Image Processing - Exercise 5

Shachar Cohen, shahar.cohen1_, 206532418

Introduction

The primary objective of this exercise is to explore and experiment with neural networks for image processing. Specifically, we worked with a Generative Adversarial Network (GAN), a deep learning model designed to generate and reconstruct images from a given input. The key principle behind GANs lies in their "indirect" training process, where a discriminator network evaluates how realistic the generated images appear, guiding the generator to improve over time.

GANs generate images using a latent vector, a compact representation of the image being reconstructed. In this exercise, we measured the similarity between the original and generated images using perceptual distance, a loss function that compares high-level features extracted from a pretrained CNN (in our case, the VGG16 network). Perceptual distance captures structural and semantic differences between images (in contrast with pixel-wise distance).

To ensure that the generated image appears “natural” and consistent with the training data distribution, we apply regularization, constraining the latent vector’s deviation from the mean of the latent space. We reconstruct the image by searching through the latent space using gradient descent, minimizing the perceptual loss (with regularization) of the image generated by a pre-trained GAN using the latent vector.

Algorithm

The latent space – StyleGAN has multiple latent spaces that could be embedded. The Z space is the initial latent space, the W space is the intermediate latent space after the fully connected mapping. Previous research[1] have shown that embedding in the early stages doesn’t convey sufficient information to the generator about the given image. Therefore, our gradient descent optimization is performed in W space, which offers a more expressive and effective encoding.

Noise – In StyleGAN, image generation is not purely deterministic—it incorporates stochastic noise at different levels to create fine details. These noise inputs are not part of the latent space (w) but are instead added separately at different layers of the synthesis network. Each layer of the synthesis network has noise inputs, referred to as "noise buffers", which contain randomly initialized noise tensors.

Image reconstruction:
Input: an image I, number of GD iteration T, latent distance regularization factor W, a pretrained generator G.

[1] Abdal, Rameen & Qin, Yipeng & Wonka, Peter. (2019). Image2StyleGAN: How to Embed Images Into the StyleGAN Latent Space?. 10.48550/arXiv.1904.03189.

1. approximation of the intermediate latent space:
    a. Generate 10,000 random latent vectors sampled from $[0,1]^d$.
    b. Map the sampled vectors from Z space to W space – the intermediate latent space. Approximate the standard deviation and mean of the W latent space.
2. Extract noise buffers from StyleGAN, which help control fine details in image generation.
3. Load a pretrained VGG16 network for computing perceptual loss.
4. Prepare the target image by resizing and extracting features using VGG16.
5. Initialize the latent vector (w_opt) from the mean latent (w_avg) and set up the optimizer (Adam).
6. Optimization loop – repeat for T iterations:
    a. Adjust learning rate according to the current iteration.
    b. Generate an image from w_opt using the generator.
    c. Apply degradation (e.g., blur, grayscale, or inpainting) to the generated image.
    d. Compute the perceptual loss between the target and generated images, plus W times the distance between w_opt and w_avg.
    e. Optimize w_opt and noise buffers using gradient descent.

Image blurring – blur the image using a fixed kernel size and sigma.
Image inpainting – delete a rectangle with a fixed size from a specified location in the image.
Image gray-scaling – turn the image to grayscale using the following formula: 0.299 * R+ 0.587 * G+ 0.114* B.

Results

Results of 3.1

Following the Grammy Awards, the image I chose to work with is of Beyoncé :) You can see that the algorithm accurately detects Beyoncé's face in the image.

Beyonce – Original Image

Beyonce – Aligned Image

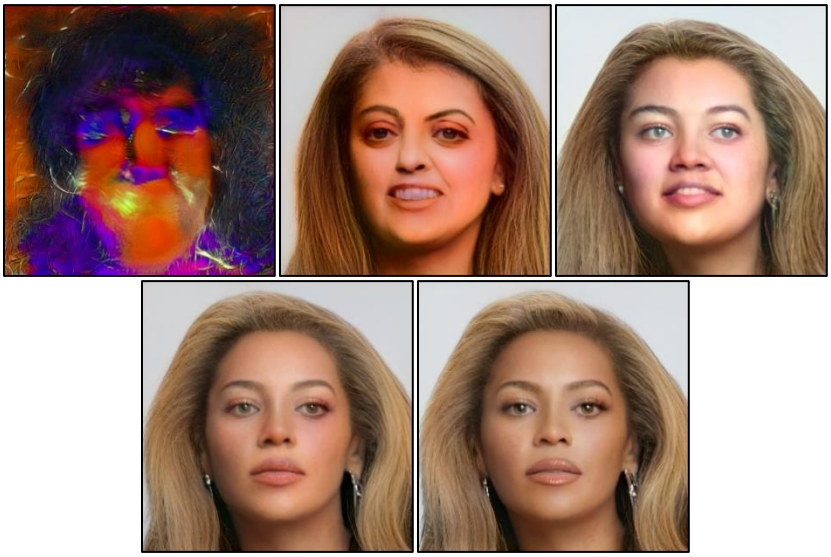
Results of 3.2

Optimization process: Below, you can see the image reconstruction process using gradient descent. Initially, the algorithm starts with a random latent vector, resulting in a blurry and unrecognizable image. In the early stages, the algorithm captures general features such as the head position, hair

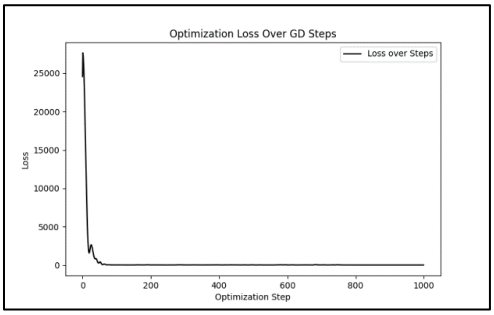
shape, and jawline. Only in the later stages does it refine the details, gradually replacing the facial features with ones that match the original image.

