Contents

1 ex5/README

```
muaz.abdeen

files in the ZIP:
1- Ex5_results.pdf
2- IMPR_Ex5_Deep_Style_Image_Prior_2021_2022.ipynb
3- ./Deblurring/im1_k85_w0.001
   ./Deblurring/im2_k75_w0.1
   ./Deblurring/Yann LeCun
4- ./Colorization/im1_s900_w1
   ./Colorization/im2_s900_w1
   ./Colorization/Alan Turing
5- ./Inpainting/paint_im1_s1000_w0.2
   ./Inpainting/paint_im2_s1000_w0.2
   ./Inpainting/fei_fei_li

Each one of the 3-5 sub-folders contains 4 files:
1- inverted_latent.npz
2- original_image.png
3- original_degraded_image.png
4- final_inverted_image.png
```

67829 – Image Processing – Fall 2021

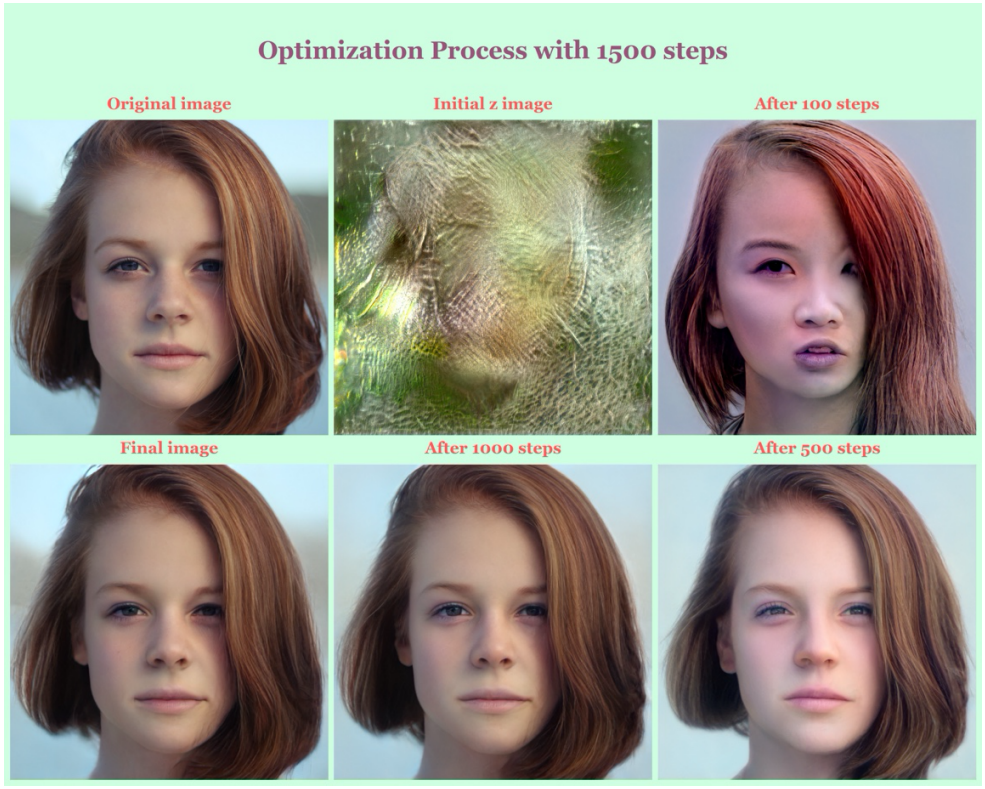
Exercise (5)

