Unet_Denoiser-dataloader

March 28, 2024

[1]:
```
import torch
import torch.nn as nn
from PIL import Image
import os
from IPython.display import display
import numpy as np
import torch.optim as optim
import torch.nn.functional as F
from torchvision import transforms
from torch.utils.data import Dataset, DataLoader
from skimage.metrics import structural_similarity as ssim
import random
from torch.utils.data import random_split
import matplotlib.pyplot as plt
import torchvision
import torchvision.transforms.functional as TF
```

[2]:
```
class UNet_Denoise(nn.Module):

    def __init__(self):
        super(UNet_Denoise, self).__init__()

        # Convolution level 1
        self.conv1_1 = nn.Conv2d(1, 64, kernel_size=3, padding=1)
        self.conv1_2 = nn.Conv2d(64, 64, kernel_size=3, padding=1)
        self.pool1 = nn.MaxPool2d(kernel_size=2, stride=2)

        # C2
        self.conv2_1 = nn.Conv2d(64, 128, kernel_size=3, padding=1)
        self.conv2_2 = nn.Conv2d(128, 128, kernel_size=3, padding=1)
        self.pool2 = nn.MaxPool2d(kernel_size=2, stride=2)

        # C3
        self.conv3_1 = nn.Conv2d(128, 256, kernel_size=3, padding=1)
        self.conv3_2 = nn.Conv2d(256, 256, kernel_size=3, padding=1)
        self.pool3 = nn.MaxPool2d(kernel_size=2, stride=2)

        # C4
```

```
        self.conv4_1 = nn.Conv2d(256, 512, kernel_size=3, padding=1)
        self.conv4_2 = nn.Conv2d(512, 512, kernel_size=3, padding=1)
        self.pool4 = nn.MaxPool2d(kernel_size=2, stride=2)

        # Bottleneck
        self.conv5_1 = nn.Conv2d(512, 1024, kernel_size=3, padding=1)
        self.conv5_2 = nn.Conv2d(1024, 1024, kernel_size=3, padding=1)

        # Transposed convolution level 5
        self.transpose4 = nn.ConvTranspose2d(1024, 512, kernel_size=3,␣
↪stride=2, padding=1, output_padding=1)
        self.upconv4_2 = nn.Conv2d(1024, 512, kernel_size=3, padding=1)
        self.upconv4_1 = nn.Conv2d(512, 512, kernel_size=3, padding=1)

        # T3
        self.transpose3 = nn.ConvTranspose2d(512, 256, kernel_size=3, stride=2,␣
↪padding=1, output_padding=1)
        self.upconv3_2 = nn.Conv2d(512, 256, kernel_size=3, padding=1)
        self.upconv3_1 = nn.Conv2d(256, 256, kernel_size=3, padding=1)

        # T2
        self.transpose2 = nn.ConvTranspose2d(256, 128, kernel_size=3, stride=2,␣
↪padding=1, output_padding=1)
        self.upconv2_2 = nn.Conv2d(256, 128, kernel_size=3, padding=1)
        self.Tconv2_1 = nn.Conv2d(128, 128, kernel_size=3, padding=1)

        # T1
        self.transpose1 = nn.ConvTranspose2d(128, 64, kernel_size=3, stride=2,␣
↪padding=1, output_padding=1)
        self.upconv1_3 = nn.Conv2d(128, 64, kernel_size=3, padding=1)
        self.upconv1_2 = nn.Conv2d(64, 64, kernel_size=3, padding=1)
        self.upconv1_1 = nn.Conv2d(64, 1, kernel_size=3, padding=1)

        # Skip connections
        self.skip_connection1 = nn.Conv2d(64, 64, kernel_size=1)
        self.skip_connection2 = nn.Conv2d(128, 128, kernel_size=1)
        self.skip_connection3 = nn.Conv2d(256, 256, kernel_size=1)
        self.skip_connection4 = nn.Conv2d(512, 512, kernel_size=1)

    def forward(self, x):
        s1_1 = self.conv1_1(x)
        rel1 = F.relu(s1_1)
        s1_2 = self.conv1_2(rel1)
        rel2 = F.relu(s1_2)
        pooled1 = self.pool1(rel2)

        s2_1 = self.conv2_1(pooled1)
```

```
        rel3 = F.relu(s2_1)
        s2_2 = self.conv2_2(rel3)
        rel4 = F.relu(s2_2)
        pooled2 = self.pool2(rel4)

        s3_1 = self.conv3_1(pooled2)
        rel5 = F.relu(s3_1)
        s3_2 = self.conv3_2(rel5)
        rel6 = F.relu(s3_2)
        pooled3 = self.pool3(rel6)

        s4_1 = self.conv4_1(pooled3)
        rel7 = F.relu(s4_1)
        s4_2 = self.conv4_2(rel7)
        rel8 = F.relu(s4_2)
        pooled4 = self.pool4(rel8)

        # bottleneck
        s5_1 = self.conv5_1(pooled4)
        rel9 = F.relu(s5_1)
        s5_2 = self.conv5_2(rel9)
        rel10 = F.relu(s5_2)

        # after bottleneck
        up4 = self.transpose4(rel10)
        skip_con4 = self.skip_connection4(rel8)
        up4 = torch.cat([up4, skip_con4], dim=1)
        upconv4_2 = self.upconv4_2(up4)
        uprel4_2 = F.relu(upconv4_2)
        upconv4_1 = self.upconv4_1(uprel4_2)
        uprel4_1 = F.relu(upconv4_1)

        up3 = self.transpose3(uprel4_1)
        skip_con3 = self.skip_connection3(rel6)
        up3 = torch.cat([up3, skip_con3], dim=1)
        upconv3_2 = self.upconv3_2(up3)
        uprel3_2 = F.relu(upconv3_2)
        upconv3_1 = self.upconv3_1(uprel3_2)
        uprel3_1 = F.relu(upconv3_1)

        up2 = self.transpose2(uprel3_1)
        skip_con2 = self.skip_connection2(rel4)
        up2 = torch.cat([up2, skip_con2], dim=1)
        upconv2_2 = self.upconv2_2(up2)
        uprel2_2 = F.relu(upconv2_2)
        Tconv2_1 = self.Tconv2_1(uprel2_2)
        uprel2_1 = F.relu(Tconv2_1)
```

```
        up1 = self.transpose1(uprel2_1)
        skip_con1 = self.skip_connection1(rel2)
        up1 = torch.cat([up1, skip_con1], dim=1)
        upconv1_3 = self.upconv1_3(up1)
        uprel1_3 = F.relu(upconv1_3)
        upconv1_2 = self.upconv1_2(uprel1_3)
        uprel1_2 = F.relu(upconv1_2)
        upconv1_1 = self.upconv1_1(uprel1_2)
        output = F.relu(upconv1_1)
        return output

class DataSet(Dataset):
    def __init__(self, folder_path, transform=None):
        self.folder_path = folder_path
        self.transform = transform
        self.images = [os.path.join(folder_path, filename) for filename in os.
 ↪listdir(folder_path)]

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

    def __getitem__(self, idx):
        image_path = self.images[idx]
        image = Image.open(image_path).convert('L')  # Convert image to␣
 ↪grayscale

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

        return image

def mse(image1, image2):
    """Calculate the Mean Squared Error (MSE) between two images."""
    return ((image1 - image2) ** 2).mean()

def PSNR(MSE,MAX):# my implementation according to documentations
    return 20*np.log10(MAX) - 10*np.log10(MSE)
```

[3]:
```
if torch.cuda.is_available():
    device = torch.device("cuda")          # a CUDA device object
    print(f"Using CUDA device: {torch.cuda.get_device_name(0)}")
else:
    device = torch.device("cpu")
    print("CUDA is not available. Using CPU instead.")
```

```
training_images_folder = "Train400"
testing_images_folder = "test68"


transformer = transforms.Compose([
    transforms.Resize((256,256)),
    transforms.ToTensor(),
])

training_dataset = DataSet(training_images_folder,transform = transformer)



print("First tensor after transform:")
print(training_dataset[0])
print("First tensor shape: ")
print(training_dataset[0].shape)
min_value = training_dataset[0].min()
max_value = training_dataset[0].max()
print(f'Min Pixel: {min_value}, Max Pixel: {max_value}')

print("\nFirst Training Image after preprocessing: ")
image_pil = TF.to_pil_image(training_dataset[0])
display(image_pil)
```

```
Using CUDA device: NVIDIA GeForce RTX 4060 Ti
First tensor after transform:
tensor([[[0.2510, 0.2588, 0.2667,  …, 0.3373, 0.3490, 0.3569],
         [0.2745, 0.2824, 0.2863,  …, 0.3294, 0.3373, 0.3451],
         [0.3020, 0.3020, 0.3020,  …, 0.3137, 0.3137, 0.3176],
         …,
         [0.2157, 0.2157, 0.2275,  …, 0.4863, 0.4314, 0.3804],
         [0.2078, 0.2000, 0.2078,  …, 0.4627, 0.4039, 0.3451],
         [0.2000, 0.1882, 0.1843,  …, 0.4431, 0.3922, 0.3373]]])
First tensor shape:
torch.Size([1, 256, 256])
Min Pixel: 0.09803921729326248, Max Pixel: 0.8745098114013672

First Training Image after preprocessing:
```

[4]:
```
#trainSize = 360, validationSize = 40, testSize = 68

testD = DataSet(testing_images_folder, transform = transformer)
trainD, valD = random_split(training_dataset, [int(0.9*len(training_dataset)),␣
 ↪int(0.1*len(training_dataset))])
train_load = DataLoader(trainD, batch_size=15,shuffle=True)
test_load = DataLoader(testD, batch_size=15,shuffle=True)
val_load = DataLoader(valD, batch_size=15,shuffle=True)
```

[5]:
```
#finished pre processing
```

[6]:
```
#for training, we have these variables set up ahead of itme
model = UNet_Denoise().to(device)

dataset_size = 400
learning_rate = 0.001
loss_fn = nn.MSELoss()  # Mean Squared Error loss
optimizer = torch.optim.Adam(model.parameters(), lr=learning_rate)  # Adam␣
 ↪optimizer
epochs = 100
mean = 0
variance = 15/255

#lists for graphing later
```

```
#each value in these lists is for a single epoch
training_loss = []
validation_loss = []
ssim_on = []#we are going to compare ORIGINAL vs NOISY which is going to be bad
ssim_oo = []# versus, ORIGINAL vs OUTPUT, which should be a lot more simikar
psnr_on = []
psnr_oo = []
```

