COMP527/ELEC519, Spring 2023 - Programming Assignment 5: Text-driven Image Manipulation

Due date: Thursday, 01-6-2023, 11:59 PM.

Overview

The purpose of this assignment is to explore the use of deep generative models for manipulating images via textual descriptions. In particular, you will be using a pretrained StyleGAN2 [1] to synthesize human face images and implement StyleCLIP-LO [2] (Latent Optimization) method to manipulate images. An example manipulation can be found in Fig. 1.

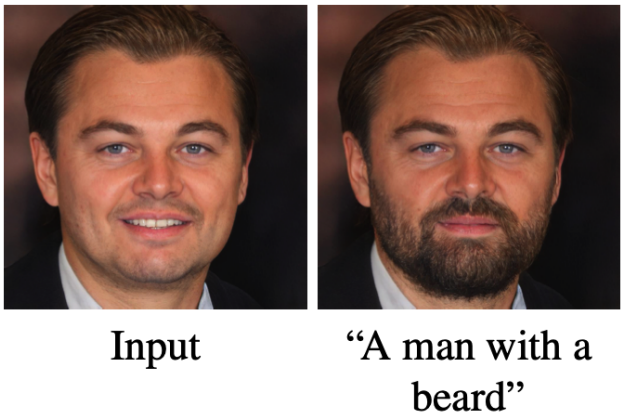

Figure 1: An example manipulation using StyleCLIP-LO

Setting Up

We recommend that you use Colab (`https://colab.research.google.com/`) for the assignment, as all the assignment notebooks have been tested on Colab. The assignment involves a basic understanding of PyTorch. If you are unfamiliar with PyTorch, you may want to take a look at the tutorials here: `https://pytorch.org/tutorials/beginner/basics/intro.html`.

From the assignment zip file, you will find the python notebook file named `assgn5.ipynb`. To setup the Colab environment, you will need to upload the notebook file using the upload tab at `https://colab.research.google.com/`. The notebook contains detailed descriptions for each part you will implement, please read through the PDF and the notebook carefully. We mark in red questions you should answer in your submitted report.

1. Getting Started with StyleGAN2 (15 points)

As you saw in the lecture, StyleGAN2 is a powerful model that can synthesize highly realistic looking images. It is built upon the StyleGAN's Style-based architecture [3], which is depicted in Fig. 2. Here, the the initially sampled latent codes are passed through a mapping network, which converts them to an intermediate latent space called the $\mathcal{W}+$ space. The generator network consists of 9 stages and in each stage, 2 latent codes from the $\mathcal{W}+$ space are used to control the style of the synthesized images. Therefore, the latent codes in the latent space have shape [B, 18, 512] where B is the batch size.

Truncation Parameter (5 points)

StyleGAN2 also uses the *truncation trick*, which you will be exploring in this part of this assignment. In the assignment notebook, the functions to sample latent codes and displaying them are given

to you. The truncation value is set to 0.7 by default. Your task is to play with the truncation variable in the sampling function and observe the outputs. How does the samples change when you increase or decrease the truncation variable? What does the truncation variable control? Please provide samples with various truncation values in your report and explain your observations.

