Image Processing - Exercise 5

Shir Hana Stern, shirste, 325945236

Introduction

The goal of this exercise was to explore and apply advanced concepts in generative adversarial networks (GANs), specifically using StyleGAN2, to perform various image reconstruction tasks. Through this exercise, I aimed to understand the underlying principles of GAN Inversion, Latent Optimization, and the role of Image Priors in achieving high-quality image reconstruction.

The main techniques I employed include- **GAN Inversion:** This involves mapping a given image back into the latent space of the StyleGAN2 model. By doing so, I learned how the latent space encodes rich semantic information, enabling the reconstruction of images that align with the original input. **Latent Optimization:** This technique taught me how iterative optimization in the latent space can refine the reconstructed image, minimizing the discrepancy between the input image and its generated counterpart. **Image Priors:** I gained insight into how leveraging priors, such as smoothness, sparsity, or other properties inherent to natural images, can guide the reconstruction process to produce visually plausible results.
This exercise deepened my understanding of how generative models operate, how their latent spaces can be utilized for reconstruction, and how optimization and priors play a critical role in achieving high-quality outcomes.

Algorithm

Problem Definition: Reconstruct a degraded target image by optimizing a latent vector in the latent space of a pre-trained GAN such that the synthesized image resembles the target image under specified degradation conditions.

Latent Space Initialization: Compute the mean latent vector and its standard deviation using a set of random samples mapped through the GAN's mapping network. Initialize the latent vector for optimization with. Initialize noise buffers used in the GAN's synthesis network with random values to add stochastic detail during synthesis.

Preprocessing of the Target Image: The target image is loaded and processed to match the GAN's expected resolution, channel count, and dynamic range. If the image doesn't match the GAN's dimensions, it is resized and normalized appropriately. Degradation is applied to the target image using the selected degradation mode: **Inpainting:** Masked regions in the image are set to zero.

Grayscale: The image is converted to grayscale and repeated across channels.
Gaussian Blur: A Gaussian blur is applied to the image.

Optimization Loop: An iterative process is performed to optimize the latent vector over a fixed number of steps. At each step:
(Learning Rate Schedule) Adjust the learning rate dynamically based on the progress of optimization, ramping up and down as needed.
(Latent Vector Perturbation) Add Gaussian noise to the current latent vector to explore the latent space.
(Synthesis) Generate an image using the GAN synthesis network with the current latent vector and noise buffers.
(Apply Degradation) Simulate the degradation process on the synthesized image to match the target's degraded characteristics.
(Feature Extraction) Use a pre-trained VGG16 network to extract perceptual features from both the target and synthesized images.
(Loss Computation) Combine the following losses: **Perceptual Loss:** Measures similarity between target and synthesized images in feature space. **Latent Distance Regularization:** Penalizes deviation of w_opt from w_avg to ensure plausible latent values. **Noise Regularization:** Regularizes noise buffers to reduce artifacts.
(Backpropagation) Compute gradients and update the latent vector and noise buffers using the optimizer (Adam). (Intermediate Results) Save and display intermediate synthesized and degraded images for visualization.

Output Results: Generate the final reconstructed image using the optimized latent vector and save it to disk. Save the optimized latent vector for potential reuse or further processing.

Results

Results of 3.1

Before image alignment script

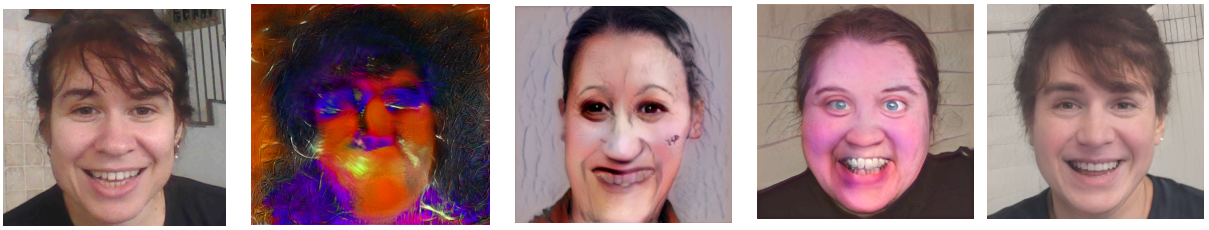
After image alignment script

The visualization presented demonstrates the impact of the image alignment script on the original image. The original image, taken before running the script, was wide and included a face as well as numerous background details. This unprocessed image provided a broad context but lacked focus on the primary subject, which is the face.

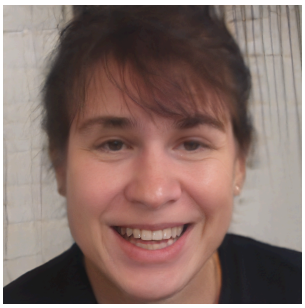
After applying the image alignment script, the resulting image is significantly refined. The background details have been minimized or entirely removed, ensuring that the image contains mostly the face, which is now the central focus. Furthermore, the aligned image is smaller in size, likely due to cropping and resizing operations performed during the alignment process to standardize the dimensions and isolate the desired subject.