[7]:
```
for ep in range(epochs):
    model.train()

    # Variables to track loss and metrics for each epoch
    train_loss = 0
    val_loss = 0
    ssim_noisy_total = 0
    ssim_output_total = 0
    psnr_noisy_total = 0
    psnr_output_total = 0

    num_batches_train = 0
    num_batches_val = 0
    num_samples_total = 0

    # Training loop
    for batch_idx, images in enumerate(train_load):
        clean_images = images.to(device)
        model_input = clean_images + (torch.randn_like(clean_images)*variance +␣
 ↪mean)

        optimizer.zero_grad()
        outputs = model(model_input)
        loss = loss_fn(outputs, clean_images)
        loss.backward()
        optimizer.step()

        train_loss += loss.item()
        num_batches_train += 1

    # Validation loop
    model.eval()
    with torch.no_grad():
        for batch_idx, images in enumerate(val_load):
            clean_images = images.to(device)
            model_input = clean_images + (torch.randn_like(clean_images) *␣
 ↪variance + mean)
            outputs = model(model_input)
            val_loss += loss_fn(outputs, clean_images).item()
```

```
            num_batches_val += 1

            for original, noisy, output in zip(clean_images, model_input,␣
↪outputs):
                original = original.squeeze().detach().cpu().numpy()  # Extract␣
↪256x256 part
                noisy = noisy.squeeze().detach().cpu().numpy()  # Extract␣
↪256x256 part
                output = output.squeeze().detach().cpu().numpy()  # Extract␣
↪256x256 part

                # Update SSIM and PSNR totals
                ssim_noisy_total += ssim(original, noisy, data_range=1.0)# we␣
↪are on a data_range of [0,1]
                ssim_output_total += ssim(original, output, data_range=1.0)
                psnr_noisy_total += PSNR(mse(original, noisy), original.max())
                psnr_output_total += PSNR(mse(original, output), original.max())
                num_samples_total += 1


    # Calculate averages
    train_loss_avg = train_loss / num_batches_train if num_batches_train > 0␣
↪else 0
    val_loss_avg = val_loss / num_batches_val if num_batches_val > 0 else 0
    ssim_noisy_avg = ssim_noisy_total / num_samples_total if num_samples_total␣
↪> 0 else 0
    ssim_output_avg = ssim_output_total / num_samples_total if␣
↪num_samples_total > 0 else 0
    psnr_noisy_avg = psnr_noisy_total / num_samples_total if num_samples_total␣
↪> 0 else 0
    psnr_output_avg = psnr_output_total / num_samples_total if␣
↪num_samples_total > 0 else 0

    training_loss.append(train_loss_avg)# documenting for graphing; each␣
↪element represents an epoch of data
    validation_loss.append(val_loss_avg)
    ssim_on.append(ssim_noisy_avg)
    ssim_oo.append(ssim_output_avg)
    psnr_on.append(psnr_noisy_avg)
    psnr_oo.append(psnr_output_avg)

    # Print results
    print(f"Epoch {ep+1}/{epochs}, Training Loss: {train_loss_avg:.4f},␣
↪Validation Loss: {val_loss_avg:.4f},"
          f" Avg SSIM of noisy images: {ssim_noisy_avg:.4f}, Avg SSIM of output␣
↪images: {ssim_output_avg:.4f},"
```

```
        f" Avg PSNR of noisy images: {psnr_noisy_avg:.4f}, Avg PSNR of output␣
↪images: {psnr_output_avg:.4f}")
```

```
Epoch 1/100, Training Loss: 0.1906, Validation Loss: 0.1180, Avg SSIM of noisy
images: 0.5373, Avg SSIM of output images: 0.3467, Avg PSNR of noisy images:
24.1060, Avg PSNR of output images: 9.6077
Epoch 2/100, Training Loss: 0.0453, Validation Loss: 0.0085, Avg SSIM of noisy
images: 0.5372, Avg SSIM of output images: 0.7670, Avg PSNR of noisy images:
24.0999, Avg PSNR of output images: 20.4079
Epoch 3/100, Training Loss: 0.0045, Validation Loss: 0.0032, Avg SSIM of noisy
images: 0.5369, Avg SSIM of output images: 0.8022, Avg PSNR of noisy images:
24.0976, Avg PSNR of output images: 24.8643
Epoch 4/100, Training Loss: 0.0029, Validation Loss: 0.0026, Avg SSIM of noisy
images: 0.5370, Avg SSIM of output images: 0.8102, Avg PSNR of noisy images:
24.1012, Avg PSNR of output images: 25.6617
Epoch 5/100, Training Loss: 0.0025, Validation Loss: 0.0022, Avg SSIM of noisy
images: 0.5372, Avg SSIM of output images: 0.8312, Avg PSNR of noisy images:
24.1029, Avg PSNR of output images: 26.5169
Epoch 6/100, Training Loss: 0.0020, Validation Loss: 0.0018, Avg SSIM of noisy
images: 0.5373, Avg SSIM of output images: 0.8436, Avg PSNR of noisy images:
24.1063, Avg PSNR of output images: 27.3162
Epoch 7/100, Training Loss: 0.0017, Validation Loss: 0.0016, Avg SSIM of noisy
images: 0.5370, Avg SSIM of output images: 0.8482, Avg PSNR of noisy images:
24.1022, Avg PSNR of output images: 27.8109
Epoch 8/100, Training Loss: 0.0017, Validation Loss: 0.0023, Avg SSIM of noisy
images: 0.5374, Avg SSIM of output images: 0.8463, Avg PSNR of noisy images:
24.1035, Avg PSNR of output images: 26.1679
Epoch 9/100, Training Loss: 0.0025, Validation Loss: 0.0017, Avg SSIM of noisy
images: 0.5372, Avg SSIM of output images: 0.8490, Avg PSNR of noisy images:
24.1047, Avg PSNR of output images: 27.5142
Epoch 10/100, Training Loss: 0.0017, Validation Loss: 0.0015, Avg SSIM of noisy
images: 0.5372, Avg SSIM of output images: 0.8506, Avg PSNR of noisy images:
24.1032, Avg PSNR of output images: 27.9605
Epoch 11/100, Training Loss: 0.0015, Validation Loss: 0.0013, Avg SSIM of noisy
images: 0.5374, Avg SSIM of output images: 0.8487, Avg PSNR of noisy images:
24.1047, Avg PSNR of output images: 28.7166
Epoch 12/100, Training Loss: 0.0012, Validation Loss: 0.0010, Avg SSIM of noisy
images: 0.5376, Avg SSIM of output images: 0.8502, Avg PSNR of noisy images:
24.1109, Avg PSNR of output images: 29.7544
Epoch 13/100, Training Loss: 0.0009, Validation Loss: 0.0009, Avg SSIM of noisy
images: 0.5374, Avg SSIM of output images: 0.8518, Avg PSNR of noisy images:
24.1035, Avg PSNR of output images: 30.4356
Epoch 14/100, Training Loss: 0.0008, Validation Loss: 0.0008, Avg SSIM of noisy
images: 0.5374, Avg SSIM of output images: 0.8520, Avg PSNR of noisy images:
24.1059, Avg PSNR of output images: 30.6845
Epoch 15/100, Training Loss: 0.0008, Validation Loss: 0.0007, Avg SSIM of noisy
images: 0.5373, Avg SSIM of output images: 0.8567, Avg PSNR of noisy images:
24.1060, Avg PSNR of output images: 30.9221
```

```
Epoch 16/100, Training Loss: 0.0007, Validation Loss: 0.0007, Avg SSIM of noisy
images: 0.5368, Avg SSIM of output images: 0.8665, Avg PSNR of noisy images:
24.0978, Avg PSNR of output images: 31.2542
Epoch 17/100, Training Loss: 0.0007, Validation Loss: 0.0007, Avg SSIM of noisy
images: 0.5376, Avg SSIM of output images: 0.8727, Avg PSNR of noisy images:
24.1073, Avg PSNR of output images: 31.5166
Epoch 18/100, Training Loss: 0.0007, Validation Loss: 0.0006, Avg SSIM of noisy
images: 0.5374, Avg SSIM of output images: 0.8764, Avg PSNR of noisy images:
24.1022, Avg PSNR of output images: 31.6506
Epoch 19/100, Training Loss: 0.0006, Validation Loss: 0.0006, Avg SSIM of noisy
images: 0.5373, Avg SSIM of output images: 0.8768, Avg PSNR of noisy images:
24.1063, Avg PSNR of output images: 31.6968
Epoch 20/100, Training Loss: 0.0006, Validation Loss: 0.0006, Avg SSIM of noisy
images: 0.5371, Avg SSIM of output images: 0.8802, Avg PSNR of noisy images:
24.1054, Avg PSNR of output images: 31.7726
Epoch 21/100, Training Loss: 0.0006, Validation Loss: 0.0006, Avg SSIM of noisy
images: 0.5373, Avg SSIM of output images: 0.8838, Avg PSNR of noisy images:
24.1056, Avg PSNR of output images: 31.8874
Epoch 22/100, Training Loss: 0.0006, Validation Loss: 0.0006, Avg SSIM of noisy
images: 0.5373, Avg SSIM of output images: 0.8865, Avg PSNR of noisy images:
24.1082, Avg PSNR of output images: 31.9799
Epoch 23/100, Training Loss: 0.0006, Validation Loss: 0.0006, Avg SSIM of noisy
images: 0.5371, Avg SSIM of output images: 0.8888, Avg PSNR of noisy images:
24.1003, Avg PSNR of output images: 32.0302
Epoch 24/100, Training Loss: 0.0006, Validation Loss: 0.0006, Avg SSIM of noisy
images: 0.5376, Avg SSIM of output images: 0.8902, Avg PSNR of noisy images:
24.1090, Avg PSNR of output images: 31.7698
Epoch 25/100, Training Loss: 0.0006, Validation Loss: 0.0006, Avg SSIM of noisy
images: 0.5371, Avg SSIM of output images: 0.8912, Avg PSNR of noisy images:
24.1014, Avg PSNR of output images: 32.1144
Epoch 26/100, Training Loss: 0.0006, Validation Loss: 0.0006, Avg SSIM of noisy
images: 0.5370, Avg SSIM of output images: 0.8902, Avg PSNR of noisy images:
24.0976, Avg PSNR of output images: 32.0555
Epoch 27/100, Training Loss: 0.0006, Validation Loss: 0.0006, Avg SSIM of noisy
images: 0.5372, Avg SSIM of output images: 0.8955, Avg PSNR of noisy images:
24.1054, Avg PSNR of output images: 32.2278
Epoch 28/100, Training Loss: 0.0006, Validation Loss: 0.0006, Avg SSIM of noisy
images: 0.5374, Avg SSIM of output images: 0.8962, Avg PSNR of noisy images:
24.1021, Avg PSNR of output images: 32.2289
Epoch 29/100, Training Loss: 0.0006, Validation Loss: 0.0006, Avg SSIM of noisy
images: 0.5370, Avg SSIM of output images: 0.8955, Avg PSNR of noisy images:
24.1006, Avg PSNR of output images: 32.2273
Epoch 30/100, Training Loss: 0.0006, Validation Loss: 0.0006, Avg SSIM of noisy
images: 0.5371, Avg SSIM of output images: 0.8961, Avg PSNR of noisy images:
24.1018, Avg PSNR of output images: 32.2879
Epoch 31/100, Training Loss: 0.0006, Validation Loss: 0.0006, Avg SSIM of noisy
images: 0.5372, Avg SSIM of output images: 0.8988, Avg PSNR of noisy images:
24.1045, Avg PSNR of output images: 32.3540
```

```
Epoch 32/100, Training Loss: 0.0005, Validation Loss: 0.0005, Avg SSIM of noisy
images: 0.5371, Avg SSIM of output images: 0.9007, Avg PSNR of noisy images:
24.1012, Avg PSNR of output images: 32.4285
Epoch 33/100, Training Loss: 0.0005, Validation Loss: 0.0005, Avg SSIM of noisy
images: 0.5369, Avg SSIM of output images: 0.9014, Avg PSNR of noisy images:
24.0958, Avg PSNR of output images: 32.4627
Epoch 34/100, Training Loss: 0.0005, Validation Loss: 0.0005, Avg SSIM of noisy
images: 0.5371, Avg SSIM of output images: 0.9018, Avg PSNR of noisy images:
24.0965, Avg PSNR of output images: 32.4355
Epoch 35/100, Training Loss: 0.0005, Validation Loss: 0.0005, Avg SSIM of noisy
images: 0.5374, Avg SSIM of output images: 0.9031, Avg PSNR of noisy images:
24.1075, Avg PSNR of output images: 32.4880
Epoch 36/100, Training Loss: 0.0005, Validation Loss: 0.0006, Avg SSIM of noisy
images: 0.5371, Avg SSIM of output images: 0.8996, Avg PSNR of noisy images:
24.1037, Avg PSNR of output images: 32.4093
Epoch 37/100, Training Loss: 0.0005, Validation Loss: 0.0005, Avg SSIM of noisy
images: 0.5371, Avg SSIM of output images: 0.9030, Avg PSNR of noisy images:
24.0991, Avg PSNR of output images: 32.4828
Epoch 38/100, Training Loss: 0.0005, Validation Loss: 0.0006, Avg SSIM of noisy
images: 0.5369, Avg SSIM of output images: 0.9006, Avg PSNR of noisy images:
24.0986, Avg PSNR of output images: 32.3598
Epoch 39/100, Training Loss: 0.0005, Validation Loss: 0.0005, Avg SSIM of noisy
images: 0.5369, Avg SSIM of output images: 0.9051, Avg PSNR of noisy images:
24.0977, Avg PSNR of output images: 32.5759
Epoch 40/100, Training Loss: 0.0005, Validation Loss: 0.0005, Avg SSIM of noisy
images: 0.5371, Avg SSIM of output images: 0.9041, Avg PSNR of noisy images:
24.1032, Avg PSNR of output images: 32.5010
Epoch 41/100, Training Loss: 0.0005, Validation Loss: 0.0005, Avg SSIM of noisy
images: 0.5372, Avg SSIM of output images: 0.9044, Avg PSNR of noisy images:
24.1073, Avg PSNR of output images: 32.5688
Epoch 42/100, Training Loss: 0.0005, Validation Loss: 0.0005, Avg SSIM of noisy
images: 0.5368, Avg SSIM of output images: 0.9058, Avg PSNR of noisy images:
24.0936, Avg PSNR of output images: 32.5934
Epoch 43/100, Training Loss: 0.0005, Validation Loss: 0.0005, Avg SSIM of noisy
images: 0.5371, Avg SSIM of output images: 0.9044, Avg PSNR of noisy images:
24.1013, Avg PSNR of output images: 32.5967
Epoch 44/100, Training Loss: 0.0005, Validation Loss: 0.0005, Avg SSIM of noisy
images: 0.5372, Avg SSIM of output images: 0.9050, Avg PSNR of noisy images:
24.1083, Avg PSNR of output images: 32.6250
Epoch 45/100, Training Loss: 0.0005, Validation Loss: 0.0005, Avg SSIM of noisy
images: 0.5373, Avg SSIM of output images: 0.9046, Avg PSNR of noisy images:
24.1044, Avg PSNR of output images: 32.5913
Epoch 46/100, Training Loss: 0.0005, Validation Loss: 0.0005, Avg SSIM of noisy
images: 0.5370, Avg SSIM of output images: 0.9063, Avg PSNR of noisy images:
24.0988, Avg PSNR of output images: 32.5956
Epoch 47/100, Training Loss: 0.0005, Validation Loss: 0.0005, Avg SSIM of noisy
images: 0.5373, Avg SSIM of output images: 0.9089, Avg PSNR of noisy images:
24.1039, Avg PSNR of output images: 32.6916
```

```
Epoch 48/100, Training Loss: 0.0005, Validation Loss: 0.0005, Avg SSIM of noisy
images: 0.5366, Avg SSIM of output images: 0.9055, Avg PSNR of noisy images:
24.0925, Avg PSNR of output images: 32.5834
Epoch 49/100, Training Loss: 0.0005, Validation Loss: 0.0005, Avg SSIM of noisy
images: 0.5372, Avg SSIM of output images: 0.9096, Avg PSNR of noisy images:
24.1041, Avg PSNR of output images: 32.7337
Epoch 50/100, Training Loss: 0.0005, Validation Loss: 0.0005, Avg SSIM of noisy
images: 0.5373, Avg SSIM of output images: 0.9088, Avg PSNR of noisy images:
24.1071, Avg PSNR of output images: 32.7263
Epoch 51/100, Training Loss: 0.0005, Validation Loss: 0.0005, Avg SSIM of noisy
images: 0.5372, Avg SSIM of output images: 0.9104, Avg PSNR of noisy images:
24.1043, Avg PSNR of output images: 32.7523
Epoch 52/100, Training Loss: 0.0005, Validation Loss: 0.0005, Avg SSIM of noisy
images: 0.5372, Avg SSIM of output images: 0.9094, Avg PSNR of noisy images:
24.1015, Avg PSNR of output images: 32.6374
Epoch 53/100, Training Loss: 0.0005, Validation Loss: 0.0005, Avg SSIM of noisy
images: 0.5375, Avg SSIM of output images: 0.9092, Avg PSNR of noisy images:
24.1094, Avg PSNR of output images: 32.6697
Epoch 54/100, Training Loss: 0.0005, Validation Loss: 0.0005, Avg SSIM of noisy
images: 0.5369, Avg SSIM of output images: 0.9074, Avg PSNR of noisy images:
24.0960, Avg PSNR of output images: 32.6131
Epoch 55/100, Training Loss: 0.0005, Validation Loss: 0.0005, Avg SSIM of noisy
images: 0.5372, Avg SSIM of output images: 0.9104, Avg PSNR of noisy images:
24.1010, Avg PSNR of output images: 32.7875
Epoch 56/100, Training Loss: 0.0005, Validation Loss: 0.0005, Avg SSIM of noisy
images: 0.5371, Avg SSIM of output images: 0.9102, Avg PSNR of noisy images:
24.1027, Avg PSNR of output images: 32.7540
Epoch 57/100, Training Loss: 0.0005, Validation Loss: 0.0005, Avg SSIM of noisy
images: 0.5371, Avg SSIM of output images: 0.9090, Avg PSNR of noisy images:
24.1037, Avg PSNR of output images: 32.7575
Epoch 58/100, Training Loss: 0.0005, Validation Loss: 0.0005, Avg SSIM of noisy
images: 0.5370, Avg SSIM of output images: 0.9123, Avg PSNR of noisy images:
24.1002, Avg PSNR of output images: 32.8027
Epoch 59/100, Training Loss: 0.0005, Validation Loss: 0.0005, Avg SSIM of noisy
images: 0.5371, Avg SSIM of output images: 0.9098, Avg PSNR of noisy images:
24.1022, Avg PSNR of output images: 32.7156
Epoch 60/100, Training Loss: 0.0005, Validation Loss: 0.0005, Avg SSIM of noisy
images: 0.5372, Avg SSIM of output images: 0.9124, Avg PSNR of noisy images:
24.1021, Avg PSNR of output images: 32.8604
Epoch 61/100, Training Loss: 0.0005, Validation Loss: 0.0005, Avg SSIM of noisy
images: 0.5370, Avg SSIM of output images: 0.9135, Avg PSNR of noisy images:
24.1023, Avg PSNR of output images: 32.8562
Epoch 62/100, Training Loss: 0.0005, Validation Loss: 0.0005, Avg SSIM of noisy
images: 0.5371, Avg SSIM of output images: 0.9110, Avg PSNR of noisy images:
24.1011, Avg PSNR of output images: 32.7753
Epoch 63/100, Training Loss: 0.0005, Validation Loss: 0.0005, Avg SSIM of noisy
images: 0.5373, Avg SSIM of output images: 0.9126, Avg PSNR of noisy images:
24.1016, Avg PSNR of output images: 32.8251
```

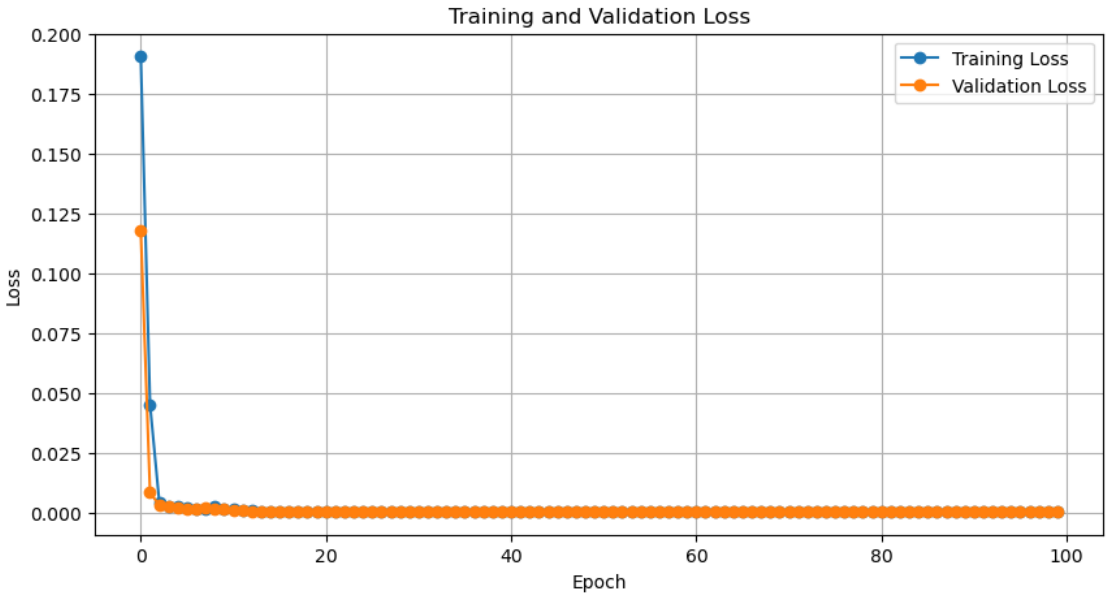
```
Epoch 64/100, Training Loss: 0.0005, Validation Loss: 0.0005, Avg SSIM of noisy
images: 0.5372, Avg SSIM of output images: 0.9132, Avg PSNR of noisy images:
24.1016, Avg PSNR of output images: 32.8742
Epoch 65/100, Training Loss: 0.0005, Validation Loss: 0.0005, Avg SSIM of noisy
images: 0.5372, Avg SSIM of output images: 0.9121, Avg PSNR of noisy images:
24.1085, Avg PSNR of output images: 32.8784
Epoch 66/100, Training Loss: 0.0005, Validation Loss: 0.0005, Avg SSIM of noisy
images: 0.5368, Avg SSIM of output images: 0.9064, Avg PSNR of noisy images:
24.1014, Avg PSNR of output images: 32.6531
Epoch 67/100, Training Loss: 0.0005, Validation Loss: 0.0005, Avg SSIM of noisy
images: 0.5372, Avg SSIM of output images: 0.9100, Avg PSNR of noisy images:
24.1050, Avg PSNR of output images: 32.6588
Epoch 68/100, Training Loss: 0.0005, Validation Loss: 0.0005, Avg SSIM of noisy
images: 0.5372, Avg SSIM of output images: 0.9116, Avg PSNR of noisy images:
24.1020, Avg PSNR of output images: 32.8381
Epoch 69/100, Training Loss: 0.0005, Validation Loss: 0.0005, Avg SSIM of noisy
images: 0.5370, Avg SSIM of output images: 0.9145, Avg PSNR of noisy images:
24.1013, Avg PSNR of output images: 32.8899
Epoch 70/100, Training Loss: 0.0005, Validation Loss: 0.0005, Avg SSIM of noisy
images: 0.5367, Avg SSIM of output images: 0.9145, Avg PSNR of noisy images:
24.0978, Avg PSNR of output images: 32.9409
Epoch 71/100, Training Loss: 0.0005, Validation Loss: 0.0005, Avg SSIM of noisy
images: 0.5375, Avg SSIM of output images: 0.9141, Avg PSNR of noisy images:
24.1117, Avg PSNR of output images: 32.9033
Epoch 72/100, Training Loss: 0.0005, Validation Loss: 0.0005, Avg SSIM of noisy
images: 0.5373, Avg SSIM of output images: 0.9157, Avg PSNR of noisy images:
24.1047, Avg PSNR of output images: 32.9638
Epoch 73/100, Training Loss: 0.0005, Validation Loss: 0.0005, Avg SSIM of noisy
images: 0.5372, Avg SSIM of output images: 0.9126, Avg PSNR of noisy images:
24.1035, Avg PSNR of output images: 32.8363
Epoch 74/100, Training Loss: 0.0005, Validation Loss: 0.0005, Avg SSIM of noisy
images: 0.5370, Avg SSIM of output images: 0.9147, Avg PSNR of noisy images:
24.0993, Avg PSNR of output images: 32.9532
Epoch 75/100, Training Loss: 0.0005, Validation Loss: 0.0005, Avg SSIM of noisy
images: 0.5373, Avg SSIM of output images: 0.9125, Avg PSNR of noisy images:
24.1056, Avg PSNR of output images: 32.8010
Epoch 76/100, Training Loss: 0.0005, Validation Loss: 0.0005, Avg SSIM of noisy
images: 0.5369, Avg SSIM of output images: 0.9148, Avg PSNR of noisy images:
24.0980, Avg PSNR of output images: 32.9362
Epoch 77/100, Training Loss: 0.0005, Validation Loss: 0.0005, Avg SSIM of noisy
images: 0.5369, Avg SSIM of output images: 0.9152, Avg PSNR of noisy images:
24.0984, Avg PSNR of output images: 32.8842
Epoch 78/100, Training Loss: 0.0005, Validation Loss: 0.0005, Avg SSIM of noisy
images: 0.5369, Avg SSIM of output images: 0.9151, Avg PSNR of noisy images:
24.1020, Avg PSNR of output images: 32.8918
Epoch 79/100, Training Loss: 0.0005, Validation Loss: 0.0005, Avg SSIM of noisy
images: 0.5372, Avg SSIM of output images: 0.9109, Avg PSNR of noisy images:
24.1062, Avg PSNR of output images: 32.8234
```