Some details remain inaccurate, such as the lashes or the eye color, which appears green instead of brown. This could be due to local minimum convergence or the mean-distance regularization, which may bias the reconstruction toward features more common in the training data.
The original image:

Algorithm results by iteration – 0, 100, 200, 400, 1000



Loss plot: The loss function converges toward zero, initially high but decreasing as general features are captured first, followed by fine detail refinement.



Bad result:

This time, I attempted to reconstruct an image of Morgan Freeman wearing a hat. While the lower half of his face is accurately reconstructed, the upper part, particularly around the hat, appears distorted. This distortion probably occurs since the hat obscures key facial features (the eyes and forehead).

The effect of regularization and number of steps:
To explore the effect of latent_dist_reg_weight I multiplied and divided the default weight by 1500. The results are shown below.

Left to right – high, default and low regularization weights

Increasing the latent distance regularization weight pulls the reconstruction closer to the model's latent mean, resulting in a more generic face that may lose key identity features, as seen in the left image, which no longer resembles Beyoncé. Lowering the weight allows greater deviation, capturing more unique details. While the difference here is subtle, the right image shows slight asymmetry in the lips, more prominent shadows and a darker left eye.

The number of optimization steps affects image accuracy—fewer steps might lead to under-optimization, as the model hasn't fully minimized perceptual loss. More steps improve detail and alignment with the target image, but beyond a certain point, the improvement might not be significant to notice it.

Results of 3.3.1

Left to right – original, blurred original, reconstruction

My solution: The algorithm takes a blurred image and attempts to reconstruct it using the generative algorithm described above. In my approach, during the generation process, I progressively blurred the intermediate images to better compare them to the original blurred image. The larger the kernel used for blurring the intermediate images, the less blurred the generated image appeared.

Hyper parameter analysis: Since the algorithm had already converged to approximately the same solution around iteration 500, I decided not to change the number of iterations and instead adjusted only the regularization parameter. This time, unlike in the first part, I increased and decreased the parameter by a factor of 100 to observe the effect on the results. The high regularization results resulted in distorted features (e.g. the lips), yet better skin texture. Surprisingly, lower weight resulted in worse result as well (distorted teeth), therefore I chose the default weight as the regularization hyper parameter.

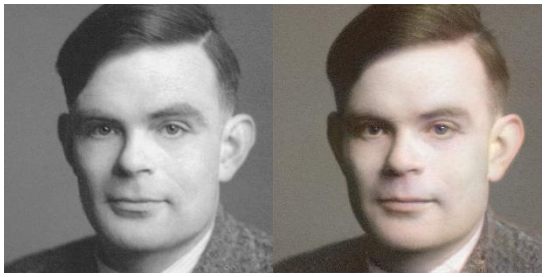
The effect of the blurring kernel size: Increasing the kernel size makes the generated image less blurred, as the texture of the generated image has less influence on the loss function between the target image and the generated image. This effect can be clearly seen in the following images:

Kernel sizes – left to right: 0, 9, 17, 25

Results of 3.3.2

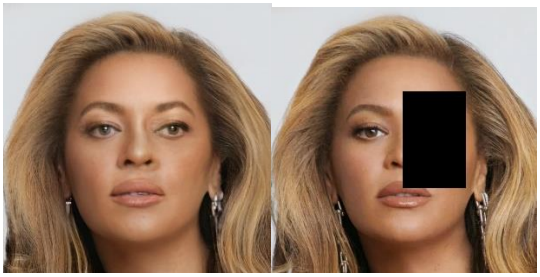
Results of the inpainting:

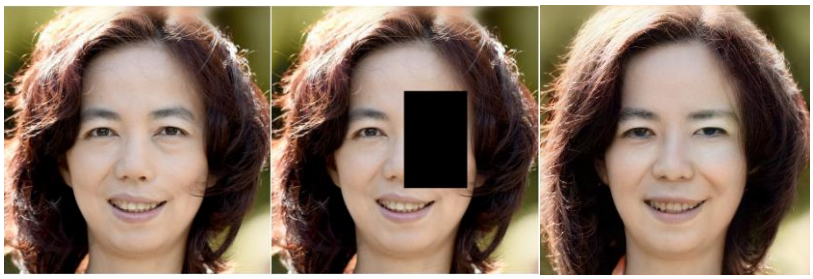
Alan Turing:



In this part of the exercise, I slightly increased the regularization on proximity to the average to enhance color realism. As a result, the colors became more natural, with fewer inconsistencies. I experimented with different weights—0.1 provided the best balance. A higher weight produced a more realistic image but deviated from the original, while a lower weight led to noticeable color inconsistencies.

Results of 3.3.3





Left to right – original, masked and reconstructed

My approach was straightforward, as the default hyperparameters worked well, requiring no adjustments. However, in Beyoncé's case, despite experimenting with different regularization values and iteration counts, the result was still not perfect.

Conclusion

Thanks for this interesting exercise and for the course! 😊