Image Processing – 67829 – Exercise 5

Deep Learning - Deep Style Image Prior

Due Date: 6.2 at 23:59

Version 1 - Last Update 15.01.2025

Note: You will be implementing the exercise using Google Colab. Colab is an IPython notebook running on Google cloud servers and provides you with GPU. Please make sure to follow the instructions provided in the following Colab Notebook (Use HUJI user - open in Colab). As neural networks use GPU for fast and efficient computations, it is strongly advised you make sure you are training using a GPU and not a CPU, more on this in the notebook.

Note 2: When initially opening the notebook there should be a text to the right of the "Help" menu saying "Changes will not be saved". To ensure you can make changes to the notebook save a copy of it to your own drive and work on that one. You can do that by going to "File" -> "Save a copy in Drive".

Failing to do so will result in code loss!

Make sure you are the only one that has access to it!

1 Overview

In this exercise you will be using *StyleGAN2* [2] to perform different image reconstruction tasks. The exercise will familiarize you with the concepts of *StyleGAN2*, *GAN Inversion*, *Latent Optimization*, and *Image Priors*. The exercise is based on the work of Gabbay et al. [1] and you are encouraged to read it in case you wish to dive deeper.

2 Background

2.1 StyleGAN2

Generative adversarial networks (GANs) have been shown to generate images of remarkable quality and photorealism. A line of work called *StyleGAN* took the generation capabilities to new levels of realism. The website "This Person Does not Exists" demonstrates these capabilities by sampling new images from StyleGAN2 at each page refresh.

As you have seen, a GAN takes as an input random noise and is trained to generate an image. Here as well, StyleGAN2 takes in a random noise vector z and generates an output image which we will denote with I_{gen}. We call the noise vector z a "*latent vector*" and the space it is sampled from the "*latent space*".

The StyleGAN2 you will be using has been pre-trained on the FFHQ dataset, which consists of $70,000$ images of human faces at 1024×1024 resolution and contains variation in terms of age, ethnicity and image background. It also has good coverage of accessories such as eyeglasses, sunglasses, hats, etc. All the images are preprocessed and aligned so that facial parts locations are consistent across the dataset, as you will see later in the exercise, using this preprocessing pipeline is crucial for good results in this exercise.

2.2 GAN Inversion and Latent Optimization

2.2.1 Latent Optimization

In class you have seen the standard process in which neural networks are usually trained. The *network weights* are optimized by running some variant of Gradient Decent and Back Propagation (i.e the differentiation is done with respect to the weights of the network). In this exercise we use a slightly different setup, rather than optimizing the network weights, in Latent Optimization **the latent vector itself (i.e the input of the GAN) is optimized while the weights of the generator remain unchanged.** That is, at each step the differentiation is done with respect to the latent code itself. This could be interpreted as finding the best latent code that minimizes the objective loss. In this exercise we will use *Latent Optimization* to perform *GAN Inversion* which we will cover next.

2.2.2 GAN Inversion

The task of GAN Inversion sets to "invert" a generator to the latent vector that generates a given a image. That is, given an input image I_{in} and a **trained** generator $G(z) = I_{gen}$, the goal is to find a latent vector $z \in \mathcal{Z}$ that can generate I_{in}. What this essentially means is that we are trying to "encode" the input

image into the latent space of the GAN. Formally, we are looking for

$$z^* = \arg\min_z \|\phi(G(z)) - \phi(I_{in})\|_2 \tag{1}$$

where ϕ is some function that transforms the image into a feature vector (i.e perceptual feature extractor), you may ignore the actual details of it in this exercise.

2.3 Generative Models as Image Priors

We will now see how to use GAN Inversion to search for the optimal latent vector that reconstructs the input image according to the domain of the training dataset.

Throughout the course you've seen many methods for image processing and reconstruction. The common ground between many of these methods is the use of some assumptions that act as a prior in order to manually tailor a solution for the problem. For example encouraging smoothness, assuming lines stay straight, etc. However, these models assume properties of natural images without modeling their underlying domain, as such, they often fail to faithfully reconstruct details. In this exercise we treat the trained GAN as having prior knowledge on the training domain. To this end, we treat the images the generator is able to generate as though they came from the target domain. In contrast, the images it can't generate are considered out of domain. To turn this binary prior into a loss we will use Eq. 2

$$\ell_{prior} = \min_z \|\phi(G(z)) - \phi(I_{in})\|_2 \tag{2}$$

where ϕ is defined as in Eq. 1.