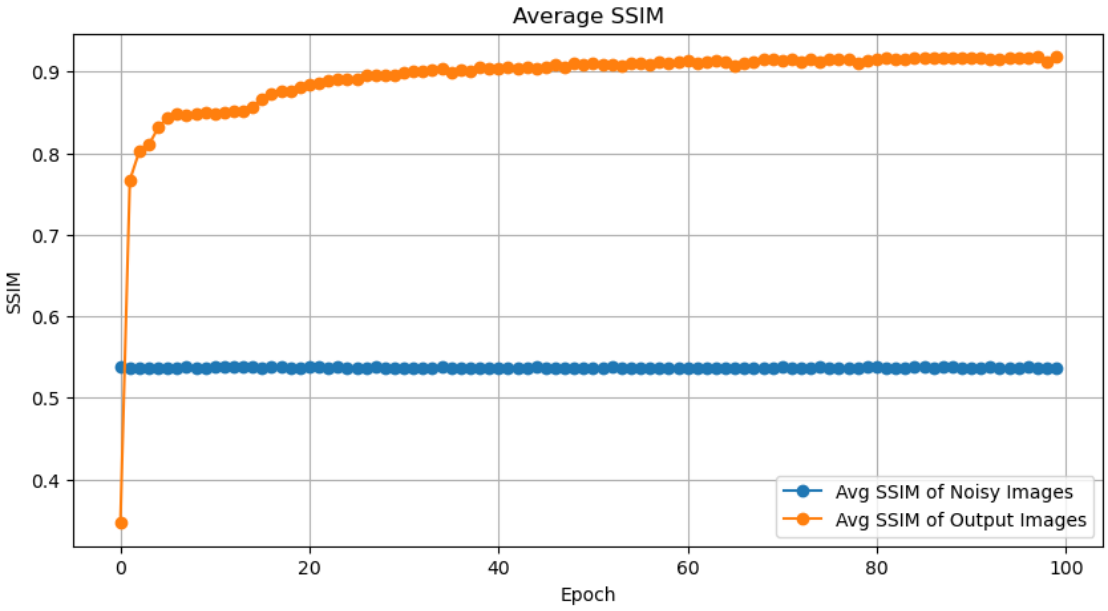
```
Epoch 80/100, Training Loss: 0.0005, Validation Loss: 0.0005, Avg SSIM of noisy
images: 0.5374, Avg SSIM of output images: 0.9136, Avg PSNR of noisy images:
24.1085, Avg PSNR of output images: 32.9274
Epoch 81/100, Training Loss: 0.0005, Validation Loss: 0.0005, Avg SSIM of noisy
images: 0.5373, Avg SSIM of output images: 0.9145, Avg PSNR of noisy images:
24.1038, Avg PSNR of output images: 32.9692
Epoch 82/100, Training Loss: 0.0005, Validation Loss: 0.0005, Avg SSIM of noisy
images: 0.5370, Avg SSIM of output images: 0.9162, Avg PSNR of noisy images:
24.1001, Avg PSNR of output images: 32.8824
Epoch 83/100, Training Loss: 0.0005, Validation Loss: 0.0005, Avg SSIM of noisy
images: 0.5371, Avg SSIM of output images: 0.9156, Avg PSNR of noisy images:
24.0997, Avg PSNR of output images: 33.0045
Epoch 84/100, Training Loss: 0.0005, Validation Loss: 0.0005, Avg SSIM of noisy
images: 0.5370, Avg SSIM of output images: 0.9153, Avg PSNR of noisy images:
24.0992, Avg PSNR of output images: 32.9072
Epoch 85/100, Training Loss: 0.0005, Validation Loss: 0.0005, Avg SSIM of noisy
images: 0.5378, Avg SSIM of output images: 0.9164, Avg PSNR of noisy images:
24.1101, Avg PSNR of output images: 32.9886
Epoch 86/100, Training Loss: 0.0005, Validation Loss: 0.0005, Avg SSIM of noisy
images: 0.5373, Avg SSIM of output images: 0.9166, Avg PSNR of noisy images:
24.1051, Avg PSNR of output images: 32.9954
Epoch 87/100, Training Loss: 0.0005, Validation Loss: 0.0005, Avg SSIM of noisy
images: 0.5372, Avg SSIM of output images: 0.9164, Avg PSNR of noisy images:
24.1013, Avg PSNR of output images: 33.0154
Epoch 88/100, Training Loss: 0.0005, Validation Loss: 0.0005, Avg SSIM of noisy
images: 0.5373, Avg SSIM of output images: 0.9164, Avg PSNR of noisy images:
24.1062, Avg PSNR of output images: 33.0168
Epoch 89/100, Training Loss: 0.0005, Validation Loss: 0.0005, Avg SSIM of noisy
images: 0.5374, Avg SSIM of output images: 0.9172, Avg PSNR of noisy images:
24.1038, Avg PSNR of output images: 33.0061
Epoch 90/100, Training Loss: 0.0005, Validation Loss: 0.0005, Avg SSIM of noisy
images: 0.5367, Avg SSIM of output images: 0.9167, Avg PSNR of noisy images:
24.0953, Avg PSNR of output images: 33.0319
Epoch 91/100, Training Loss: 0.0005, Validation Loss: 0.0005, Avg SSIM of noisy
images: 0.5372, Avg SSIM of output images: 0.9167, Avg PSNR of noisy images:
24.1031, Avg PSNR of output images: 32.9546
Epoch 92/100, Training Loss: 0.0005, Validation Loss: 0.0005, Avg SSIM of noisy
images: 0.5372, Avg SSIM of output images: 0.9164, Avg PSNR of noisy images:
24.1056, Avg PSNR of output images: 33.0373
Epoch 93/100, Training Loss: 0.0005, Validation Loss: 0.0005, Avg SSIM of noisy
images: 0.5375, Avg SSIM of output images: 0.9154, Avg PSNR of noisy images:
24.1082, Avg PSNR of output images: 32.9675
Epoch 94/100, Training Loss: 0.0005, Validation Loss: 0.0005, Avg SSIM of noisy
images: 0.5372, Avg SSIM of output images: 0.9159, Avg PSNR of noisy images:
24.0993, Avg PSNR of output images: 32.9953
Epoch 95/100, Training Loss: 0.0005, Validation Loss: 0.0005, Avg SSIM of noisy
images: 0.5368, Avg SSIM of output images: 0.9170, Avg PSNR of noisy images:
24.0929, Avg PSNR of output images: 33.0289
```

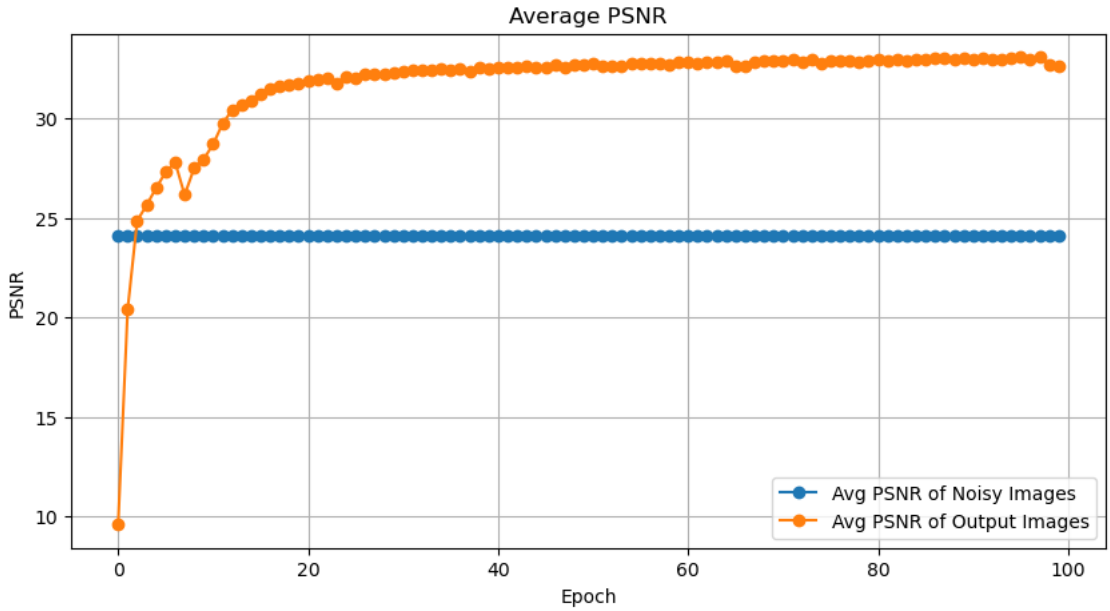
```
Epoch 96/100, Training Loss: 0.0005, Validation Loss: 0.0005, Avg SSIM of noisy
images: 0.5371, Avg SSIM of output images: 0.9169, Avg PSNR of noisy images:
24.1007, Avg PSNR of output images: 33.0842
Epoch 97/100, Training Loss: 0.0005, Validation Loss: 0.0005, Avg SSIM of noisy
images: 0.5373, Avg SSIM of output images: 0.9173, Avg PSNR of noisy images:
24.1026, Avg PSNR of output images: 32.9763
Epoch 98/100, Training Loss: 0.0005, Validation Loss: 0.0005, Avg SSIM of noisy
images: 0.5371, Avg SSIM of output images: 0.9182, Avg PSNR of noisy images:
24.1007, Avg PSNR of output images: 33.0970
Epoch 99/100, Training Loss: 0.0005, Validation Loss: 0.0005, Avg SSIM of noisy
images: 0.5372, Avg SSIM of output images: 0.9128, Avg PSNR of noisy images:
24.1076, Avg PSNR of output images: 32.7073
Epoch 100/100, Training Loss: 0.0005, Validation Loss: 0.0005, Avg SSIM of noisy
images: 0.5371, Avg SSIM of output images: 0.9178, Avg PSNR of noisy images:
24.1030, Avg PSNR of output images: 32.6682
```

[8]:
```python
#graphing for PSNR, SSIM and Loss


# Plot training and validation loss
plt.figure(figsize=(10, 5))
plt.plot(training_loss, label='Training Loss', marker='o')
plt.plot(validation_loss, label='Validation Loss', marker='o')
plt.title('Training and Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.grid(True)
plt.show()

# Plot SSIM
plt.figure(figsize=(10, 5))
plt.plot(ssim_on, label='Avg SSIM of Noisy Images', marker='o')
plt.plot(ssim_oo, label='Avg SSIM of Output Images', marker='o')
plt.title('Average SSIM')
plt.xlabel('Epoch')
plt.ylabel('SSIM')
plt.legend()
plt.grid(True)
plt.show()

# Plot PSNR
plt.figure(figsize=(10, 5))
plt.plot(psnr_on, label='Avg PSNR of Noisy Images', marker='o')
plt.plot(psnr_oo, label='Avg PSNR of Output Images', marker='o')
plt.title('Average PSNR')
plt.xlabel('Epoch')
```

```
plt.ylabel('PSNR')
plt.legend()
plt.grid(True)
plt.show()
```

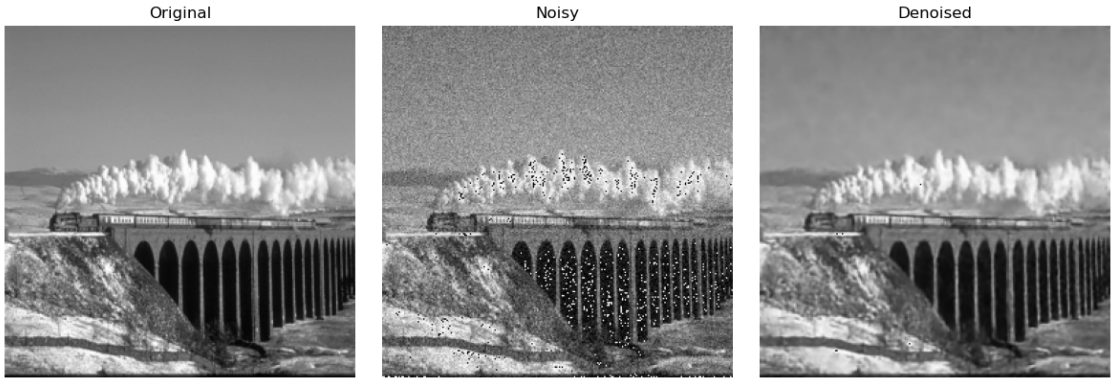






[10]:
```
# testing data
model.eval()
print("\n")

with torch.no_grad():  # Disable gradient computation

    for idx, images in enumerate(test_load):
        clean_images = images.to(device)  # original
        model_input = clean_images + (torch.randn_like(clean_images) * variance␣
↪+ mean)  # noisy image
        outputs = model(model_input)  # denoised image

        for i, (original, noisy, output) in enumerate(zip(clean_images,␣
↪model_input, outputs)):
            test_loss = loss_fn(output, original.to(device))  # Compute loss␣
↪for each image
            original_image = TF.to_pil_image(original.cpu())
            noisy_image = TF.to_pil_image(noisy.cpu())
            output_image = TF.to_pil_image(output.cpu())

            # Create a figure with subplots
            fig, axes = plt.subplots(1, 3, figsize=(12, 4))

            # Display original image
            axes[0].imshow(original_image, cmap='gray')
            axes[0].set_title('Original')
```

```
            axes[0].axis('off')

            # Display noisy image
            axes[1].imshow(noisy_image, cmap='gray')
            axes[1].set_title('Noisy')
            axes[1].axis('off')

            # Display denoised image
            axes[2].imshow(output_image, cmap='gray')
            axes[2].set_title('Denoised')
            axes[2].axis('off')

            # Adjust layout to prevent overlapping
            plt.tight_layout()

            # Show the plot
            plt.show()
            print(f"Image {i+1} in Batch {idx+1}, Test Loss: {test_loss.
↪item()}")
            print("\n")
```




```
Image 1 in Batch 1, Test Loss: 0.0009447195916436613
```