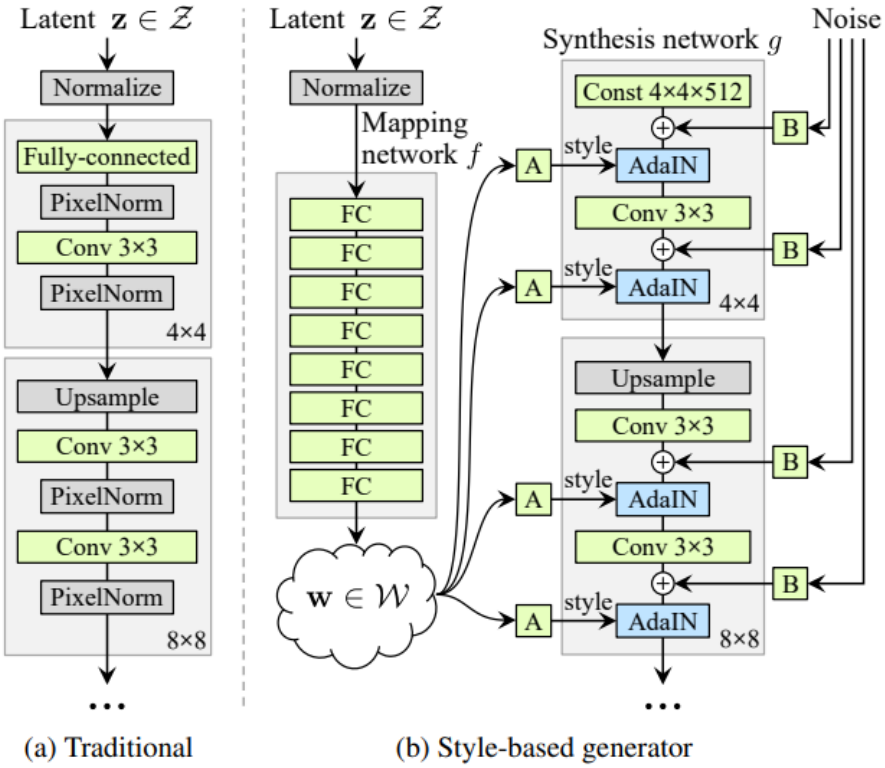

Figure 2: StyleGAN architecture

Style Mixing (10 points)

In this part, you will be exploring the concept of Style Mixing. As mentioned above, there is a mapping network that maps the initially sampled latents to an intermediate latent space called the $\mathcal{W}+$ space. In the $\mathcal{W}+$ space, one latent has the shape [1, 18, 512] and these 18 vectors (512 dimensional) are fed to the generator at different stages to control the style of the synthesized image. Let's call each of these 512 dimensional vectors a **style vector**.

In style mixing, we mix these latent codes, such that some of these 18 vectors are coming from one sample and the rest of them are coming from another sample. In this part, you will observe the effects of style mixing by mixing two latents by different amounts. An example of style mixing is given in Fig. 3. Here, we mixed the first 4 style vectors from w_1 and the last 14 style vectors from w_2.

Here's what you need to do:

- Sample 2 latents from StyleGAN2 (don't forget to return the latents too). Let's call these latents w_1 and w_2
- Display the 2 corresponding images side by side (you can do this by concatenating the samples among the last dimension and using the `tensor2im()` function)

- Mix the latents at various amounts:
  - First 1 style vector from w_1 and the remaining 17 style vectors from w_2
  - First 4 style vectors from w_1 and the remaining 14 style vectors from w_2
  - First 8 style vectors from w_1 and the remaining 10 style vectors from w_2
  - First 12 style vectors from w_1 and the remaining 6 style vectors from w_2
  - First 1 style vector from w_2 and the remaining 17 style vectors from w_1
  - First 4 style vectors from w_2 and the remaining 14 style vectors from w_1
  - First 8 style vectors from w_2 and the remaining 10 style vectors from w_1
  - First 12 style vectors from w_2 and the remaining 6 style vectors from w_1
- Visualize each of the mixed latents by feeding them to the generator and synthesizing the corresponding image.

Figure 3: **An example of style mixing.** Here, the first 4 style vectors are coming from w_1 and the remaining 14 are coming from w_2.

What are your observations? How does the outputs change when you mix the latents at different stages? Which original image does your outputs look like when you mix at the first stages or the last stages? What kind of semantic features each of these dimensions control? Make sure to include your outputs and your observations in your report.

2. Editing Images with StyleCLIP-LO (55 points)

Now that you are familiar with StyleGAN2 and synthesizing images, it is time to explore the latent space of StyleGAN2 to manipulate images. StyleCLIP-LO optimizes the initial latent code, such that when the final latent code is fed to the generator, an image with certain attributes mentioned by a textual description is synthesized. An overview of this is depicted in Fig. 4. In this approach, there are no learnable modules, as the latent code is directly updated in each iteration until we reach the desired number of iterations. You will be given an initial latent code w_{init}. When you pass this latent through the generator, it generates the image x_{init}. We want to optimize the initial latent, such that when we feed the optimized latent $w_{optimized}$ to the generator, the output image $x_{optimized}$ possesses some attributes that we desire, such as the purple hair in Fig 4. To supervise this optimization procedure, you will be using 3 different loss functions. The following subsections describe the details of each of these losses. You may also want the check the corresponding section from StyleCLIP (Section 4) for more details.