Muaz Abdeen – 300575297

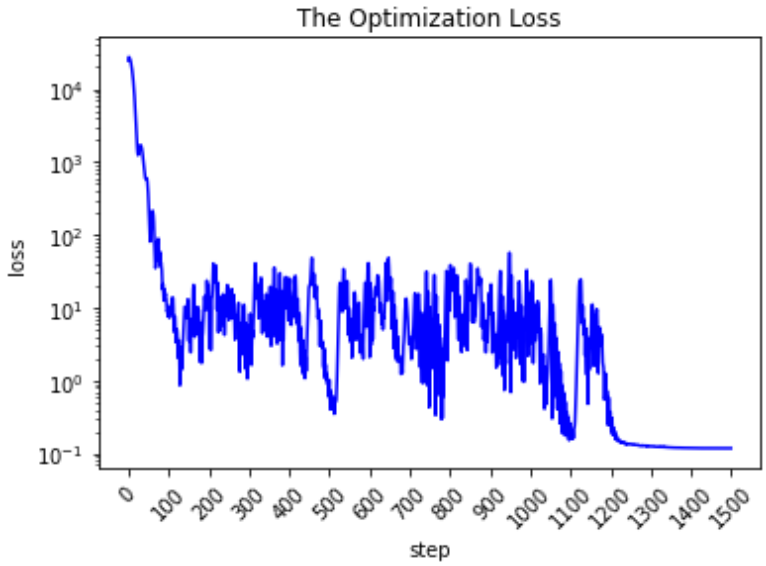
Image Alignment:

Before	After
	

The Optimization Progressions:



A Plot of the Optimization Loss:



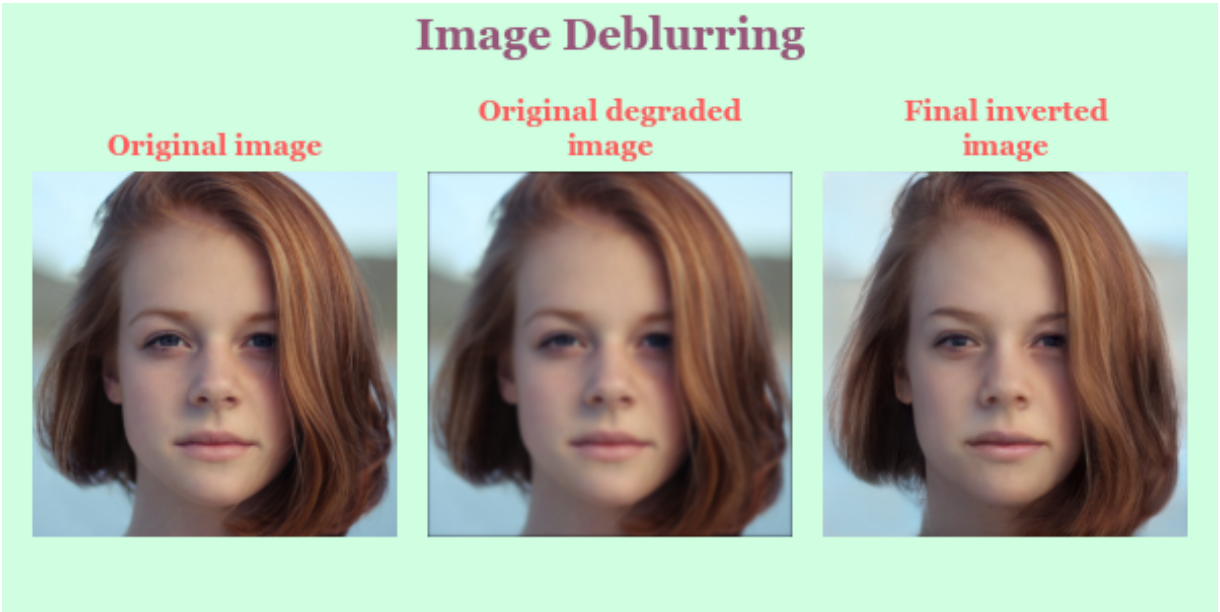
The effect of *latent_dist_reg_weight* and *num_steps* on the results:

I noticed that given a *num_steps*, increasing the *latent_dist_reg_weight* made the optimization process generates good results earlier, that is, closer images to the original one will start to show earlier.

