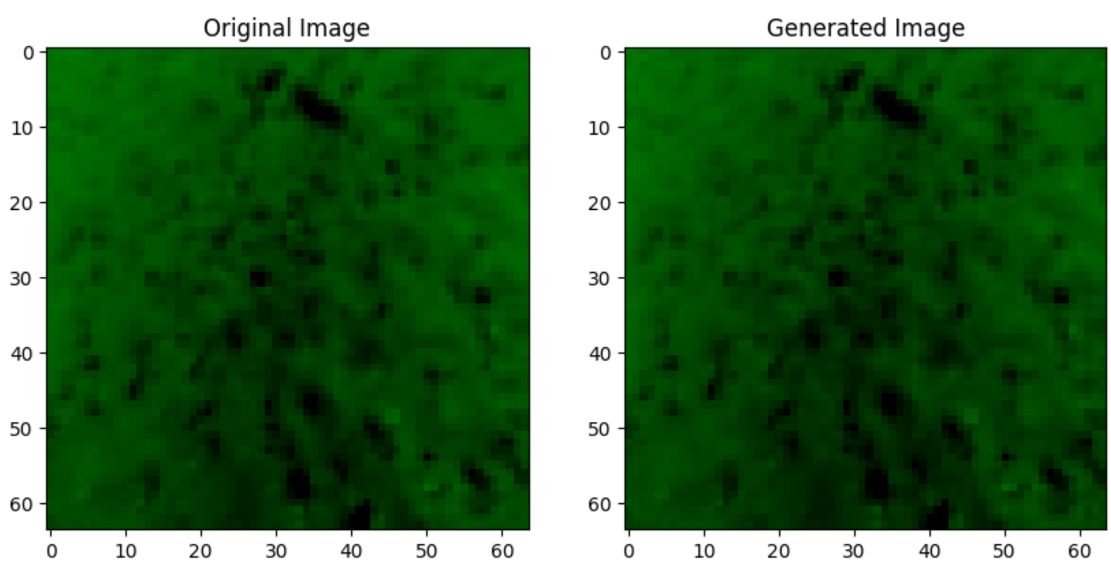

```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```



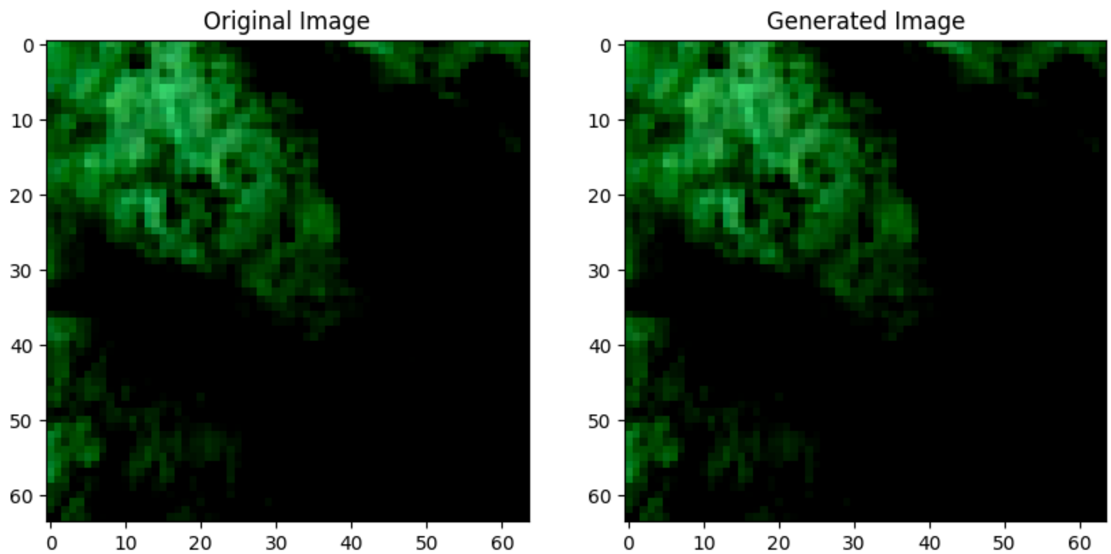
```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```



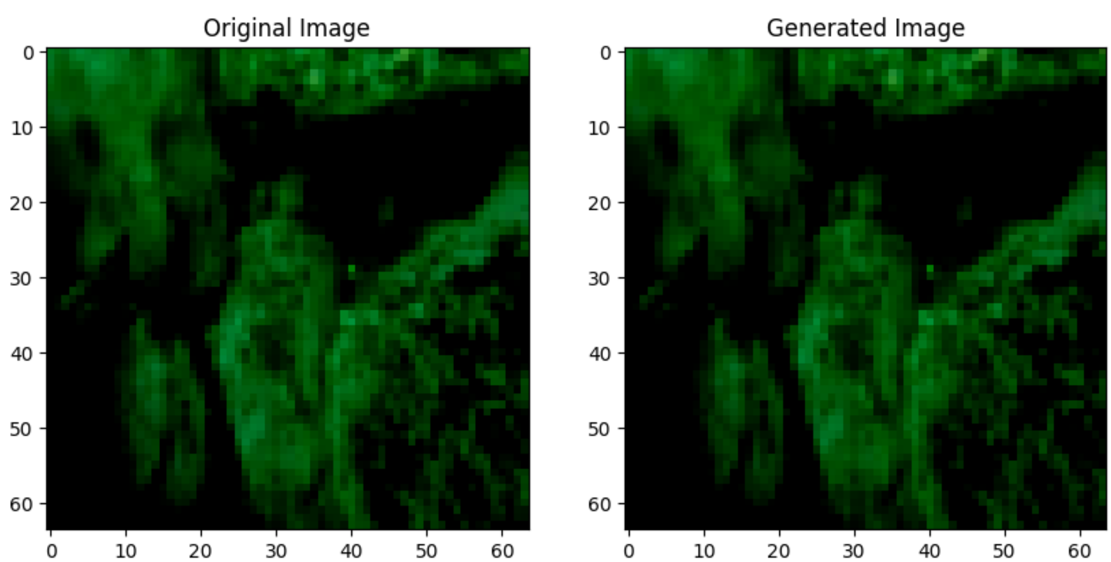

```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```



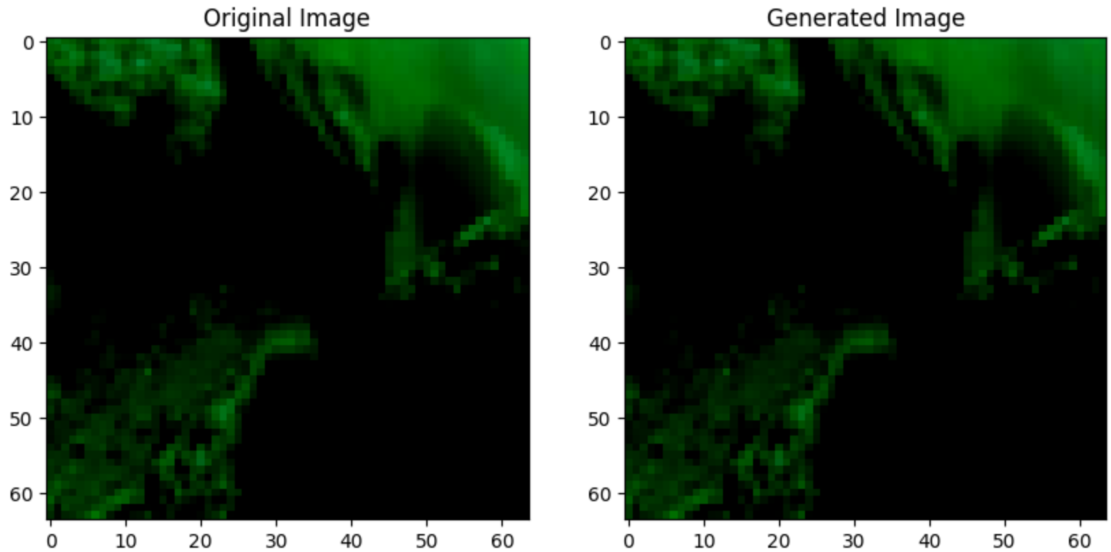
```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```




```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```



```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```




```
Average PSNR: 35.36
Average Precision: 0.9621
Average Recall: 0.9025
Average Accuracy: 0.9925
Average UCIQE: 0.2514
Average UIQM: 1.1910
```