Essentially, what this means is that smaller values of Eq. 2 imply an image that comes from the domain the generator was trained on, while large values imply out of that domain. That is, the smaller the value, the more likely it is to look natural. Hence we can use Eq. 2 as our loss function.

In the next section we will define the specific tasks you will need to solve.

3 The Exercise

Equipped with the understanding of the previous section, we will now define the specific tasks we wish to solve. As mentioned above, we will be performing image reconstruction using the above method.

3.1 Image Alignment

As mentioned above, StyleGAN2 was trained on FFHQ, each training image was preprocessed to be aligned so that the facial parts are in the same location in each image. Since GANs are trained to

model the training domain which in our case was preprocessed as above, all images you use should also be preprocessed with the same script. Run the image alignment script on an image of your choice and include the before and after examples in your submission PDF. In the next sections, make sure all the images you use have been aligned before you use them.
For this section, in your PDF include the following:

- The results of the image alignment on some image of your choice (before and after images).

3.2 GAN Inversion

In this section you will perform *GAN Inversion* as described above.
Go over the code, see that you understand it. In particular, understand the GAN Inversion part and see that it indeed implements Eq. 1 and Eq. 2. Remember, the inversion process optimizes z while the generator never changes.
Next, run the code with an image of your choice and observe the optimization process, play with the number of steps (*num_steps*) and the regularization weight on the distance of the optimized latent from the mean latents (*latent_dist_reg_weight*). What *latent_dist_reg_weight* essentially does is to regularize the distance z can be from the average latent, this corresponds to how far the generated image can be from the mean of natural images the GAN was trained on. See that you are able to reconstruct the image faithfully. Examples of a good and bad inversions can be seen in Fig. 1.
For this section, in your PDF include the following:

- A figure with the optimization progressions with over 5 images. That is, the image generated by the initial z, 3 other images from the optimization process that showcase the progression, and the final optimized image. Include the original image as well.
- A plot of the optimization loss.
- Discuss the effect of *latent_dist_reg_weight* and *num_steps* on the final results.

Note: You may need to try several images before finding the types of images which are well reconstructed. Keep in mind there may be some biases in StyleGAN2. Also, recall that it was trained on FFHQ, a dataset of high quality images taken in good conditions.

Figure 1: On the left you can see an example of a bad reconstruction, on the right an example of a "good" reconstruction (although it could be done better, it is not needed).

3.3 Image Reconstruction Tasks

Recall that Eq. 2 aims to reconstruct the original input image by finding a z that results in an image as similar to the input image as possible. However, in the following sections you have an image that underwent some form of degradation (blur, decolorization, etc.). Our goal is not to reconstruct the degraded image but to reconstruct a "fixed" image. To achieve this you will need to modify Eq. 2 so as to "trick" the inversion process to think the images it is producing are also degraded. This is done by applying the degrading function on the generated image before computing the loss. That way, the generator outputs "good" images, but the loss is computed as if they are degraded.

3.3.1 Image Deblurring

In the task of image deblurring we take as an input a blurry image and return a deblurred version of the image. The deblurred version should be semantically meaningful as seen in Fig. 2, yet it does not need to be identical to the original image.
When creating the blurred image, try to use different kernel sizes, your kernel should result in a blurry image that can also be deblurred (meaning, it should not be too strong to the point the debulrring fails but not too weak that no deblurring can be seen).

For this section, in your PDF include the following:

- The results of the deblurring for an image of your choice (original, blurred, deblurred). Also, include this image and the latent npz file responsible for the reconstruction result alongside your submitted code (the npz file is a zipped format of np files which are used to save numpy arrays, in the notebook

there is already an example of saving the latents to an npz file).

- Your results of the deblurring on the given blurry image of Yann LeCun (the blurry image from Fig. 2 - you can find it in the Moodle in a zip called "Ex5 Supplementary"). Also, include this result and the latent npz file responsible for the reconstruction result alongside your submitted code.
- Discuss your solution, explain what you did and the motivation for it.
- If you ran into any issues, elaborate on them and how you solved them.
- A short analysis of the effect of the blur kernel size (with examples!) and discuss how the different hyper parameters effect this task.