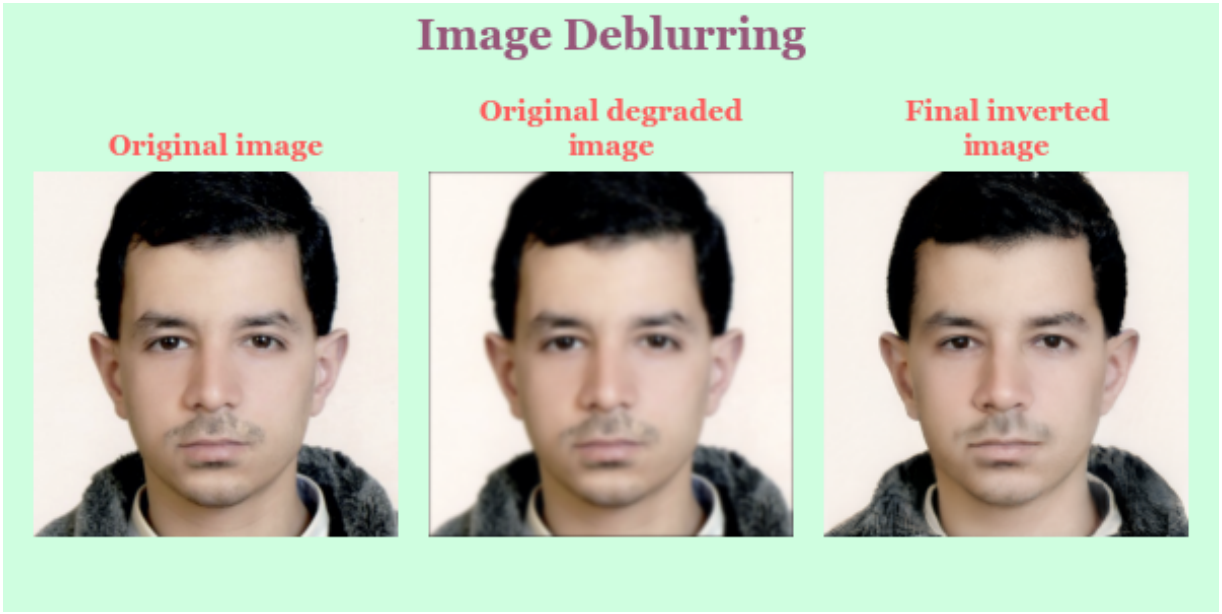
Also, increasing the *num_steps* will give us a better result, to some extent, that is after some threshold, there will be almost no improvement.

Image Deblurring:

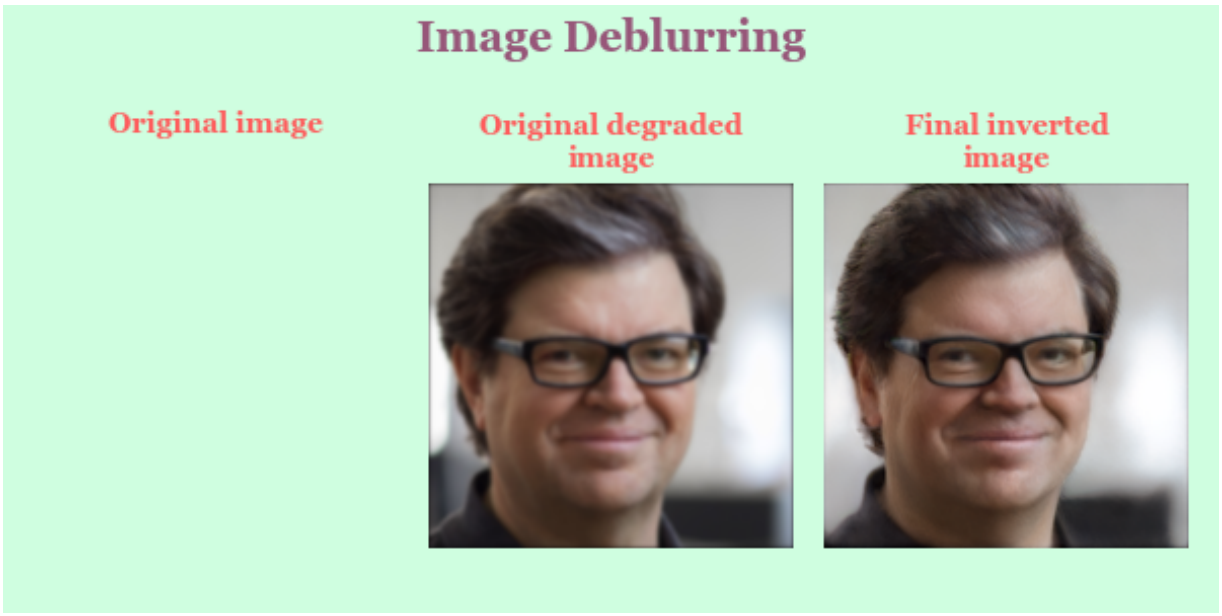
First image



Second image (my personal picture):