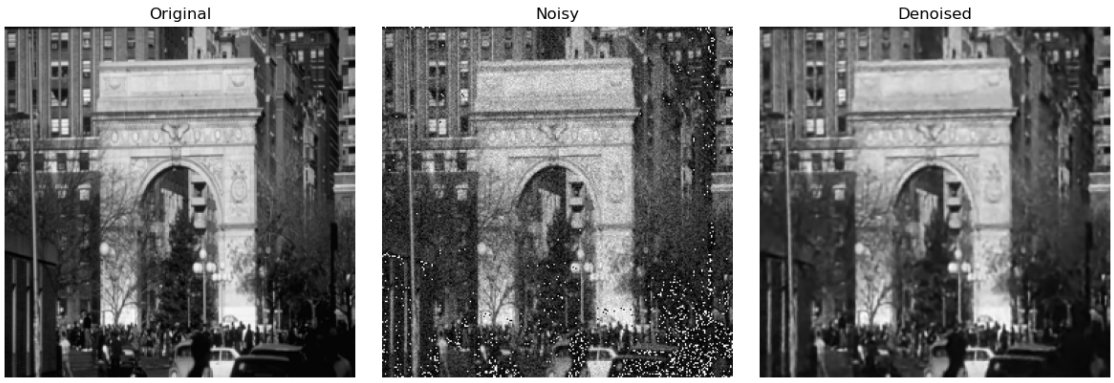


Image 2 in Batch 1, Test Loss: 0.0012510655215010047

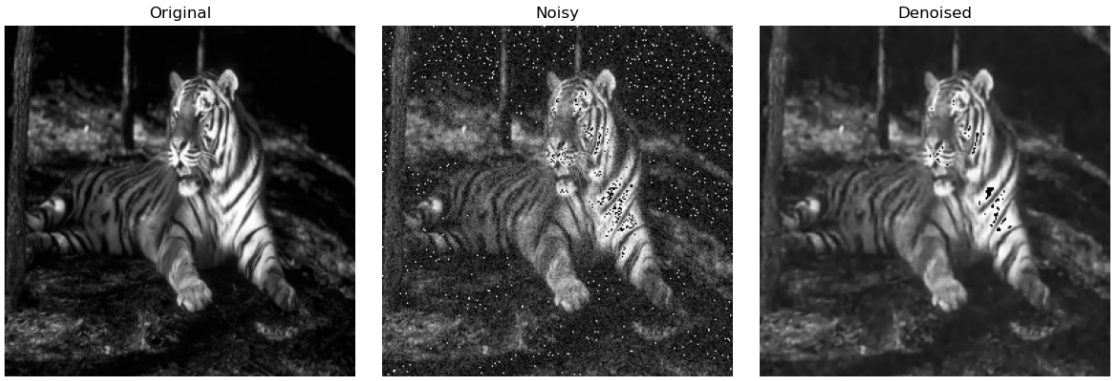


Image 3 in Batch 1, Test Loss: 0.0007775571430101991

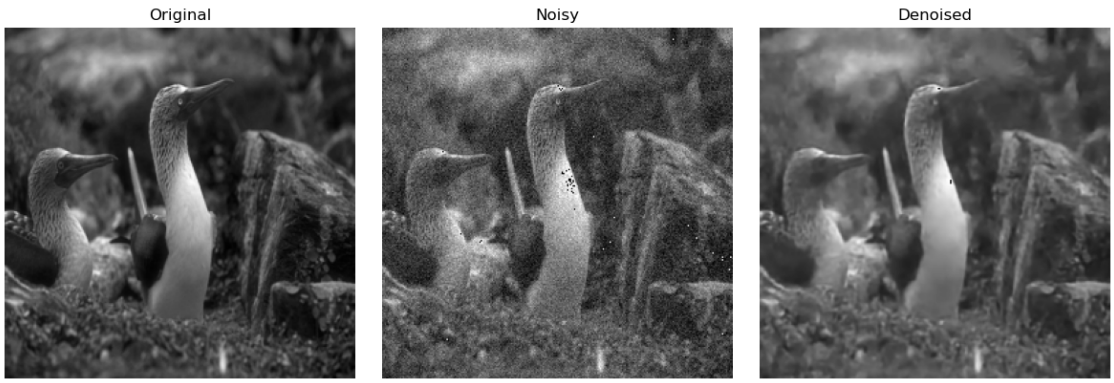


Image 4 in Batch 1, Test Loss: 0.000581097905524075

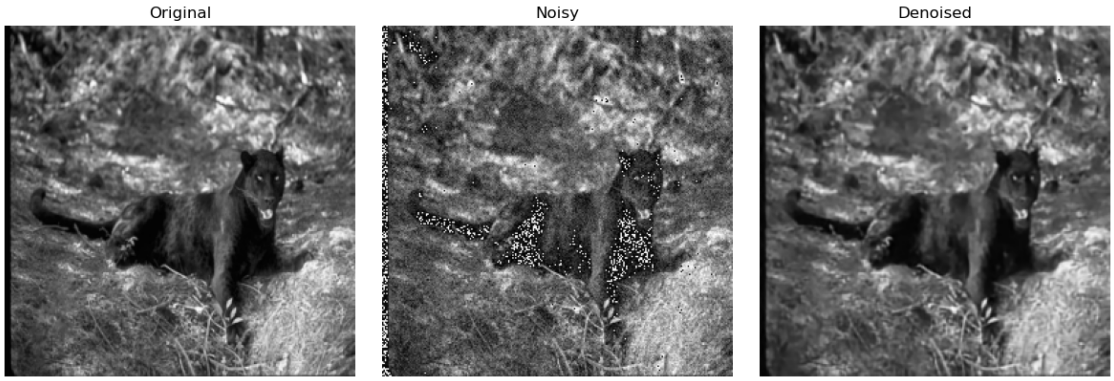


Image 5 in Batch 1, Test Loss: 0.0012937632855027914

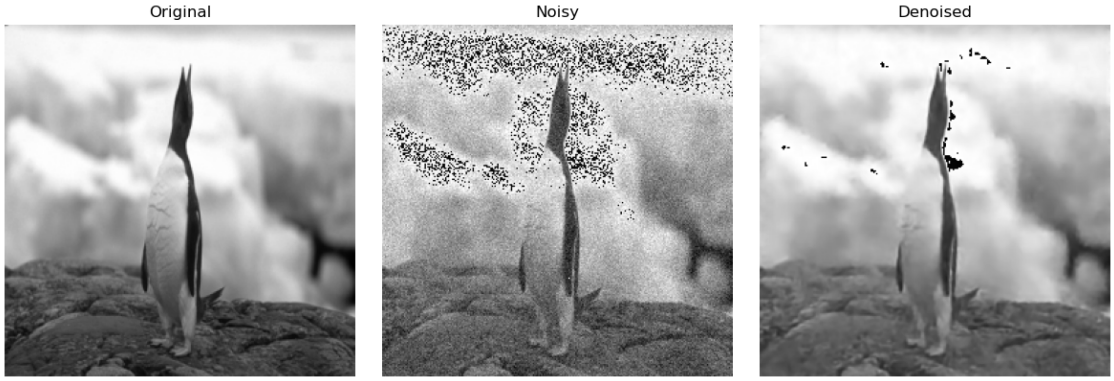


Image 6 in Batch 1, Test Loss: 0.00040622311644256115

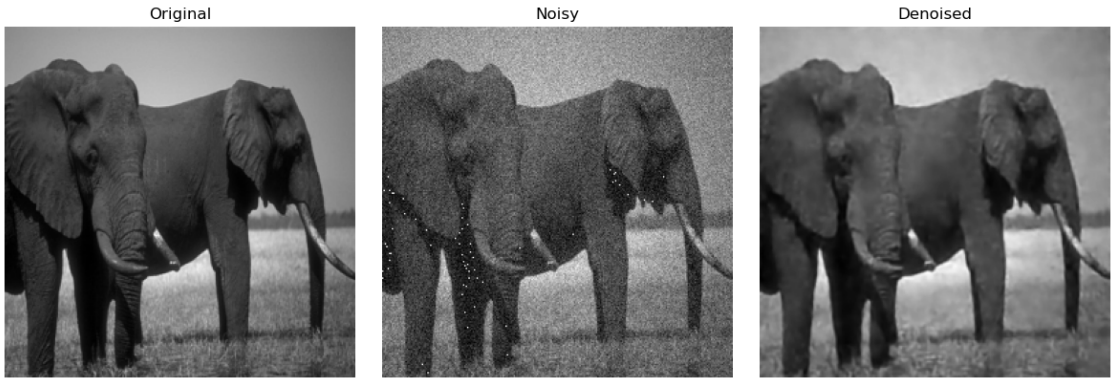


Image 7 in Batch 1, Test Loss: 0.0005104281008243561

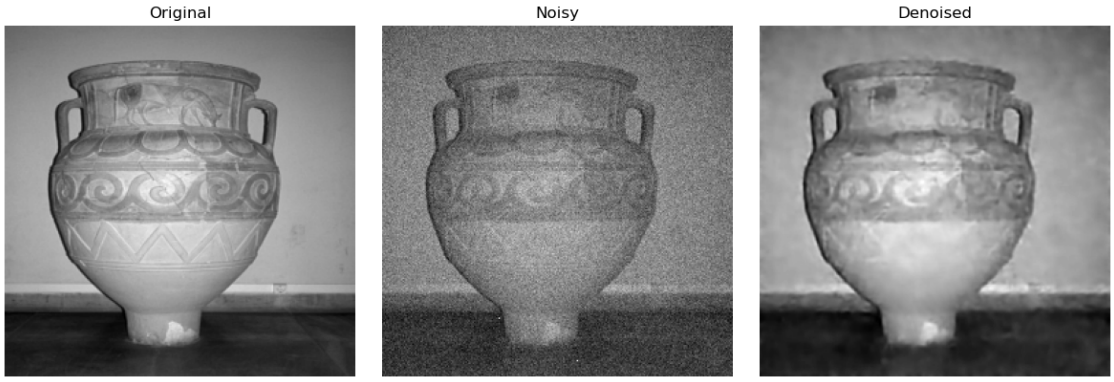


Image 8 in Batch 1, Test Loss: 0.00033817286021076143

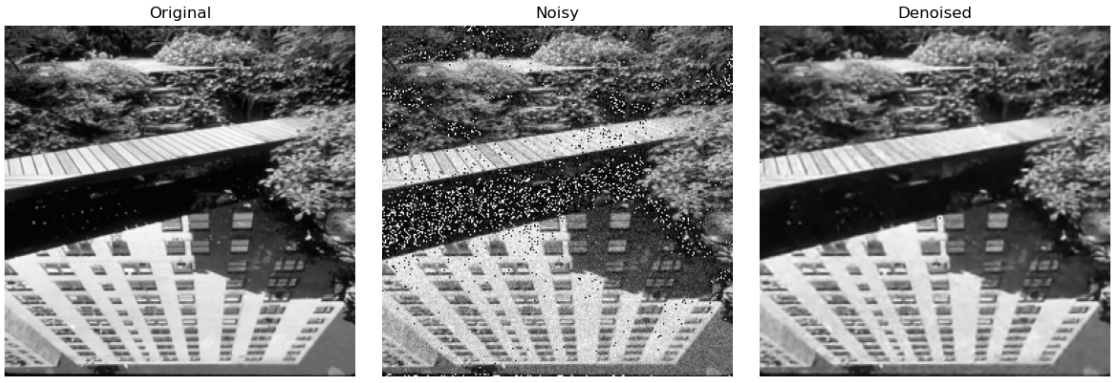


Image 9 in Batch 1, Test Loss: 0.0015757277142256498

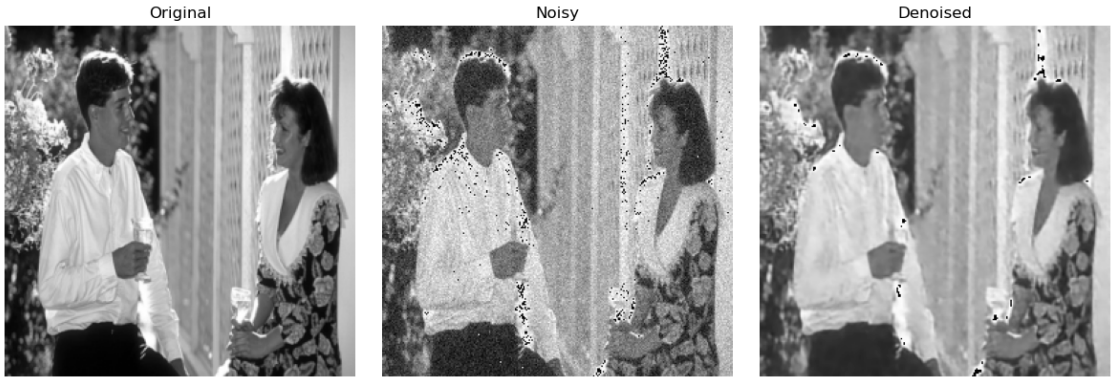


Image 10 in Batch 1, Test Loss: 0.0009379835100844502

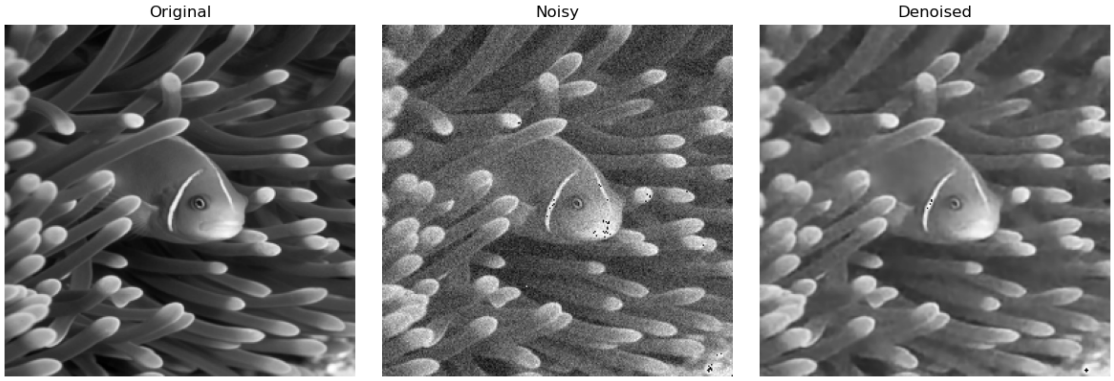


Image 11 in Batch 1, Test Loss: 0.0005113217048346996

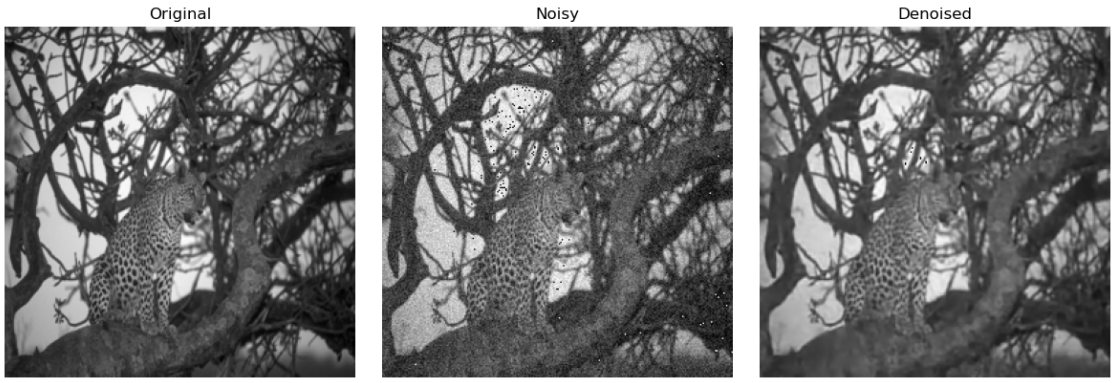


Image 12 in Batch 1, Test Loss: 0.0012131371768191457

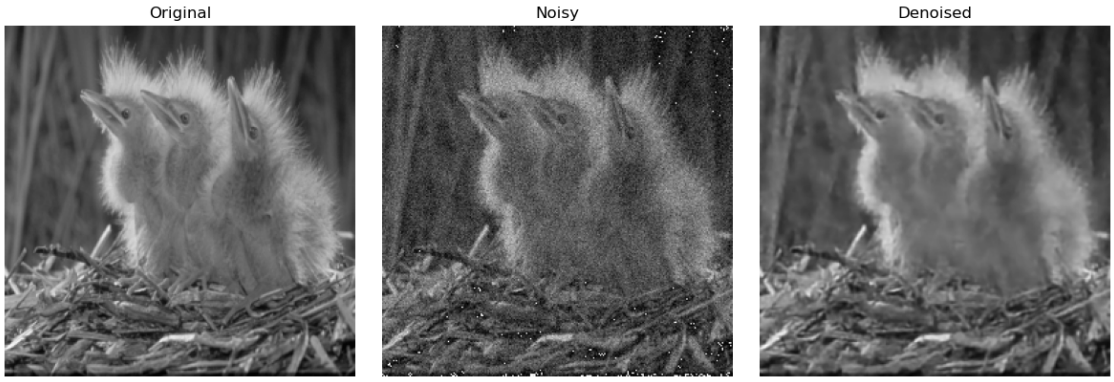


Image 13 in Batch 1, Test Loss: 0.0006418081466108561

