Assignment 3 - Asmae Mouradi

Mathematical Details on how to calculate SSIM.

November 2025

1 Question 2 Design a denoising autoencoder model for removing noises from the documents

1.1 Structural Similarity Index Measure (SSIM)

To evaluate the quality of the denoised images, I used the Structural Similarity Index Measure (SSIM) between my model's output and the corresponding clean ground-truth image.

Let

- x denote the clean (ground-truth) image
- y denote the denoised image predicted by the autoencoder.

Both images are grayscale and normalized to the range $[0, 1]$. I compute SSIM using the `structural_similarity` function from the `skimage.metrics` module as

```
ssim_val = ssim(target, pred, data_range=1.0)
```

where `target` is the clean image and `pred` is the denoised output.

SSIM is computed *locally* on small windows (patches) extracted from the two images and then averaged over the whole image.

For a given local window containing N pixels, I denote:

- μ_x as the mean intensity of the clean patch,
- μ_y as the mean intensity of the predicted (denoised) patch
- σ_x^2 as the variance of the clean patch
- σ_y^2 as the variance of the predicted patch,
- σ_{xy} as the covariance between the two patches.
- C_1 and C_2 are small positive constants added to avoid division by zero and to ensure numerical stability.

They are computed as

$$\mu_x = \frac{1}{N}\sum_{i=1}^{N} x_i, \qquad \mu_y = \frac{1}{N}\sum_{i=1}^{N} y_i, \tag{1}$$

$$\sigma_x^2 = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \mu_x)^2, \qquad \sigma_y^2 = \frac{1}{N-1}\sum_{i=1}^{N}(y_i - \mu_y)^2, \tag{2}$$

$$\sigma_{xy} = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \mu_x)(y_i - \mu_y), \tag{3}$$

where x_i and y_i are the pixel values in the local window from the clean and denoised images, respectively.

For each local window meaning tiny square of pixels, SSIM is defined as

$$\text{SSIM}(x, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}, \tag{4}$$

C_1 and C_2 are small positive constants introduced to avoid division by zero.

$$C_1 = (K_1 L)^2, \qquad C_2 = (K_2 L)^2 githu \tag{5}$$

with

- L the dynamic range of the pixel intensities
- $K_1 \approx 0.01$
- $K_2 \approx 0.03$

I don't pass K_1 and K_2 as arguments, so skimage.metrics.ssim uses its default values.

In my case, the images are normalized to $[0, 1]$, so $L = 1$, and I set `data_range=1.0`.

This image-level SSIM formula shows that, after computing SSIM on many small local windows (patches) across the image, I simply average all these local SSIM values to obtain a single similarity score for the entire image.

The function I use slides this local window over the entire image. For each position, it computes the local SSIM value according to (4). The final SSIM score for an image is obtained by averaging all local SSIM values:

$$\text{SSIM}_{\text{image}}(x, y) = \frac{1}{M}\sum_{j=1}^{M} \text{SSIM}_j(x, y), \tag{6}$$

where $\text{SSIM}_j(x, y)$ is the SSIM value for the j-th window and M is the total number of windows.

In my experiments, for each independent test image I compute this image-level SSIM between the denoised output and its corresponding clean ground-truth image. This gives me a scalar value in $[0, 1]$ (for my normalized images) that quantitatively measures the structural similarity and thus the quality of the denoising.

2 Question 3 Generative Models Hands-on Experiments

2.1 Structural Similarity Index Measure (SSIM)

2.1.1 Variational Autoencoder (VAE)

To measure the perceptual similarity between an original image x and its reconstruction $\hat{x}$, I use the Structural Similarity Index (SSIM). SSIM is computed on small local patches from both images and then averaged over all patches.

For a given patch from the original image with values $x_1, \ldots, x_N$ and the corresponding patch from the reconstruction with values $\hat{x}_1, \ldots, \hat{x}_N$, I first compute:

$$\mu_x = \frac{1}{N}\sum_{i=1}^{N} x_i, \qquad \mu_y = \frac{1}{N}\sum_{i=1}^{N} \hat{x}_i,$$

$$\sigma_x^2 = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \mu_x)^2, \qquad \sigma_y^2 = \frac{1}{N-1}\sum_{i=1}^{N}(\hat{x}_i - \mu_y)^2,$$

$$\sigma_{xy} = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \mu_x)(\hat{x}_i - \mu_y).$$

Using these statistics, the SSIM value for this patch is defined as

$$\mathrm{SSIM}(x, \hat{x}) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)},$$

where C_1 and C_2 are small constants for numerical stability.

These constants are usually defined as

$$C_1 = (k_1 L)^2, \qquad C_2 = (k_2 L)^2,$$

where L is the dynamic range of the pixel values (in my case $L = 1$, since images are in $[0, 1]$) and typically $k_1 = 0.01$, $k_2 = 0.03$.

The overall SSIM score for an image is obtained by averaging $\mathrm{SSIM}(x, \hat{x})$ over all local patches.

2.1.2 Deep Convolutional Generative Adversarial Network (DCGAN).

To quantitatively compare real CIFAR-10 images and the images generated by my DCGAN generator, I compute the Structural Similarity Index (SSIM) between them.

SSIM (and most image quality metrics) are usually defined for images whose pixel values lie in the range $[0, 1]$ or $[0, 255]$. In my DCGAN pipeline, however, both real and generated images are normalized to $[-1, 1]$. To be consistent with

the SSIM definition and to use `data_range = 1.0`, I first map the images back to $[0, 1]$ via the linear rescaling

$$x' = \frac{x+1}{2}, \qquad \hat{x}' = \frac{\hat{x}+1}{2},$$

where x and $\hat{x}$ denote the normalized real and generated images, and x' and $\hat{x}'$ are the corresponding rescaled images in $[0, 1]$. SSIM is then computed on local patches of x' and $\hat{x}'$. For a given patch with values $x_1, \ldots, x_N$ from x' and $\hat{x}_1, \ldots, \hat{x}_N$ from $\hat{x}'$, I compute

$$\mu_x = \frac{1}{N}\sum_{i=1}^{N} x_i, \qquad \mu_y = \frac{1}{N}\sum_{i=1}^{N} \hat{x}_i,$$

$$\sigma_x^2 = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \mu_x)^2, \qquad \sigma_y^2 = \frac{1}{N-1}\sum_{i=1}^{N}(\hat{x}_i - \mu_y)^2,$$

$$\sigma_{xy} = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \mu_x)(\hat{x}_i - \mu_y).$$

Using these statistics, the SSIM value for that patch is

$$\mathrm{SSIM}(x', \hat{x}') = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)},$$

where C_1 and C_2 are small constants for numerical stability. As in the standard definition, I set

$$C_1 = (k_1 L)^2, \qquad C_2 = (k_2 L)^2,$$

with $L = 1$ (because pixel values are in $[0, 1]$) and $k_1 = 0.01$, $k_2 = 0.03$.

In my implementation I use the `skimage.metrics.structural_similarity` function with a window size of 11, `data_range = 1.0`, and `channel_axis = -1`, so the SSIM statistics are computed jointly over the three RGB channels. For each mini-batch, I compute the SSIM value for each pair $(x_i', \hat{x}_i')$ of real and generated images, average these values within the batch, and then average over all batches to obtain the final mean SSIM score for the DCGAN.