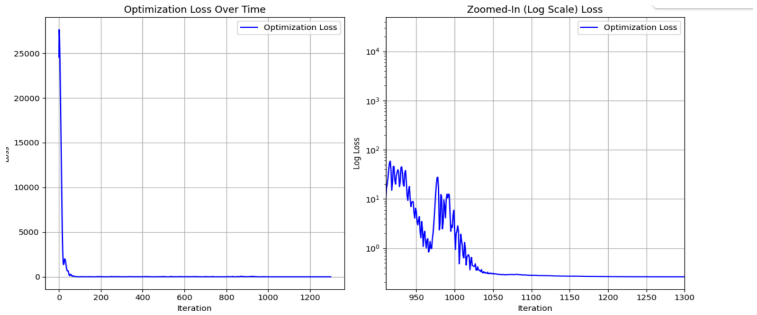
Results of 3.2

original image **image by the initial z** **3 other images from the optimization process**



final image **plot of the optimization loss**





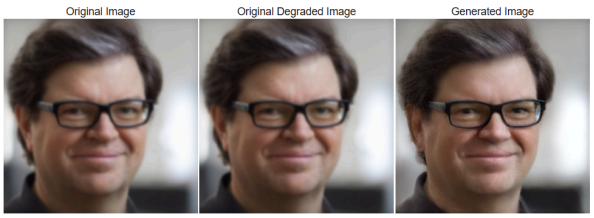
The optimization process can be analyzed through the generated images and the loss curve, both of which provide insight into how the model refines the latent representation over time. Initially, the loss is significantly high, indicating that the generated image is far from the target image. This is evident in the first few iterations, where the synthesized image appears highly distorted and lacks resemblance to the original. However, as the optimization progresses, the loss decreases rapidly, and the generated images start capturing the main structural features of the target.

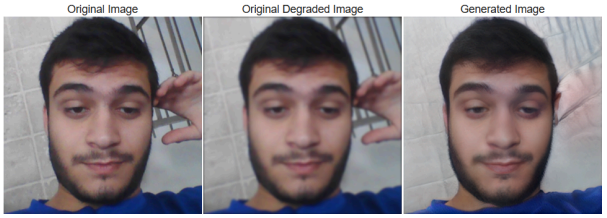
In the later stages of optimization, the rate of loss reduction slows down, and the visual differences between consecutive iterations become less noticeable. While

early iterations introduce substantial changes that bring the generated image closer to the target, later iterations primarily refine details. This stabilization in loss suggests that the optimization has reached a point where further improvements are minimal, and the generated image has closely approximated the target.

The latent distance regularization weight plays a crucial role in this process. A higher weight constrains the optimization, keeping the latent vector close to the original latent space distribution. While this prevents drastic changes and helps maintain realism, it may limit the accuracy of the reconstruction. On the other hand, a lower weight allows more flexibility in optimization but increases the risk of overfitting or producing unrealistic artifacts. Similarly, the number of optimization steps determines how well the generated image can approximate the target. Too few iterations result in incomplete convergence, while excessive iterations provide diminishing improvements beyond a certain point.

Results of 3.3.1



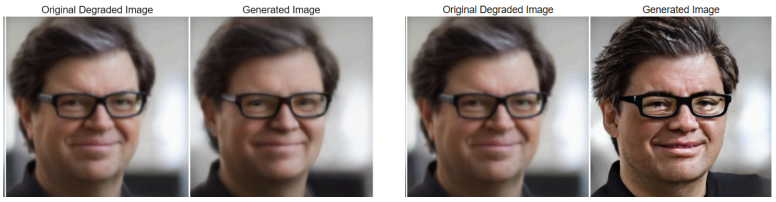


For this image deblurring task, I used a Gaussian blur function to create a blurred version of the original image. The function applies a kernel-based convolution to simulate the blurring effect. Afterwards, I used a loss function to compare the blurred generative images with the original blurred image, allowing for the reconstruction of a deblurred image. The function I implemented uses a Gaussian kernel with a customizable kernel size and sigma value. By adjusting these parameters, the degree of blurring can be controlled. This gives flexibility in generating blurred images that can be effectively deblurred.

One of the main challenges I faced was that I encountered an issue where the images did not blur as expected, which led to poor deblurring results. Upon investigating, I realized that the kernel size I was using was too small, which did not introduce enough blur to make the deblurring task feasible. To solve this, I experimented with larger kernel sizes and adjusted the sigma parameter. This helped achieve the desired level of blurring, allowing the deblurring process to produce meaningful results.