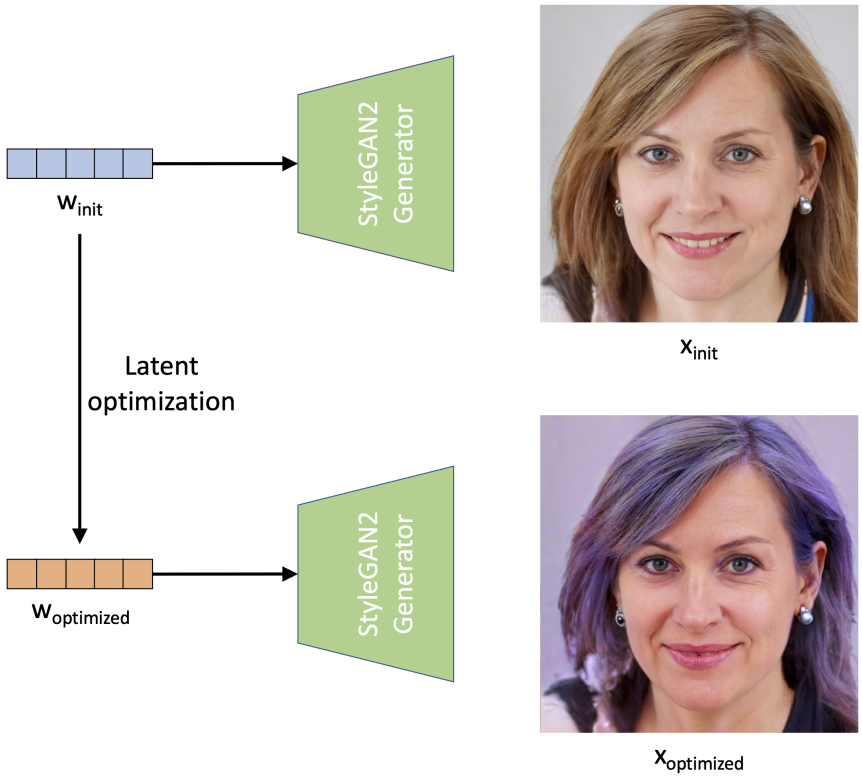

Figure 4: Latent optimization.

CLIP Loss (10 points)

Contrastive Language-Image Pre-training (CLIP) [4] is a multi-modal approach that maps images and textual descriptions in the same embedding space. You can calculate the similarity between an image and text in the CLIP space based on their embeddings. Complete the implementation of the CLIP loss in the notebook. To do this, first you need to pass the image through the preprocessing layers (upsample and average pooling). Then, you will pass the image and the text through the CLIP model to obtain the semantic similarity of them. As we want to maximize the similarity, we will subtract this from 1 and minimize the obtained loss value. Check the notebook for more details about the implementation.

ID Loss

When manipulating images, we ideally want the identity of the original person to be preserved. For this purpose, you will be using a pre-trained neural network, namely ArcFace [5], that measures the identity similarity between two images of human faces. The initialization of this ArcFace model and the ID loss function is given to you in the notebook. Inspect the code and understand how it works, you will need to use it in your optimization process. For this part, you do not need to do any implementation, simply execute the given cell.

L2 Loss (5 points)

Similar to the identity loss, we do not want to manipulate the ”unwanted features”, i.e., the features of the input image that are not mentioned in the textual description. To achieve this, you will be implementing a L2 loss between the initial latent code and the optimized latent code. This loss ensures that we do not deviate too much from the initial latent, preserving the similarity between

the sythnesized images. Implement the L2 loss in the notebook. You will simply take the difference between the two latents, square it and sum all the entries in the resulting vector to obtain the loss value.