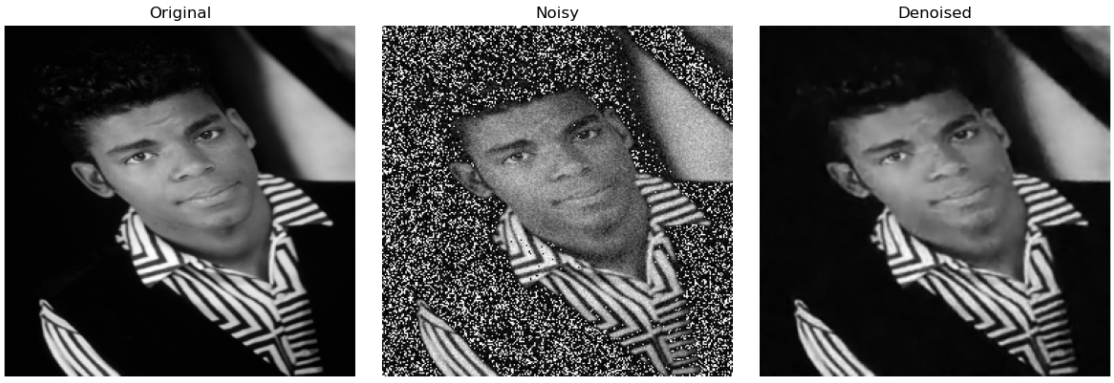


Image 14 in Batch 1, Test Loss: 0.0004344853514339775

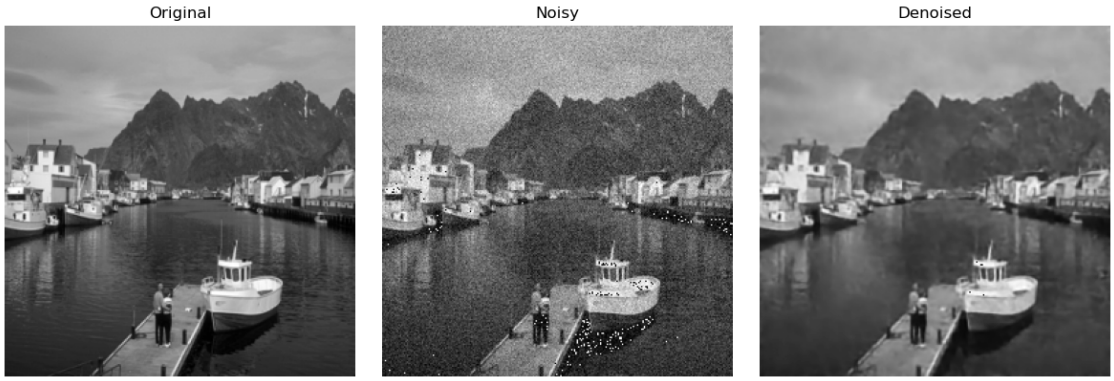


Image 15 in Batch 1, Test Loss: 0.0007312194211408496

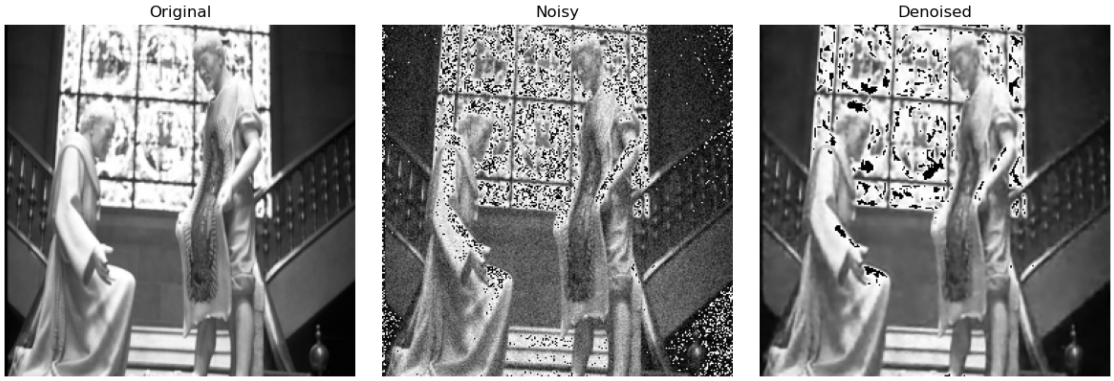


Image 1 in Batch 2, Test Loss: 0.0008299873443320394

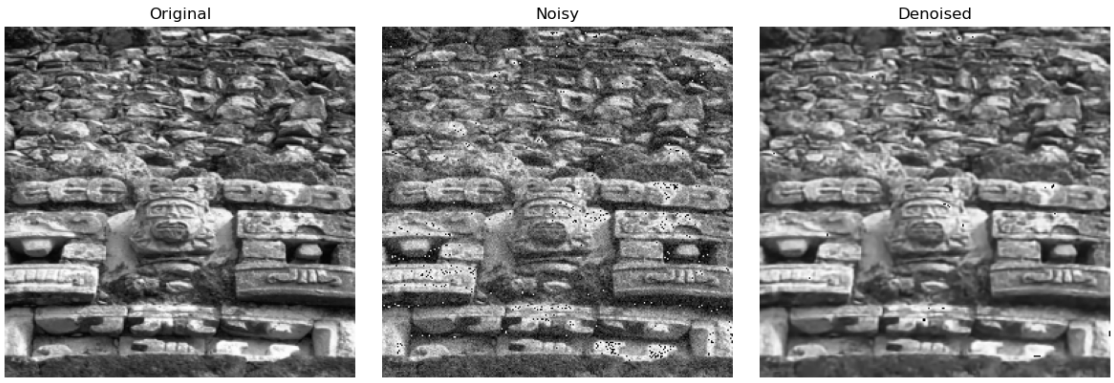


Image 2 in Batch 2, Test Loss: 0.0017535919323563576

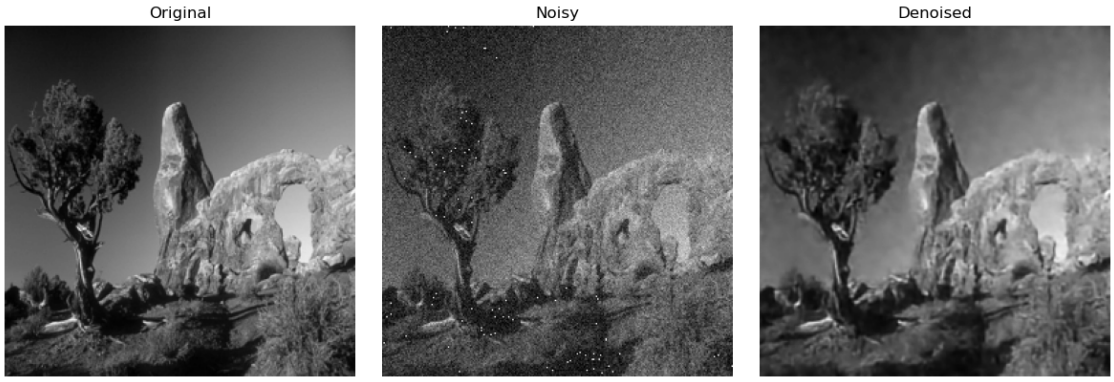


Image 3 in Batch 2, Test Loss: 0.0007451613200828433

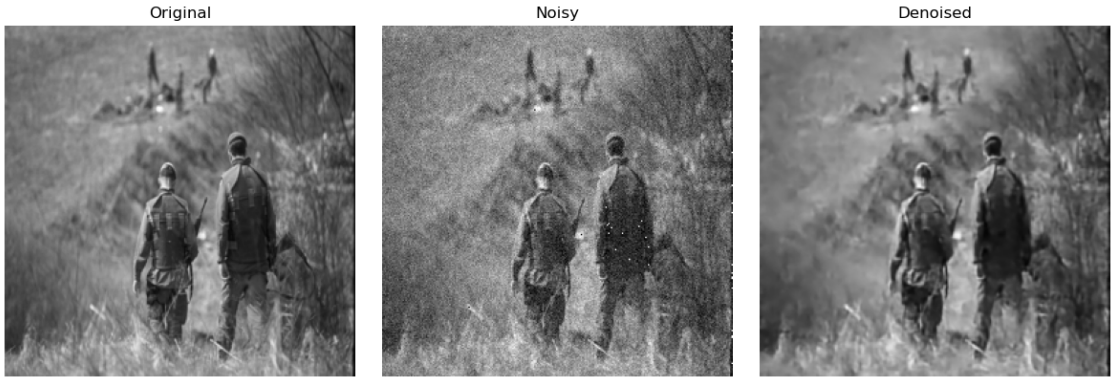


Image 4 in Batch 2, Test Loss: 0.0007074034074321389

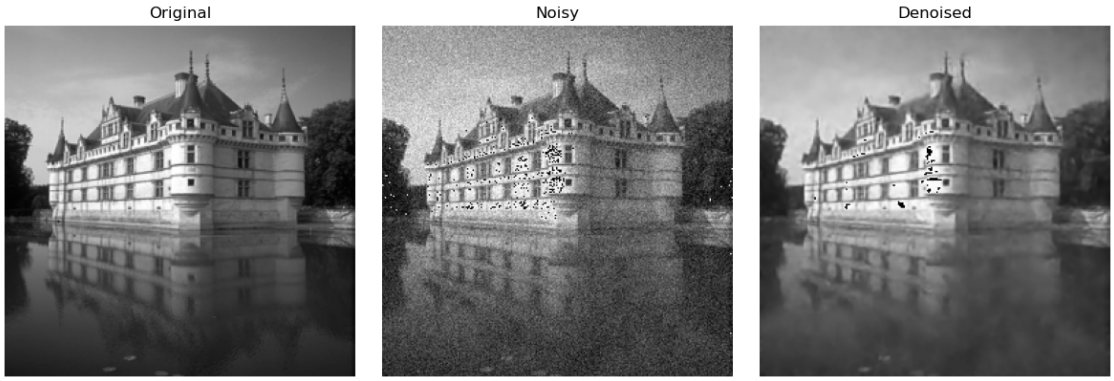


Image 5 in Batch 2, Test Loss: 0.0006250332808122039

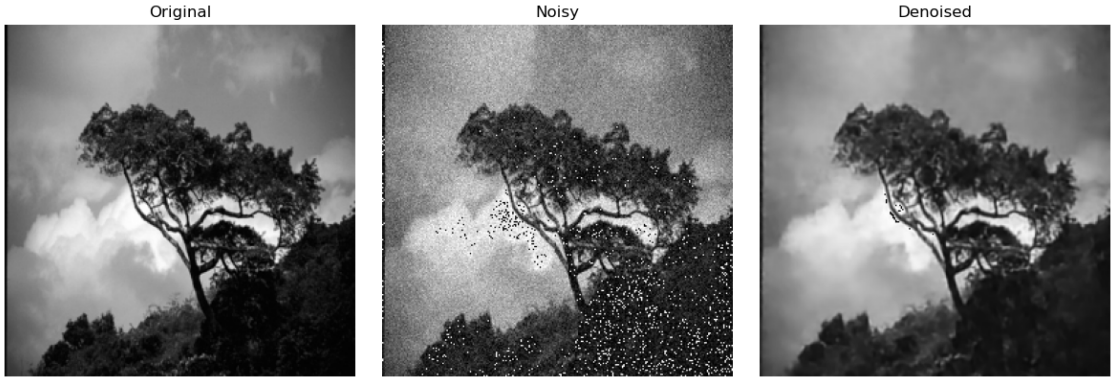


Image 6 in Batch 2, Test Loss: 0.0008123343577608466

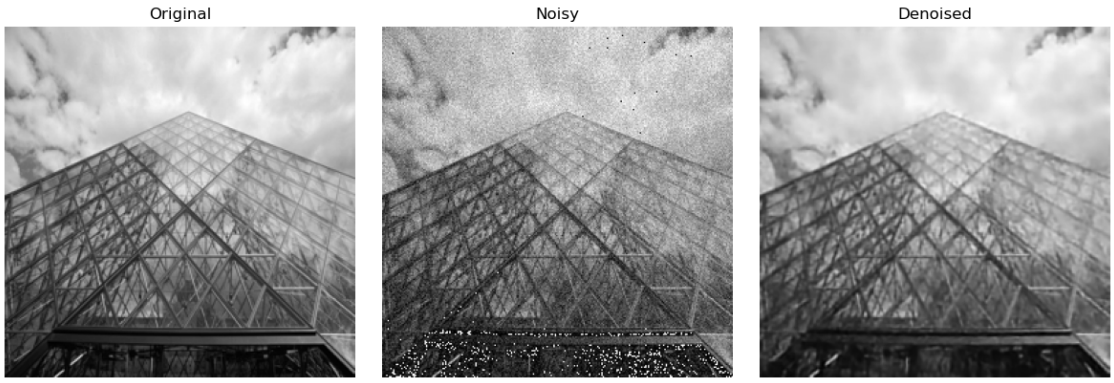


Image 7 in Batch 2, Test Loss: 0.0011590528301894665

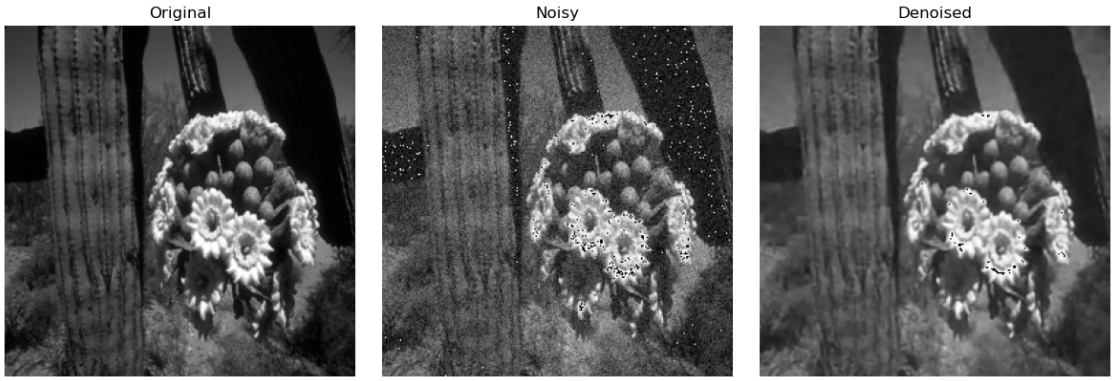


Image 8 in Batch 2, Test Loss: 0.0008835093467496336

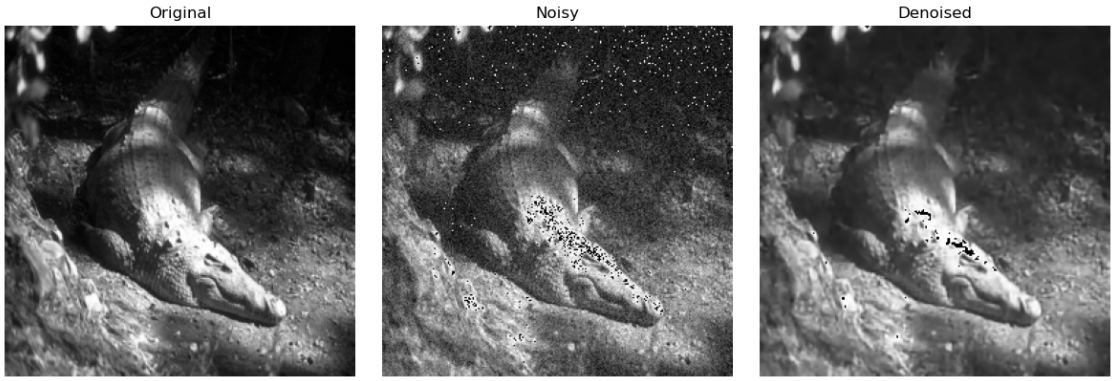


Image 9 in Batch 2, Test Loss: 0.001111391349695623

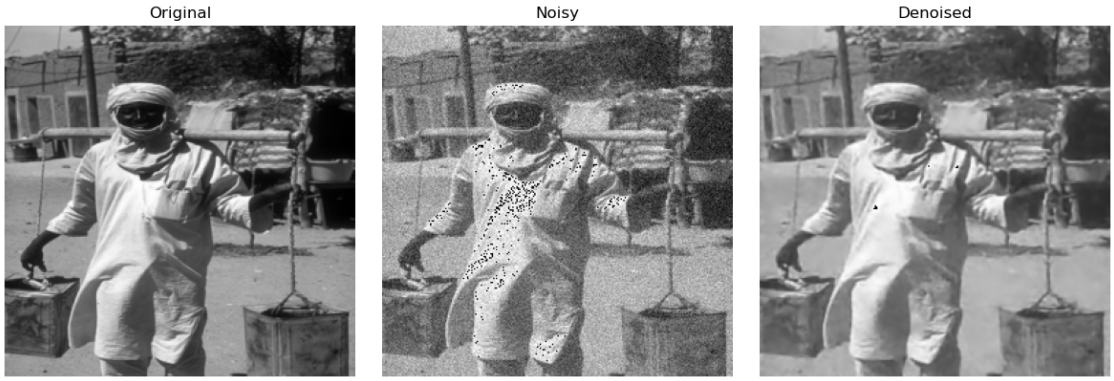