Yann LeCun



Solution Discussion:

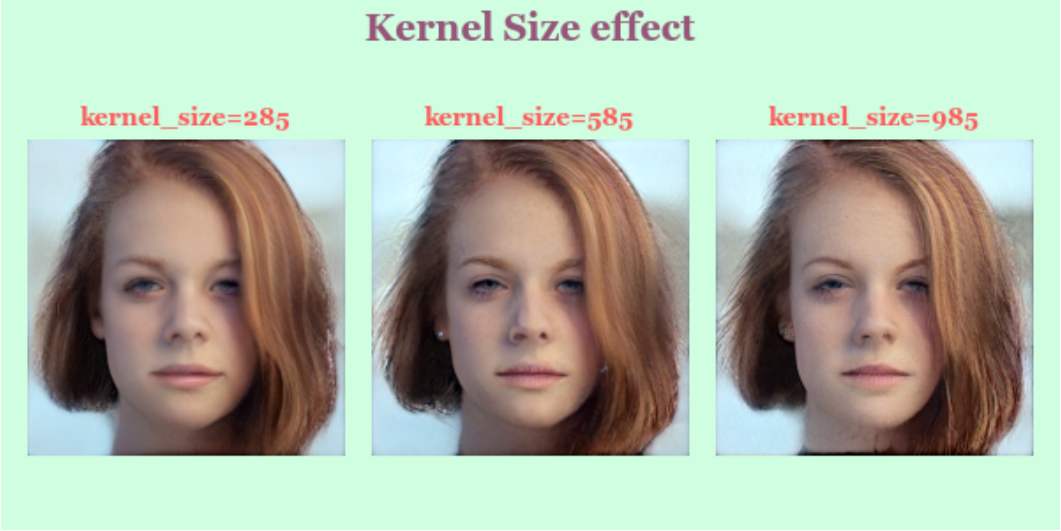
Hyperparameters: $kernel\_size = 85$, $num\_steps = 1000$,

$$latent\_dist\_reg\_weight = 0.001$$

I applied the degrading function on the generated image, that is, I blurred the generated image $G(z)$, before computing the loss.

I defined two functions to blur the image: *gaussian_kernel* and *blur_spatial*, the first one generates a gaussian kernel according to the size passed to it, the second blurs the image passed to it by convolve it with this kernel twice, horizontally and vertically, as we have learned in the class. I made some changes to the API; I passed the kernel to the *run_latent_optimization* function and use it to blur the generated image before computing the loss.

The kernel size affected the results; the smaller the kernel the less blurred the image, and we get a result like the one of undegraded image. Conversely, if we choose a big kernel, that is, more blurred image, the result will be less clear, many fine details will be lost, it will be somehow noisy.