Latent Optimization Procedure (40 points)

Now that you have implemented the necessary loss functions, it is time to complete the latent optimization loop and edit images using this function. Complete the implementation of the `run_optimization()` function in the notebook. The details on how to implement this function is given as comments in the notebook. You will initialize the optimizers and the loss functions. In the optimization loop, you will update the latent code using the loss values you calculate. If your implementation is correct, you should be obtaining results similar to Fig. 5.

Figure 5: Results for latent optimization using description *A man with a beard.*

3. Coarse-Medium-Fine Latent Analysis (30 points)

As described in Problem 1, the intermediate latent codes we are using are of shape [1, 18, 512]. Each of the 18 vectors are used to control the generation at different stages. These latents are grouped into coarse (1-4), medium (5-8) and fine (9-18) latents. In this part, your task is to chunk the initial latent code into these three groups. After the grouping, run 3 separate optimizations where you optimize only the corresponding chunk. In this part, you can copy and adapt your implementation of `run_optimization()` as most of the code will be the same. Discuss your observations. How does the outputs change when you optimize different chunks of latents. How are the results different than optimizing all the chunks together? What level of detail each chunk controls? Furthermore, try another description, such as *A woman with heavy makeup.* Show the results for both descriptions and discuss them.

4 - BONUS. Edit your own photo (30 points)

In this part, you will manipulate your own image using the same latent optimization procedure you implemented. To do this, you first need to find a latent code that corresponds to your face in the latent space of StyleGAN2. There are many approaches to map real faces to the latent space of GANs, and this process is called GAN Inversion. Here, we will share a pretrained model with you so that you can obtain the initial latent code w_{init} corresponding to your face.

Use the setup code to load the model and invert your own image to the latent space. As you will observe, the inversion network is not perfect, and might cause slightly identity changes. However, it is good enough the capture the semantic features from the input image. After inversion, use the function you implemented to edit your own image. You may experiment with any textual description you wish. Make sure to include at least 3 different manipulation results with 3 different descriptions. We will organize a competition to select the best manipulation examples among the ones you submitted. Make sure to indicate which result is your nomination for the competition. You may experiment with creative prompts.

What to Hand In

Your submitted solution should include the following:

- A PDF report explaining what you did for each problem, including answers to all questions asked throughout Problems 1, 2, 3 and 4. The report should include any figures and intermediate results that you think may help. Make sure to include explanations of any issues that you may have run into that prevented you from fully solving the assignment, as this will help us determine partial credit.
- The codes you will submit should be well commented and in the form of a Jupyter notebook.
- Image files for Problems 1-4, showing the results you obtained.

You should prepare a ZIP file named `username-assgn5.zip` containing your solution / implementation `assgn5.ipynb` and your report `assgn5-report.pdf`, and submit it to Blackboard.

Late policy

You may use up to 7 grace days (in total) over the course of the semester for the assignments. That is, you can submit your solutions without any penalty if you have free grace days left. Any additional unapproved late submission will be punished (1 day late: 20% off, 2 days late: 40% off) and **no submission 2 days after the original deadline will be accepted.**

Academic Integrity

All work on assignments must be done individually unless stated otherwise. You are encouraged to discuss with your other classmates about the given assignments, but these discussions should be carried out in an abstract way. That is, discussions related to a particular solution to a specific problem (either in actual code or in the pseudocode) will not be tolerated. In short, turning in someone else's work, in whole or in part, as your own will be considered as a violation of academic integrity. Please note that the former condition also holds for the material found on the web as everything on the web has been written by someone else.

Bibliography

[1] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of StyleGAN. In *Proc. CVPR*, 2020.

[2] Or Patashnik, Zongze Wu, Eli Shechtman, Daniel Cohen-Or, and Dani Lischinski. Styleclip: Text-driven manipulation of stylegan imagery. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2085–2094, October 2021.

[3] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks, 2019.

[4] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021.

[5] Jiankang Deng, Jia Guo, Jing Yang, Niannan Xue, Irene Kotsia, and Stefanos Zafeiriou. ArcFace: Additive angular margin loss for deep face recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(10):5962–5979, oct 2022.