Image 10 in Batch 2, Test Loss: 0.0008454472990706563

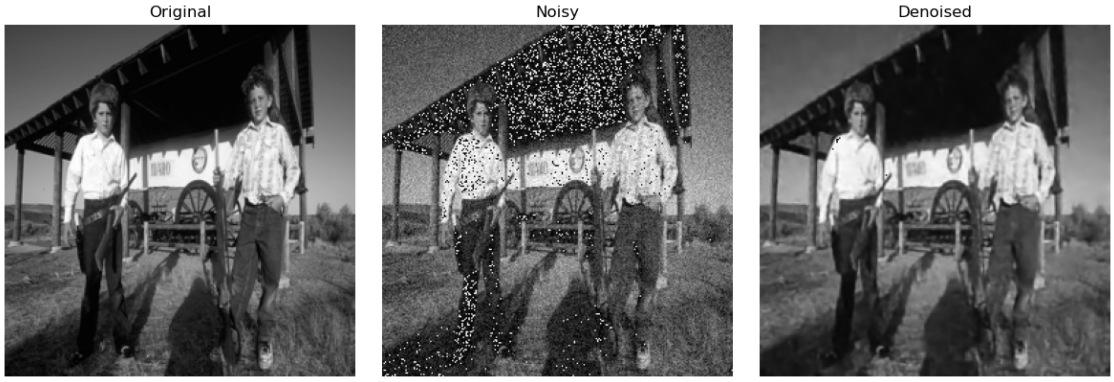


Image 11 in Batch 2, Test Loss: 0.0009167669340968132

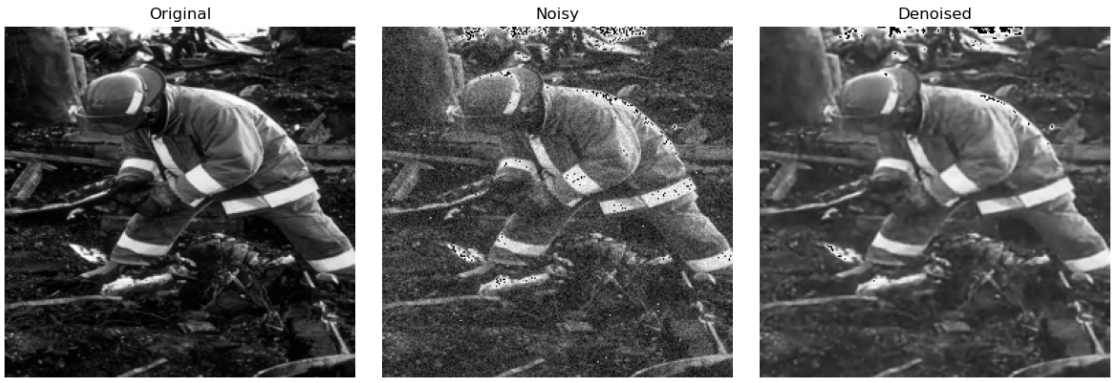


Image 12 in Batch 2, Test Loss: 0.0010788342915475368

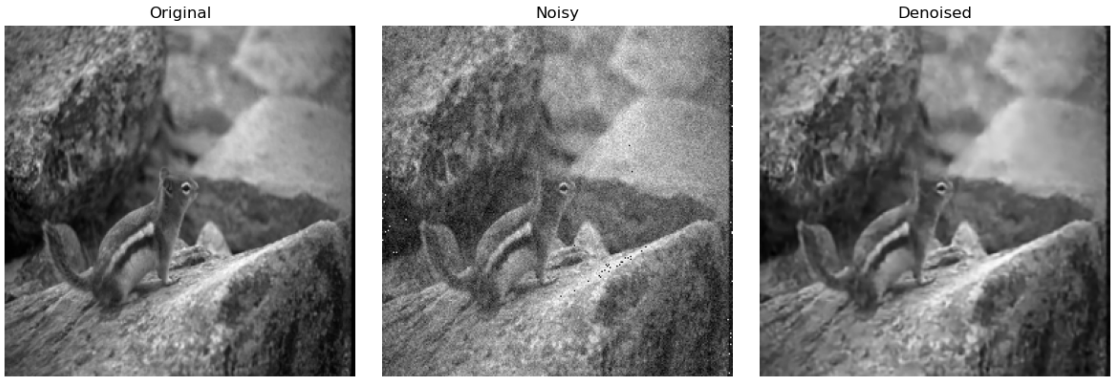


Image 13 in Batch 2, Test Loss: 0.0007231123745441437

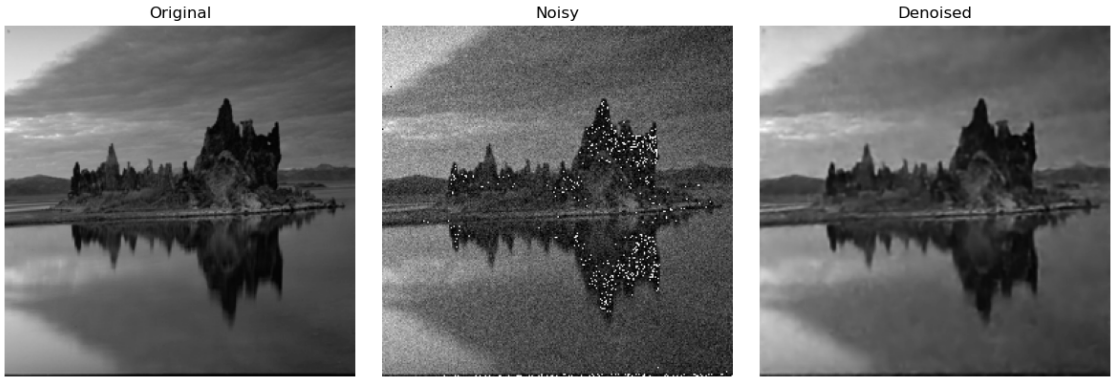


Image 14 in Batch 2, Test Loss: 0.00035990815376862884

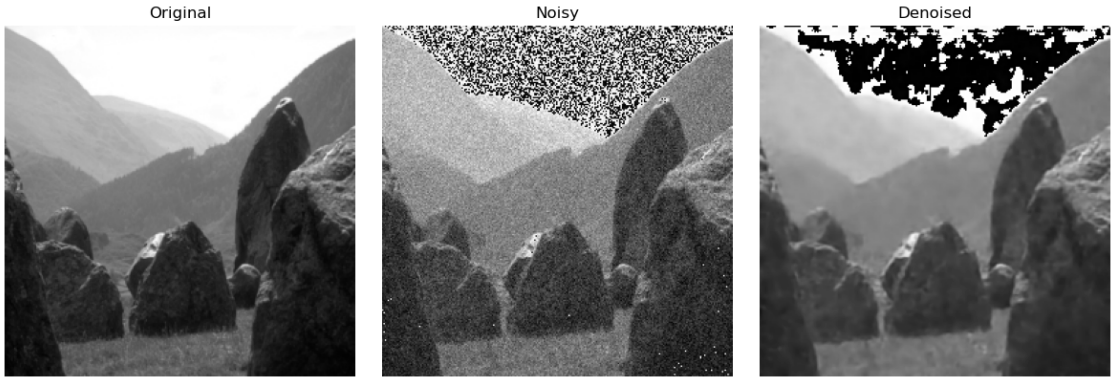


Image 15 in Batch 2, Test Loss: 0.00043999083572998643

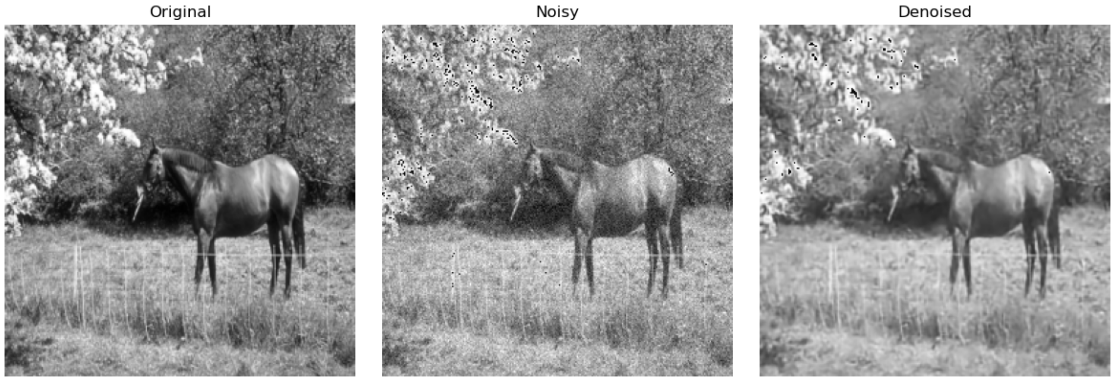


Image 1 in Batch 3, Test Loss: 0.001893174252472818

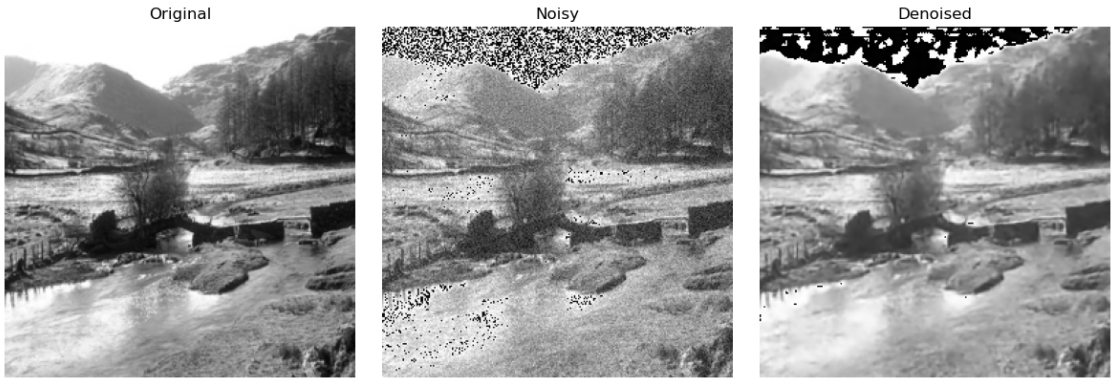


Image 2 in Batch 3, Test Loss: 0.0010620387038215995

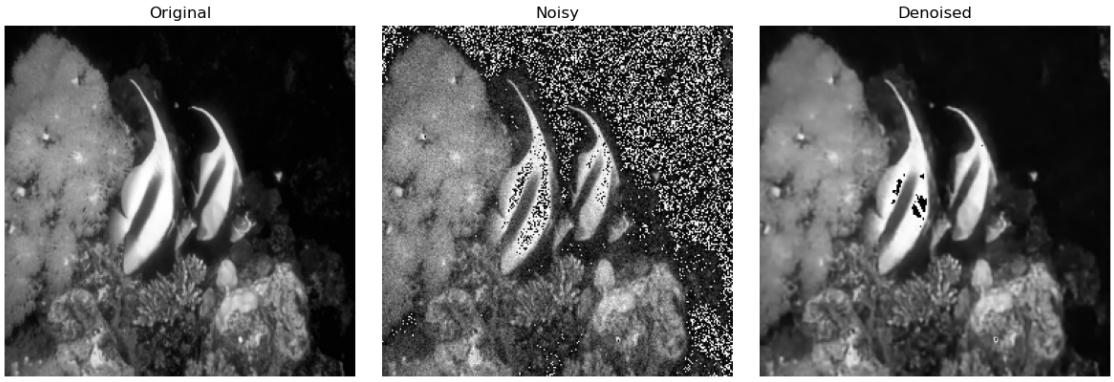
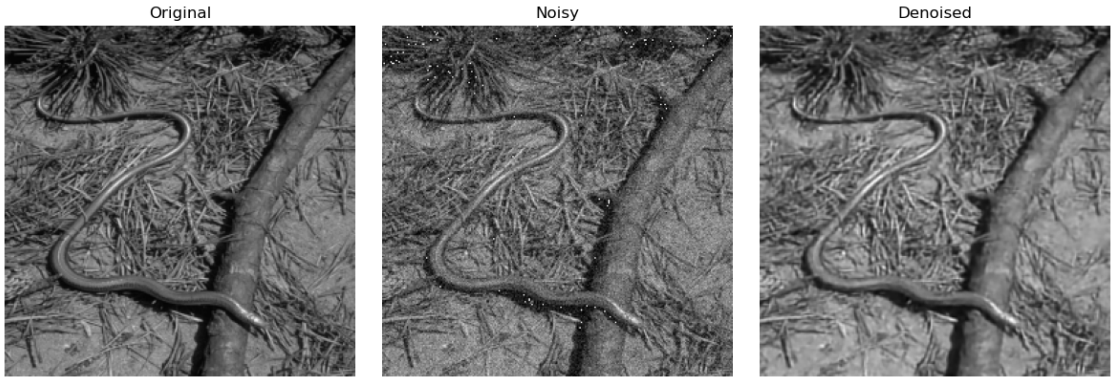
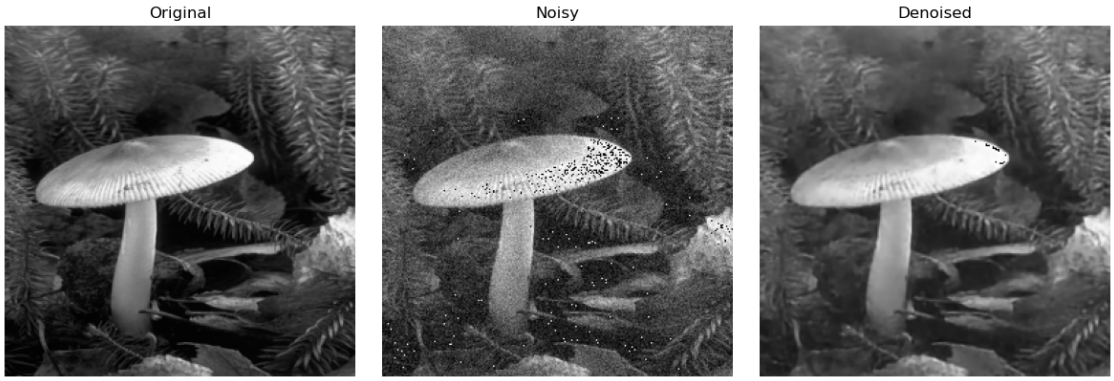


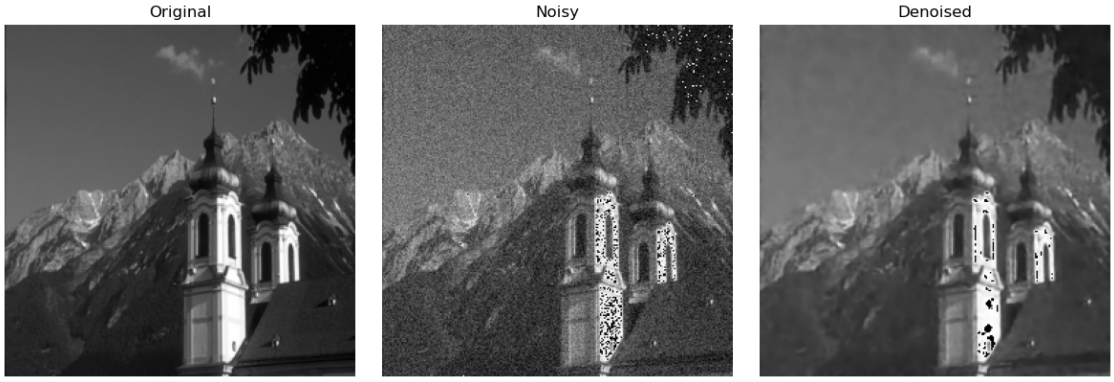
Image 3 in Batch 3, Test Loss: 0.0007670946652069688



```
Image 4 in Batch 3, Test Loss: 0.0019842800684273243
```




```
Image 5 in Batch 3, Test Loss: 0.0008231191895902157
```




```
Image 6 in Batch 3, Test Loss: 0.0005628000944852829
```