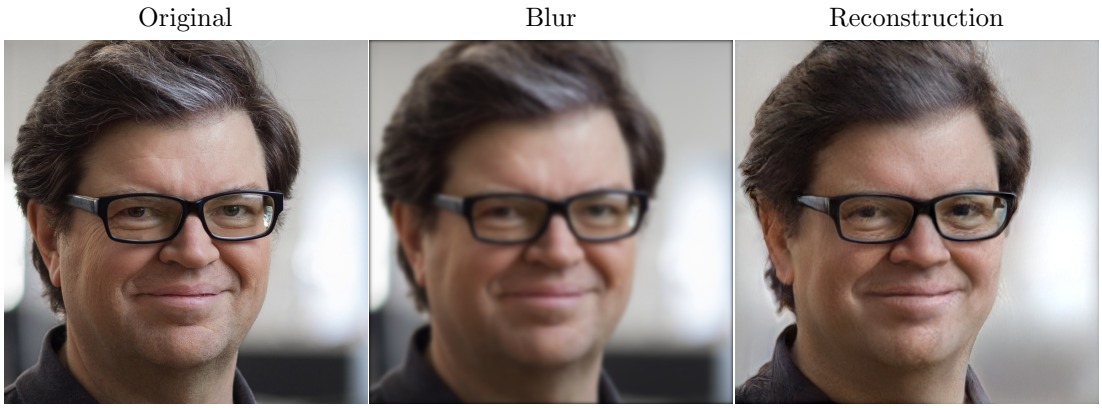

Figure 2: Image Deblurring

3.3.2 Image Colorization

In the task of image colorization we take as an input a grayscale image and return a colored version of the image. The colored version should be semantically meaningful as seen in Fig. 3 (i.e be colored in real colors a human face may have and not be for example, green).
For this section, in your PDF include the following:

- The results of the colorization for an image of your choice (original, grayscale, colorized). Also, include this image and the latent npz file responsible for the reconstruction result alongside your submitted code.

- Your results of the colorization on the given grayscale image of Alan Turing (the grayscale image from Fig. 3). Also, include this result and the latent npz file responsible for the reconstruction result alongside your submitted code.
- Discuss your solution, explain what you did and the motivation for it.
- If you ran into any issues, elaborate on them and how you solved them.

Figure 3: Image Colorization examples. Notice how the bad example has both unrealistic colors, but also a different facial geometry.

Note: You may decide to use an RGB image, in which case you **MUST** first convert it into a grayscale image, and colorize that image.
Note 2: Your grayscale images should have 3 channels (i.e each channel is a copy of the one channel), and not 1 channel as we've seen throughout the course. This is so that image features can be extracted from the images using the perceptual feature extractor (ϕ in Eq. 1).

3.3.3 Image Inpainting

In the task of image inpainting some portion of the image is missing and we wish to reconstruct it (think of a mask over part of the image). The resulting version should be semantically meaningful as seen in Fig. 4 (i.e both the original area and the area which was reconstructed should contain semantically meaningful details that are consistent with each other).
To inpaint an image you can multiply your original image with a mask image that "erases" some part of the image.

- The results of the inpainting for an image of your choice (original, masked, inpainted). Also, include the image and the latent npz file responsible for the reconstruction result alongside your submitted code.
- Your results of the inpainting on the given masked image of Fei-Fei Li (the masked image from Fig. 4). Also, include this result and the latent npz file responsible for the reconstruction result alongside your submitted code.
- Discuss your solution, explain what you did and the motivation for it.
- If you ran into any issues, elaborate on them and how you solved them.