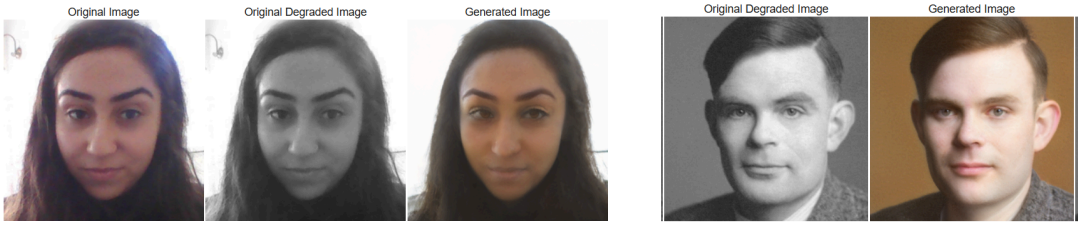
The main difficulty was determining the appropriate hyperparameters (kernel size and sigma). This required a lot of experimentation and testing, as I had to find a balance between not overly blurring the image, yet creating enough distortion for the deblurring model to work. The process involved iterating through various kernel sizes and sigma values while constantly monitoring the outputs to optimize the results.

kernel size - 5 **kernel size - 51**



Larger kernel size and higher sigma create stronger blur, making it harder for the model to reconstruct details. If the reconstruction parameters differ from the original blur, the model will struggle to match the original.
Smaller kernel size and lower sigma retain more detail in the blurred image, making it easier for the model to recover the non-blurred version.
The reconstruction's success depends on the alignment of blur parameters (kernel size and sigma) between the original and the modeled image. Misalignment can result in poor performance. The latent_dist_reg_weight hyperparameter affects how closely the latent vector used to generate the image matches the original latent vector. Higher value: The optimization process focuses more on keeping the latent vector close to the original, which can limit the model's ability to recover high-frequency details lost due to the blur, resulting in smoother reconstructions.
Lower value: The model has more freedom to adjust the latent vector, potentially improving the recovery of blurred details but at the risk of straying too far from the original image characteristics.

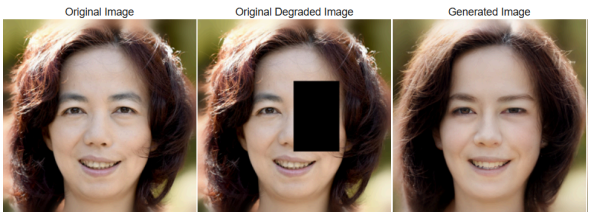
Results of 3.3.2

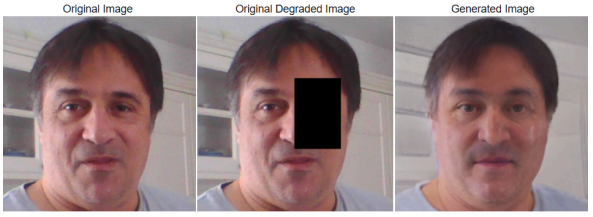


The colorization of the grayscale image was achieved through a latent space optimization approach using a pre-trained generative model. During this process, I encountered an issue where the colorized output appeared with a noticeable greenish tint instead of natural skin tones. This discrepancy indicated that the optimization was overfitting to non-realistic color variations in the latent space. To mitigate this, I increased the latent_dist_reg_weight parameter to 0.9. By doing so, the optimization was forced to stay closer to the average latent distribution, which helped produce more realistic colors in the final image.
The code was designed to handle different degradation modes, including GRAYSCALE_DEGRADATION. Specifically, for grayscale images, the degradation was applied by converting the synthesized color images to grayscale during the optimization process. This conversion was achieved by averaging the RGB channels

to create a single-channel grayscale image, which was then replicated across all three channels to maintain compatibility with the model's expected input format. This approach effectively simulated a grayscale degradation, ensuring that the colorization algorithm was tested against a true grayscale input.
Overall, by adjusting the latent_dist_reg_weight and carefully monitoring the optimization process, I was able to improve the quality of the colorization and achieve a more natural-looking result.

Results of 3.3.3





For this image inpainting task, I implemented a process where a binary mask is applied to the original image to simulate missing portions. The mask serves to "erase" part of the image, and the goal is to reconstruct this missing part so that the final unpainted image is semantically consistent with the original. To achieve this, I used a method where, for each original image, I immediately applied the binary mask. In the training process, for each generated image from the model, I applied the same mask and used a loss function to compare the masked generated image with the original image (which also has the same mask applied). This comparison helps guide the model to reduce the loss gradually, leading to the inpainting of the masked region and ultimately reconstructing the original image.
The motivation behind this approach is to encourage the model to generate semantically meaningful content in the masked areas by comparing it with the unmasked parts of the original image. The use of the binary mask ensures that the model focuses specifically on the missing areas, allowing for an effective inpainting solution. I did not encounter any significant issues during this part of the exercise.

Conclusion

This exercise significantly deepened my understanding of how to extract more information from images through the training process. By applying optimization techniques to adjust the latent space and using various degradation modes (such as inpainting, grayscale, and Gaussian blur), I observed how fine-tuning the latent representation of an image allows for better reconstruction and restoration of missing or degraded regions. The process of running latent optimization, applying masks, and iterating over steps to minimize perceptual and latent losses gave me insight into how generative models can be guided to reconstruct semantically meaningful details.