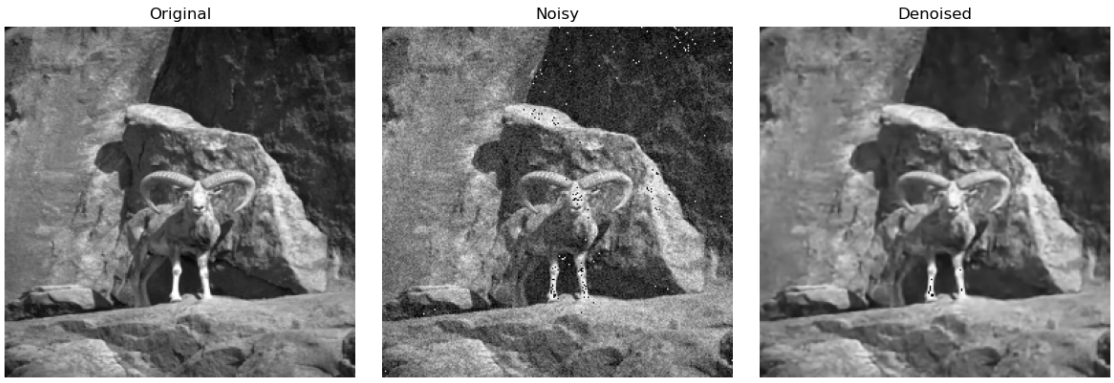


Image 7 in Batch 3, Test Loss: 0.001059655100107193

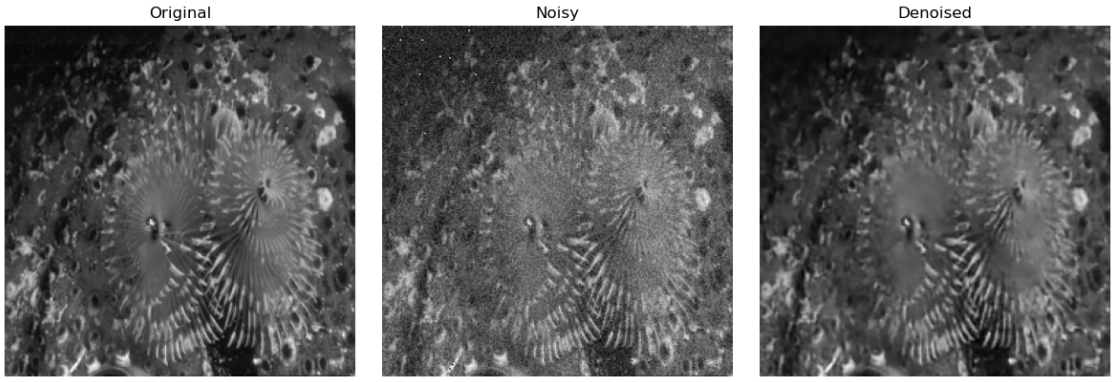


Image 8 in Batch 3, Test Loss: 0.000940916477702558

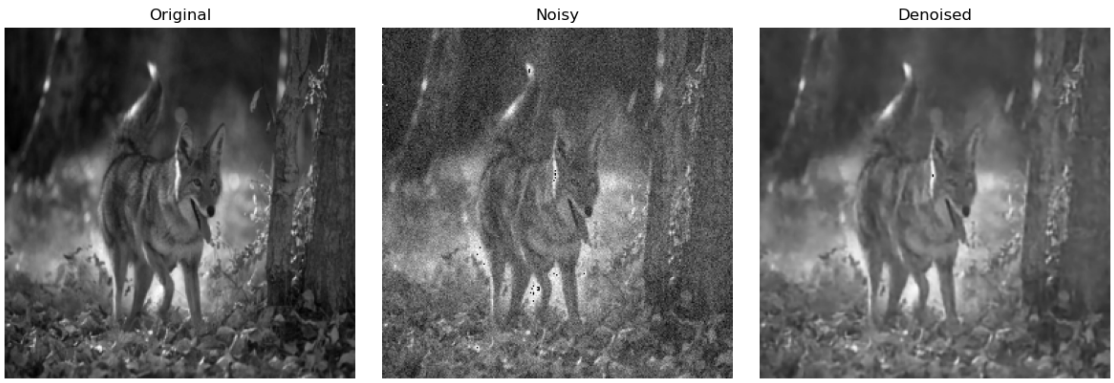


Image 9 in Batch 3, Test Loss: 0.0006638909108005464

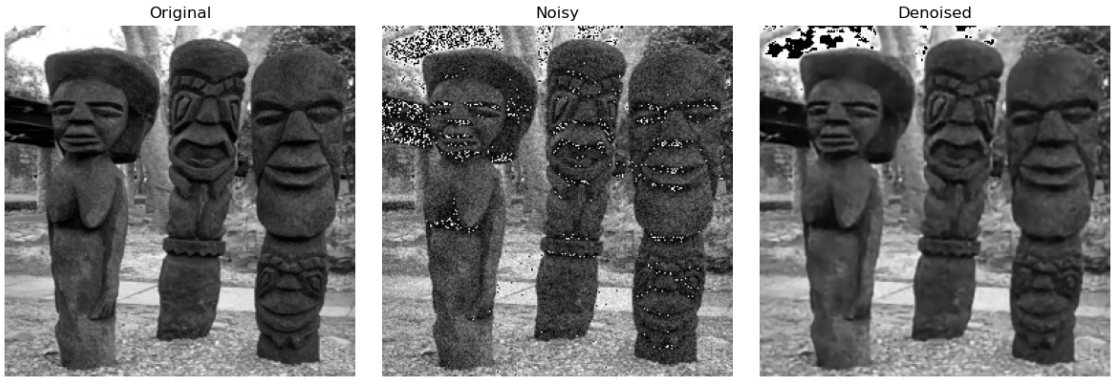


Image 10 in Batch 3, Test Loss: 0.0012776501243934035

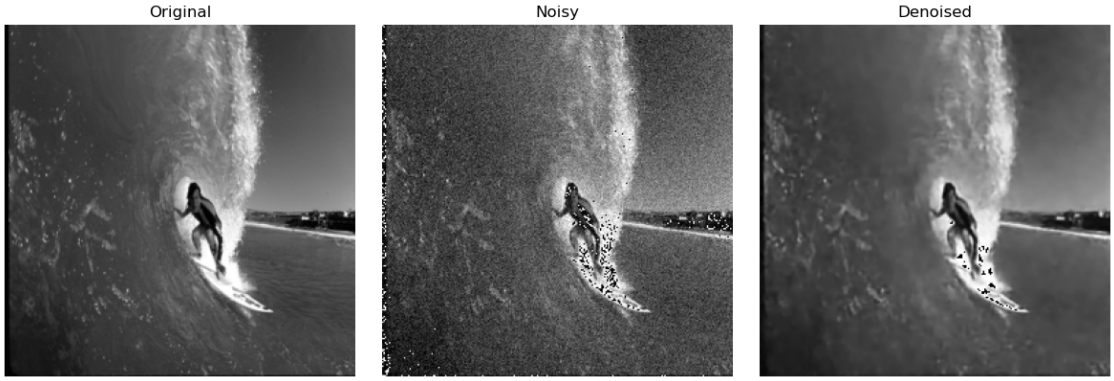


Image 11 in Batch 3, Test Loss: 0.0005931183695793152

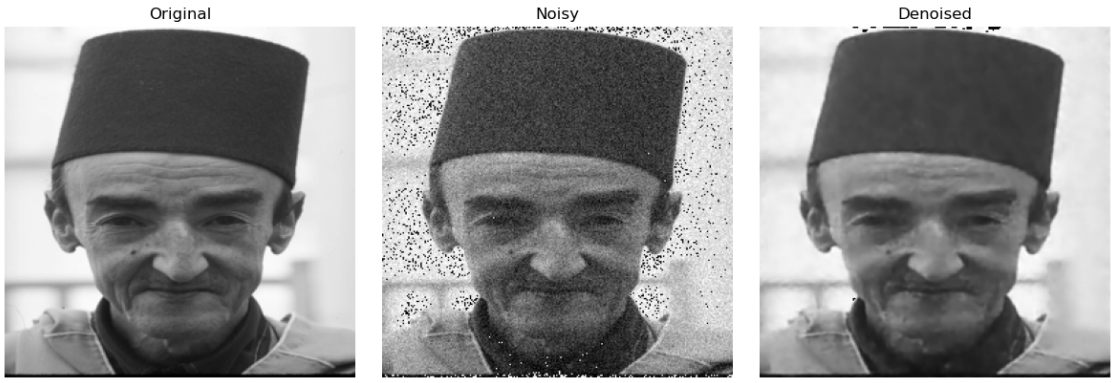


Image 12 in Batch 3, Test Loss: 0.00038743476034142077

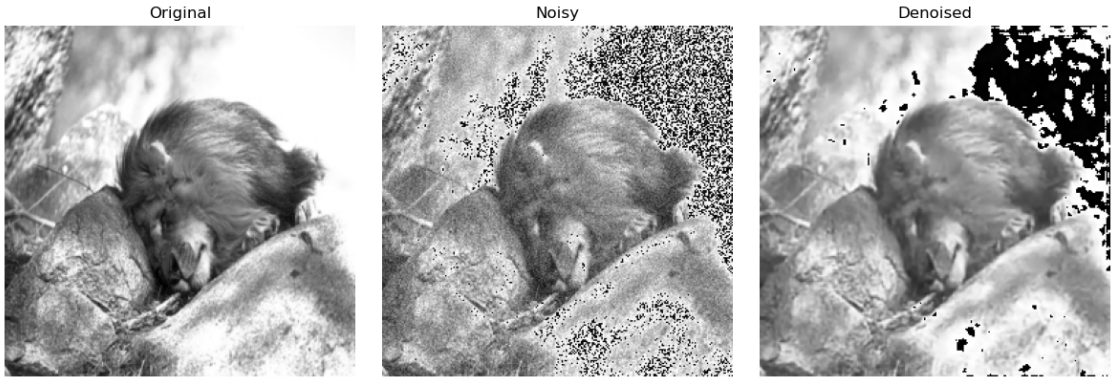


Image 13 in Batch 3, Test Loss: 0.0010267633479088545

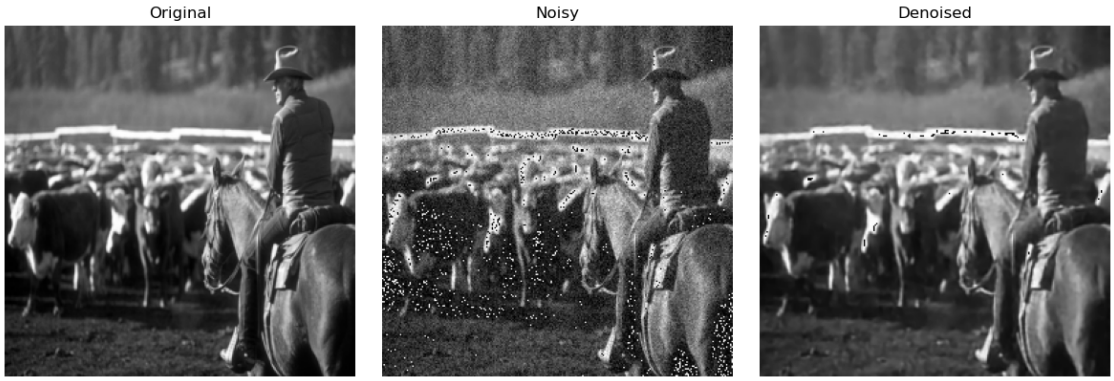


Image 14 in Batch 3, Test Loss: 0.00071401905734092

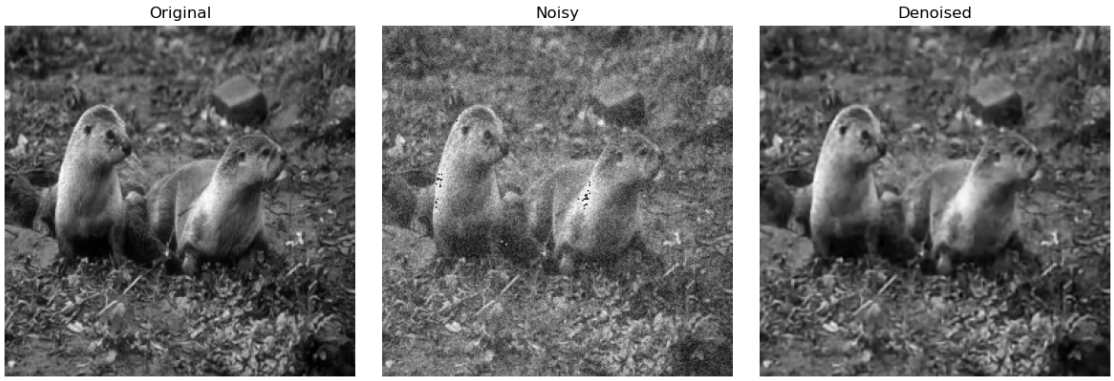


Image 15 in Batch 3, Test Loss: 0.001198161393404007

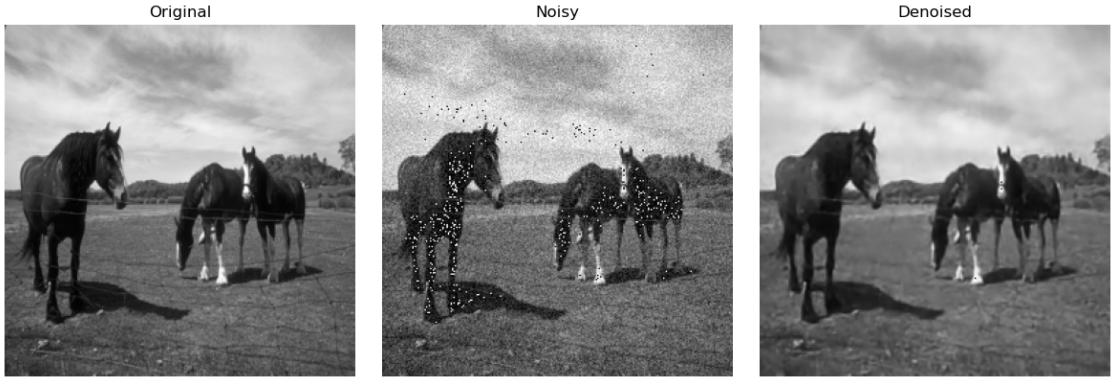


Image 1 in Batch 4, Test Loss: 0.0007412271806970239

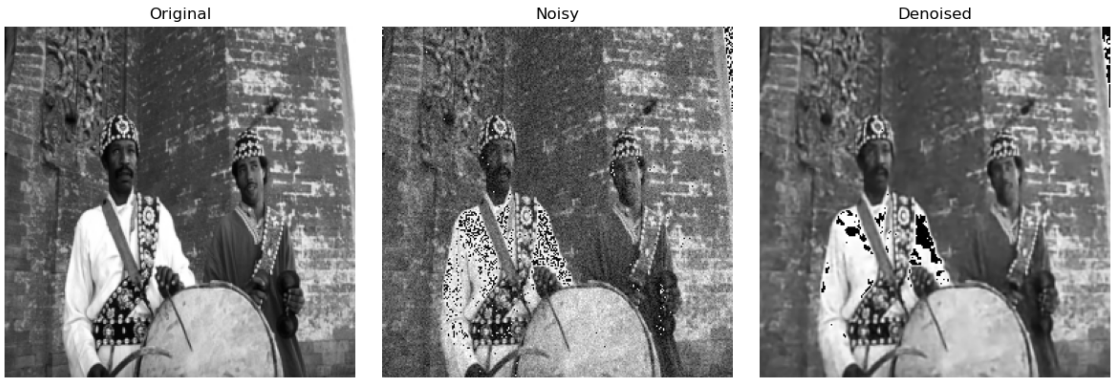


Image 2 in Batch 4, Test Loss: 0.0014204350300133228

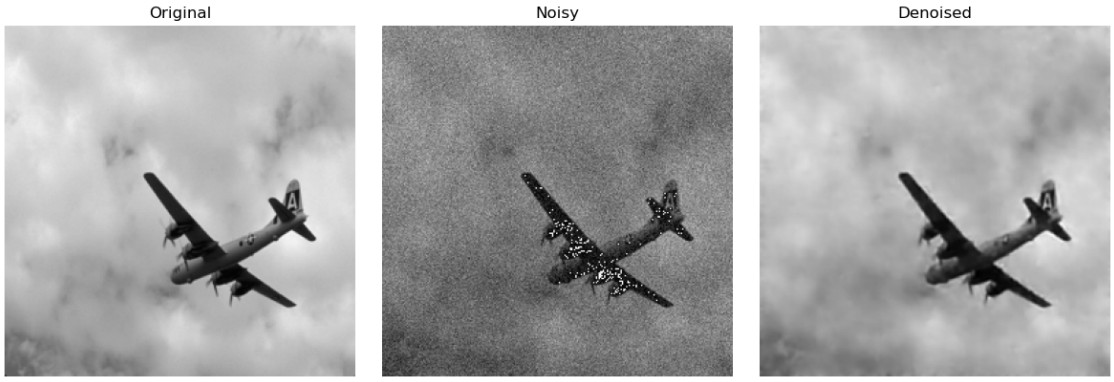


Image 3 in Batch 4, Test Loss: 0.0001880867057479918

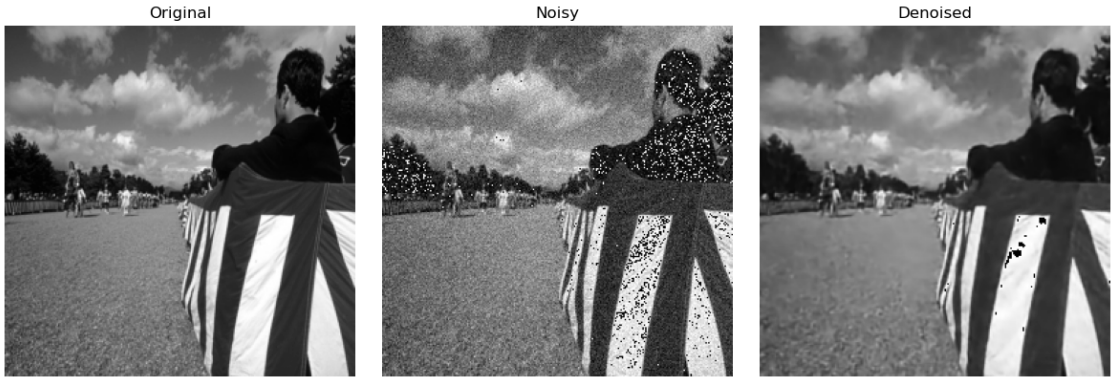


Image 4 in Batch 4, Test Loss: 0.0008712181588634849

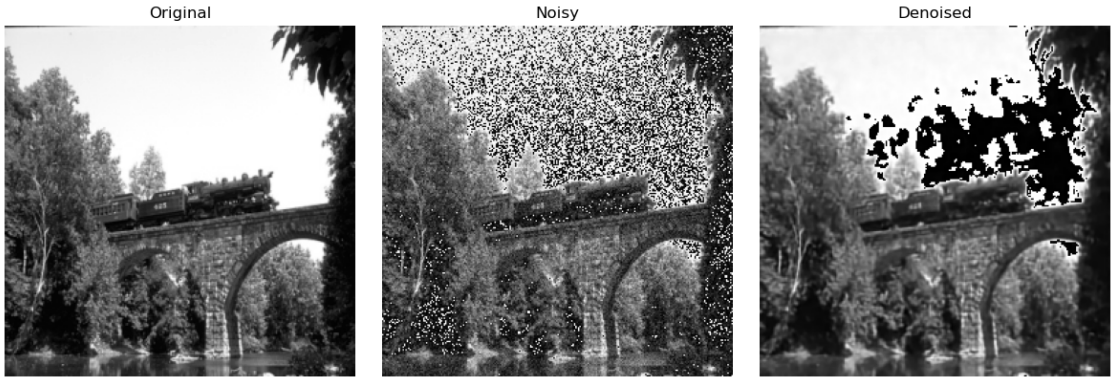


Image 5 in Batch 4, Test Loss: 0.0013445103541016579

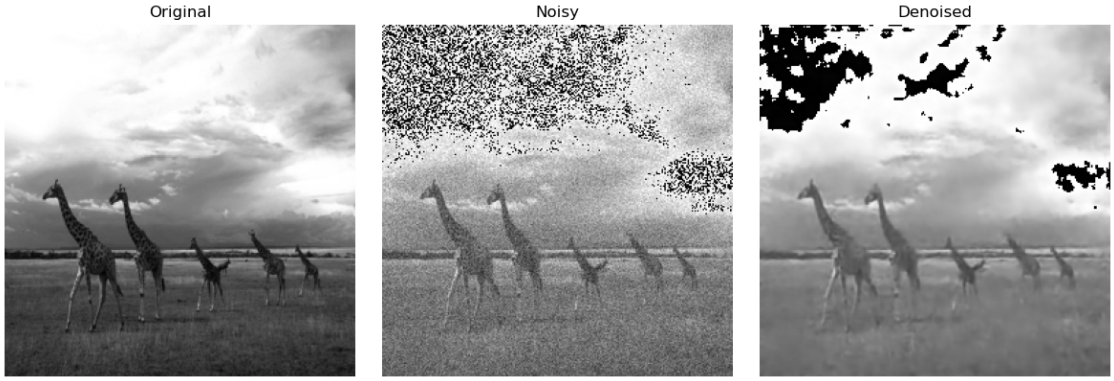


Image 6 in Batch 4, Test Loss: 0.00042360989027656615

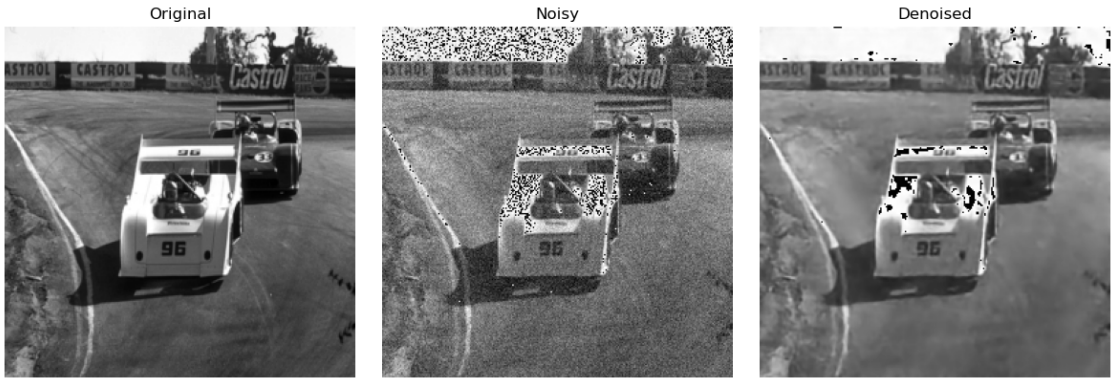


Image 7 in Batch 4, Test Loss: 0.0007451277342624962