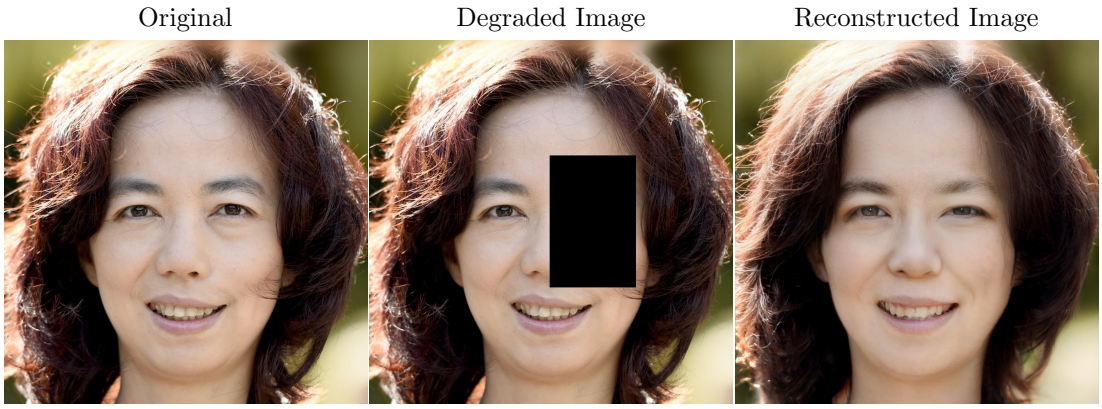

Figure 4: Image Inpainting examples

4 Submission

Submission instructions are given in the "Submissions Guidelines" document published on the course web page (here). Please read and follow them carefully. Any updates to those guidelines will be posted in the news forum, so be sure you follow the forum.

You must submit your code but there is no API you need to follow and there are no presubmission tests.

4.1 Report Guidelines

In addition to the code, you should submit a report describing your solution. The report must follow the following structure and address the topics below. We provide an English and Hebrew Google Docs

template (you need to copy it to use it). In case you choose not to use it, please maintain a similar structure (font size, same sections, same number of figures, same number of pages, etc.), in particular, the report should be no longer than 6 pages and include the following sections and topics:

1. Introduction
   (a) In your own words, state the goal of the exercise and what were the main techniques (i.e. an idea or concept you've learned in class, not a technical tool like numpy) you've used to solve it.
2. Algorithm
   (a) Clearly describe the algorithm used to perform the reconstruction (the general algorithm, not a specific reconstruction type) (i.e. describe the conceptual steps and building blocks, e.g., if the algorithm is how to make coffee, the main algorithm can be using a coffee maker to make coffee and foamed milk).
3. Results
   (a) For each sections 3.1, 3.2 and 3.3.1 to 3.3.3 include the required things requested by that section (there is a short bullet list at the end of each section). You should also explain in writing your visualization. If you have tried different hyper-parameters or algorithms, compare them.
4. Conclusion
   (a) Summarize your key findings and insights.

Your final submission should have a PDF and a zip file that contains the **(i)** All the images you choose, their degraded versions, the reconstruction, and the corresponding latent files **(ii)** The results from sections 3.3.1, 3.3.2, 3.3.3 with their corresponding latent files **(iii)** The notebook with your code.

Note: The PDF and zip should be submitted to the respective submission in the Moodle.

5 Grading

Your exercise will be graded based on a manual inspection of your report (and code). As mentioned above, there will be no presubmission tests and no automatic tests.

General tips for getting the most out of Colab:

- Colab provides a few hours of free GPU time every day, this may vary depending on the amount you have been using it and the demand they currently have. If you see your time is running out too quickly you can purchase a monthly colab subscription which gives you access to stronger GPU for more time.
- You can only connect to 1 Colab runtime per account at any moment, however, you can copy the notebook to your none HUJI account and run it from there, thus gaining two GPU instances.
- Inverting a single image should not take long, if it is taking you much more than 20 minutes for 1000 steps you should take a closer look at your implementation and ensure you are using the GPU.

- Colab might disconnect the runtime, in such cases, the information saved in the notebook may be deleted, due to that it is recommended you save any important information (i.e plots, trained models, etc.) locally or within your Google Drive. Moreover, downloading the code may take a few minutes, you can easily modify your notebook to save the downloaded code in your google drive to save this time.
- **In addition to your PDF you should submit a zip with your notebook (in Colab, go to "File" -> "Download" -> "Download .ipynb"), the image files for all the results (i.e the original image, the degraded one, and the reconstructed one), and the latent npz file for each reconstructed image.**
- For the best results, your input images (after running the alignment script) should be 1024×1024. However, for development purposes it might be faster to work on smaller images (256×256 or 512×512 should do the job).

Good luck and enjoy!

References

[1] Aviv Gabbay and Yedid Hoshen. Style generator inversion for image enhancement and animation, 2019. 1

[2] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan, 2020. 1