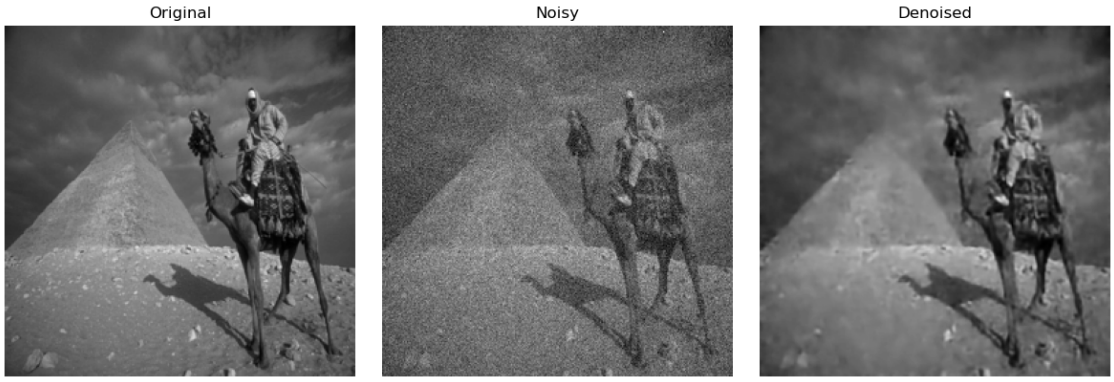


Image 8 in Batch 4, Test Loss: 0.0004463637596927583

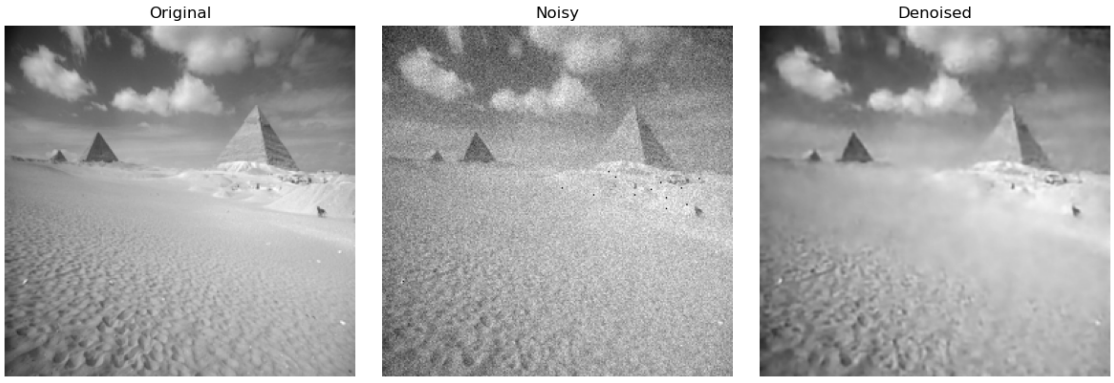


Image 9 in Batch 4, Test Loss: 0.00048592424718663096

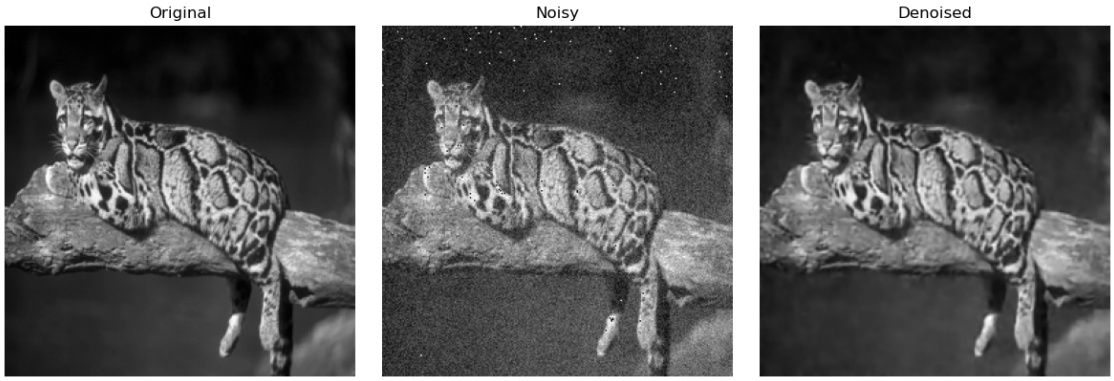


Image 10 in Batch 4, Test Loss: 0.0006811622879467905

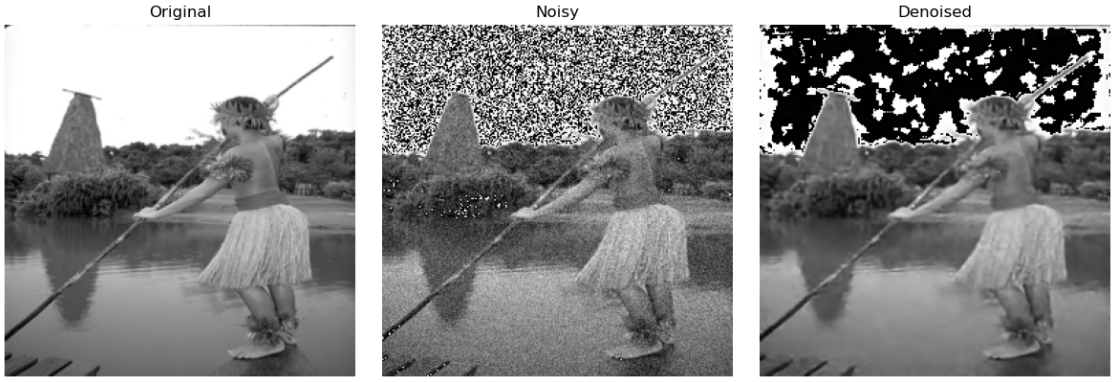


Image 11 in Batch 4, Test Loss: 0.0006814382504671812

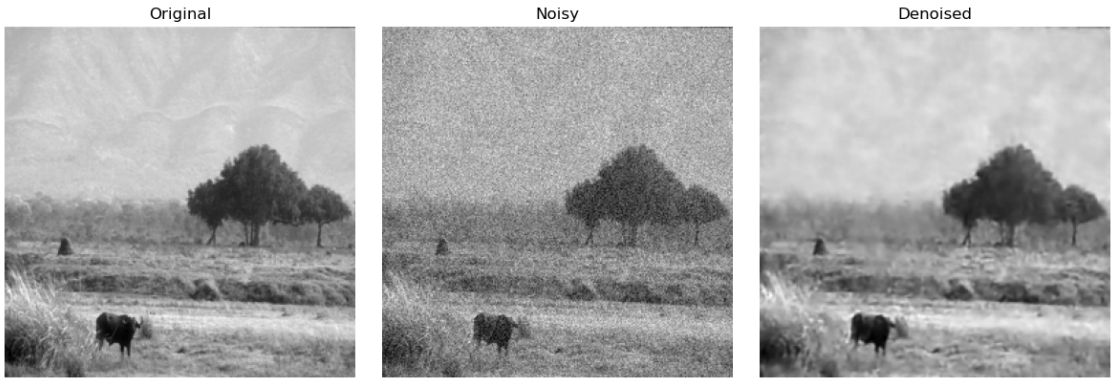


Image 12 in Batch 4, Test Loss: 0.00044971625902689993

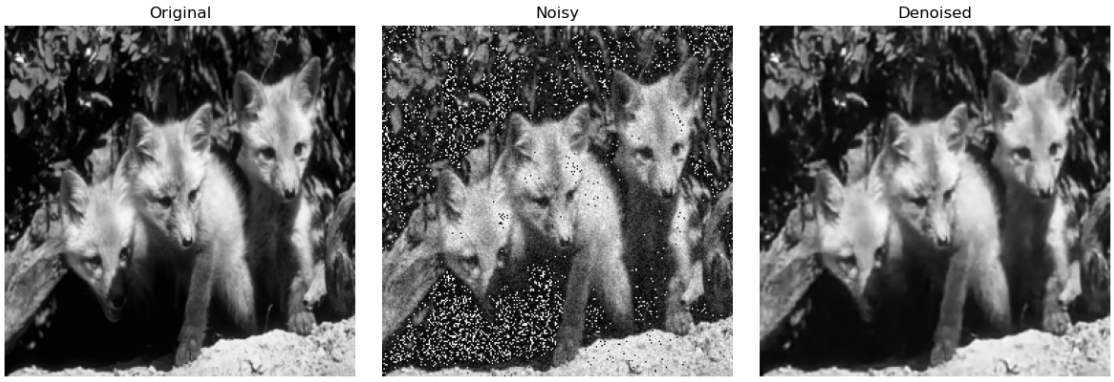


Image 13 in Batch 4, Test Loss: 0.0009051400702446699

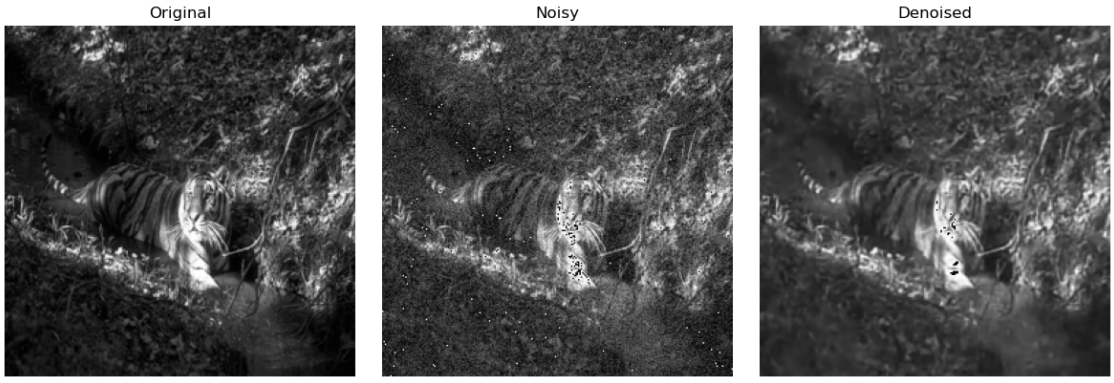


Image 14 in Batch 4, Test Loss: 0.0013808859512209892

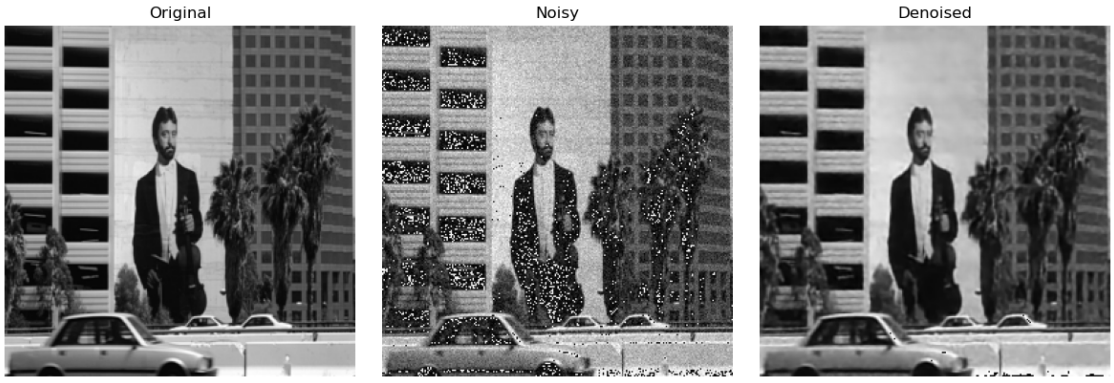


Image 15 in Batch 4, Test Loss: 0.0010248390026390553

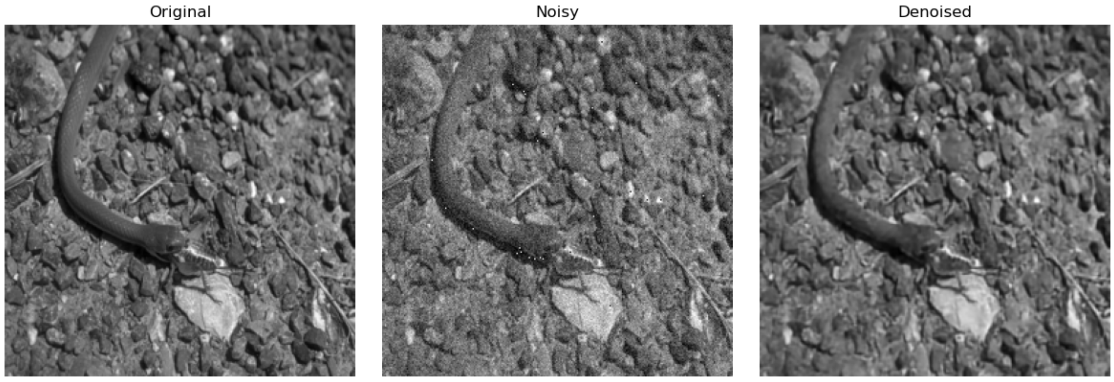


Image 1 in Batch 5, Test Loss: 0.0012659365311264992

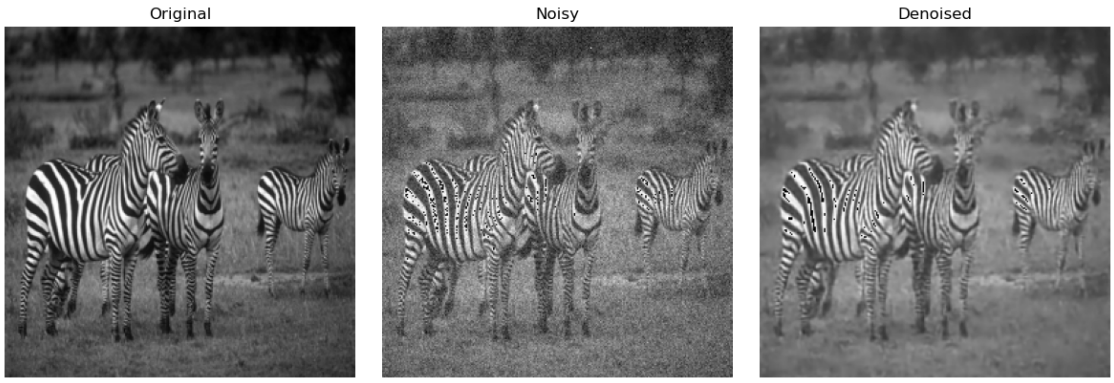


Image 2 in Batch 5, Test Loss: 0.0011564248707145452

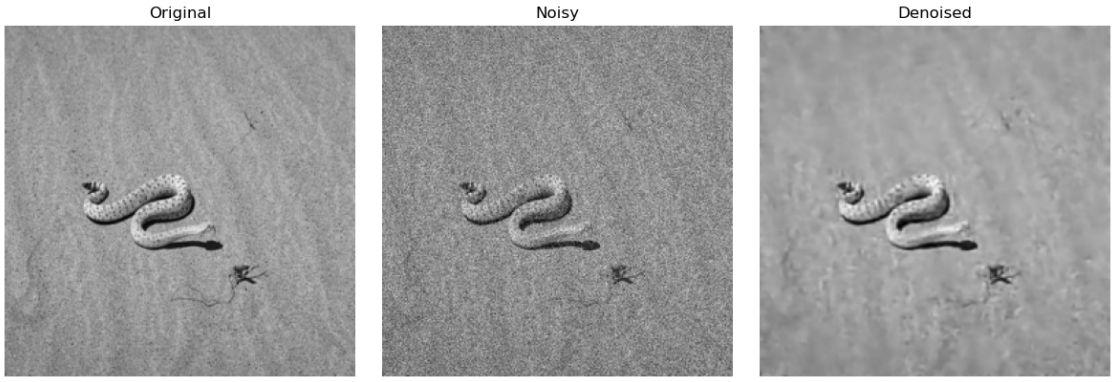


Image 3 in Batch 5, Test Loss: 0.0006930177332833409

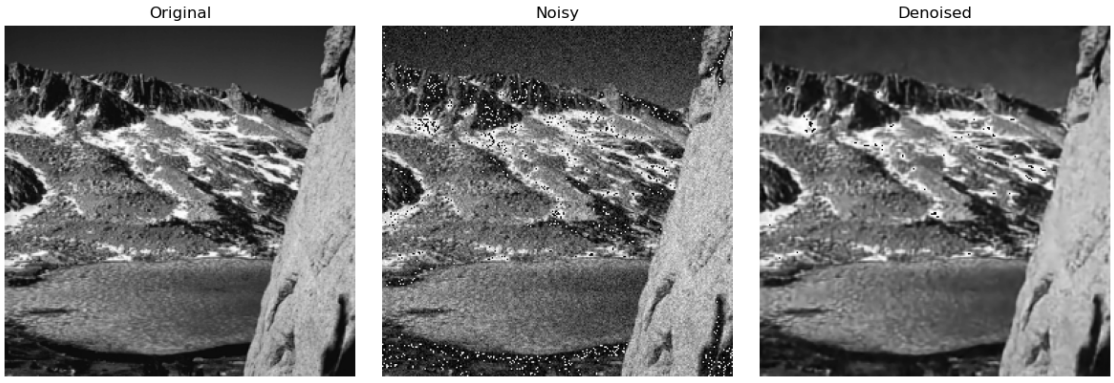


Image 4 in Batch 5, Test Loss: 0.001982608810067177

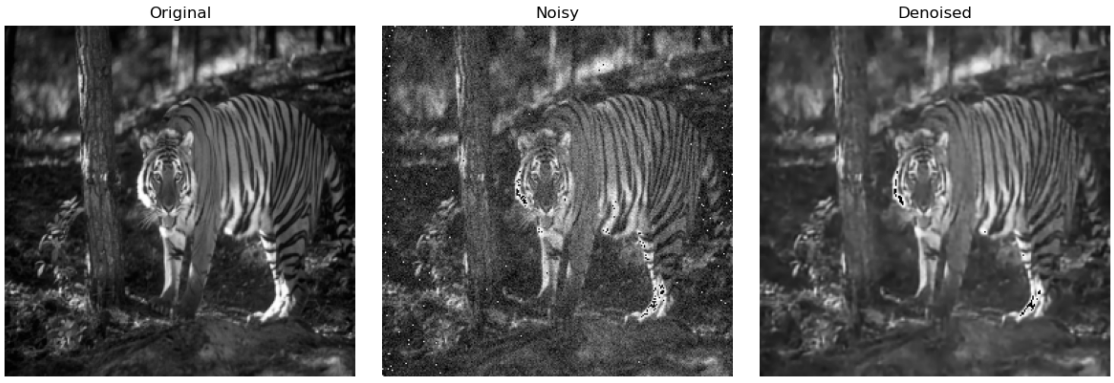


Image 5 in Batch 5, Test Loss: 0.0009404188022017479

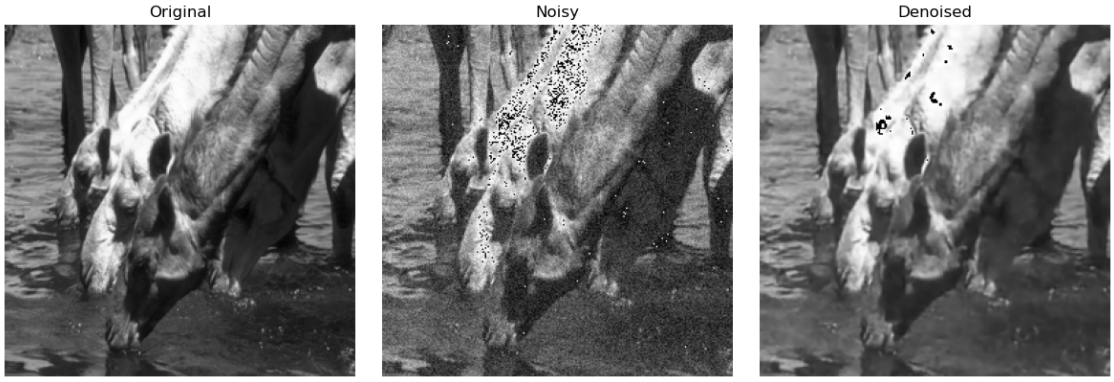
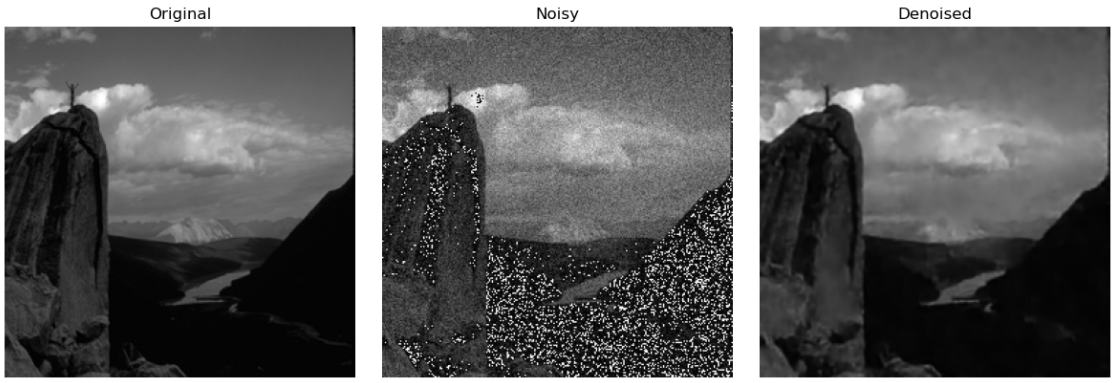
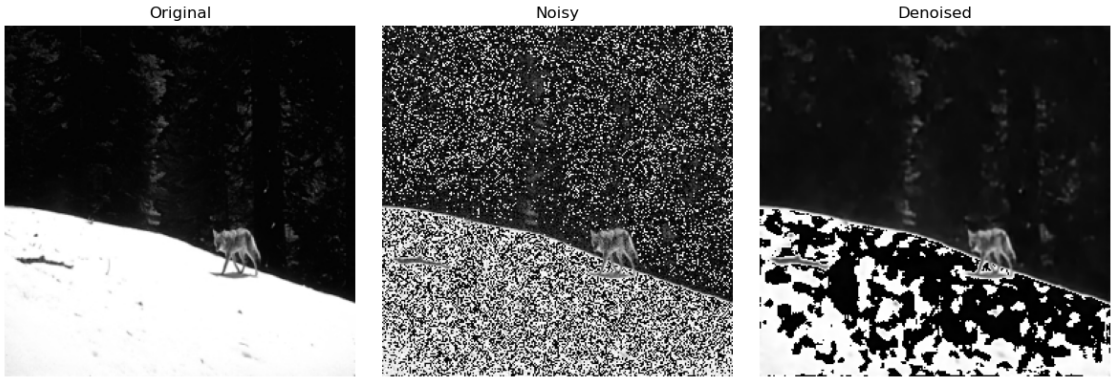


Image 6 in Batch 5, Test Loss: 0.0008354891906492412




```
Image 7 in Batch 5, Test Loss: 0.0003731028991751373
```




```
Image 8 in Batch 5, Test Loss: 0.0003757805097848177
```

```
[ ]:
```