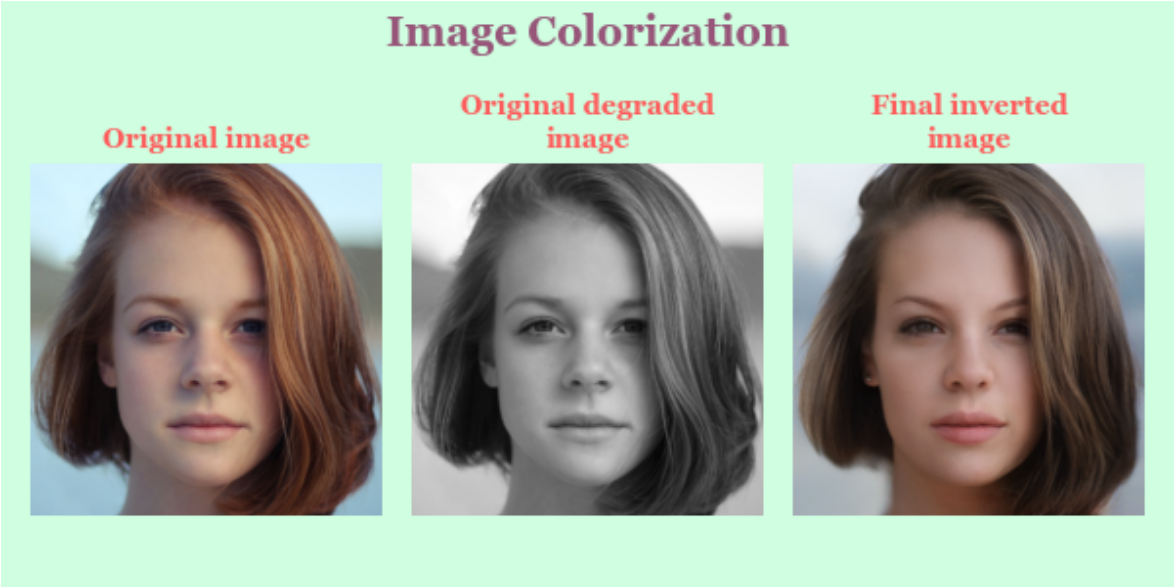


I noticed that after at most 1000 steps, the loss almost converged, and the result did not improve more. Also, and the bigger the *latent_dist_reg_weight* the closer the constructed image to the mean of natural images, and vice versa.

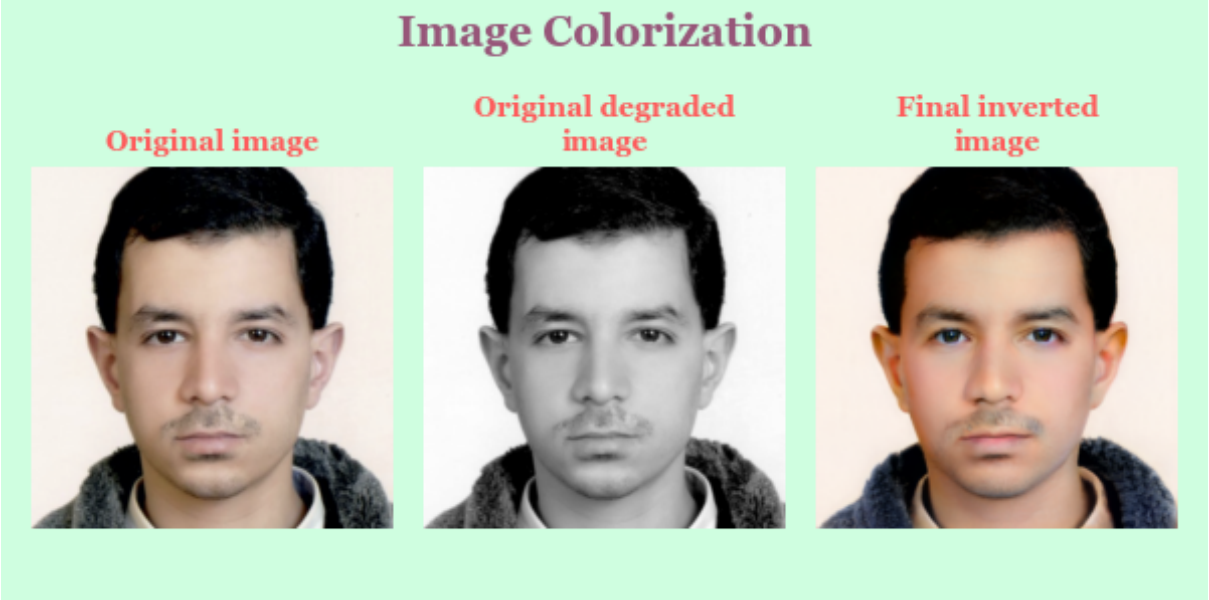
For example, a more blurred image (*kernel_size* = 195) with large *latent_dist_reg_weight* (= 0.2) will give a clear result but deviated relatively from the original one. In the other hand, small *latent_dist_reg_weight* (= 0.01) will give closer result to the original image, but less clear.

Image Colorization:

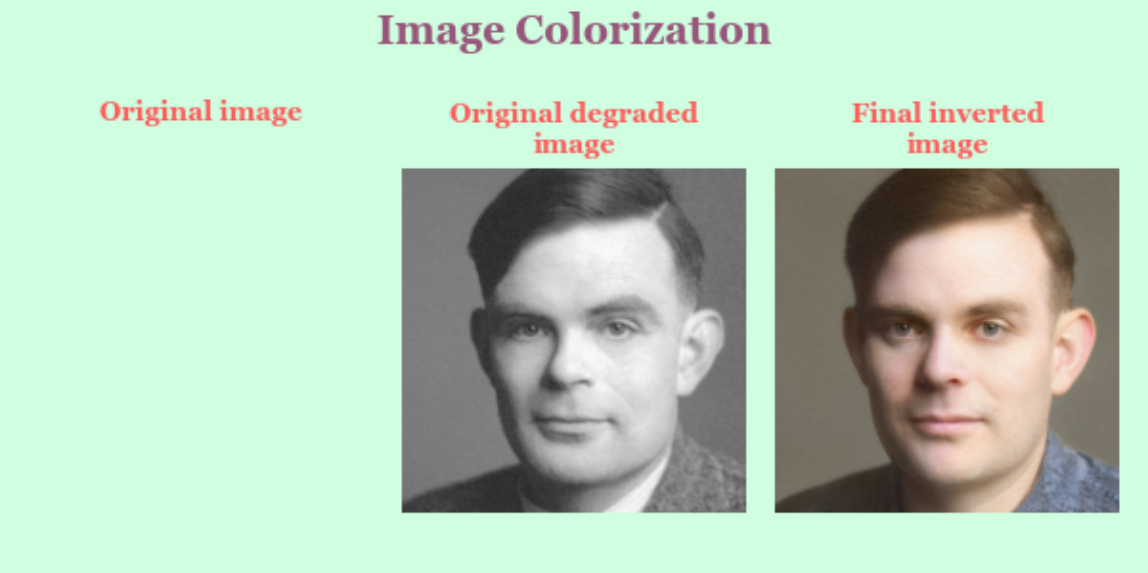
First image



Second image (my personal picture)



Alan Turing



Solution Discussion:

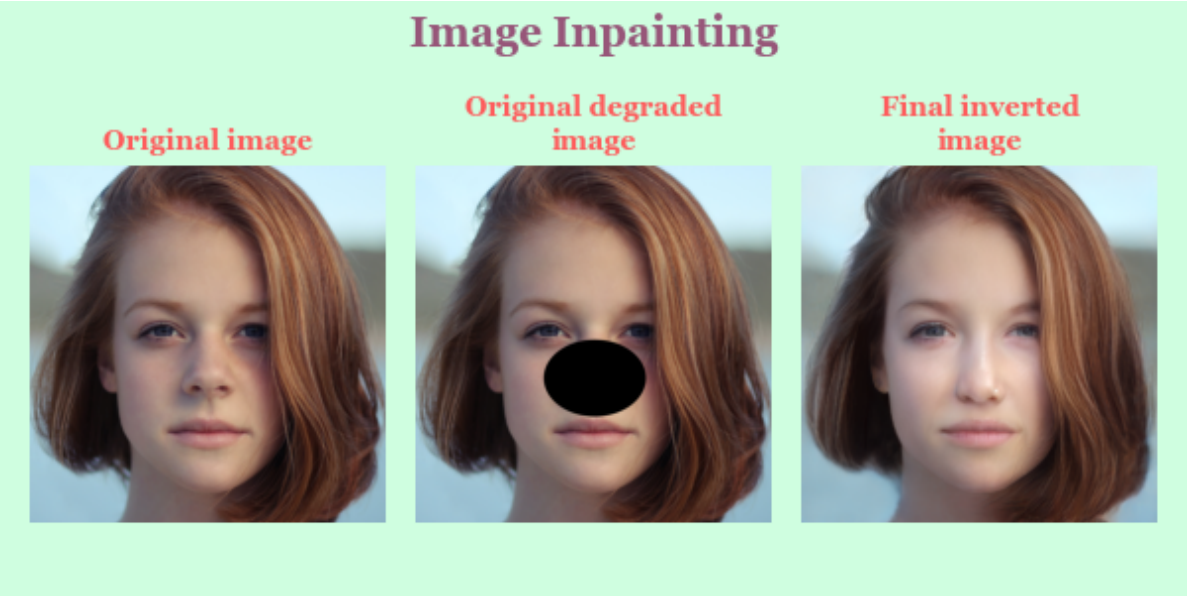
Hyperparameters: $num\_steps = 1000$, $latent\_dist\_reg\_weight = 1$

I applied the degrading function on the generated image, that is, I converted the generated image $G(z)$ to a grayscale one, before computing the loss. I used the `Grayscale` function from `torchvision.transforms` module to convert an image to grayscale mode.

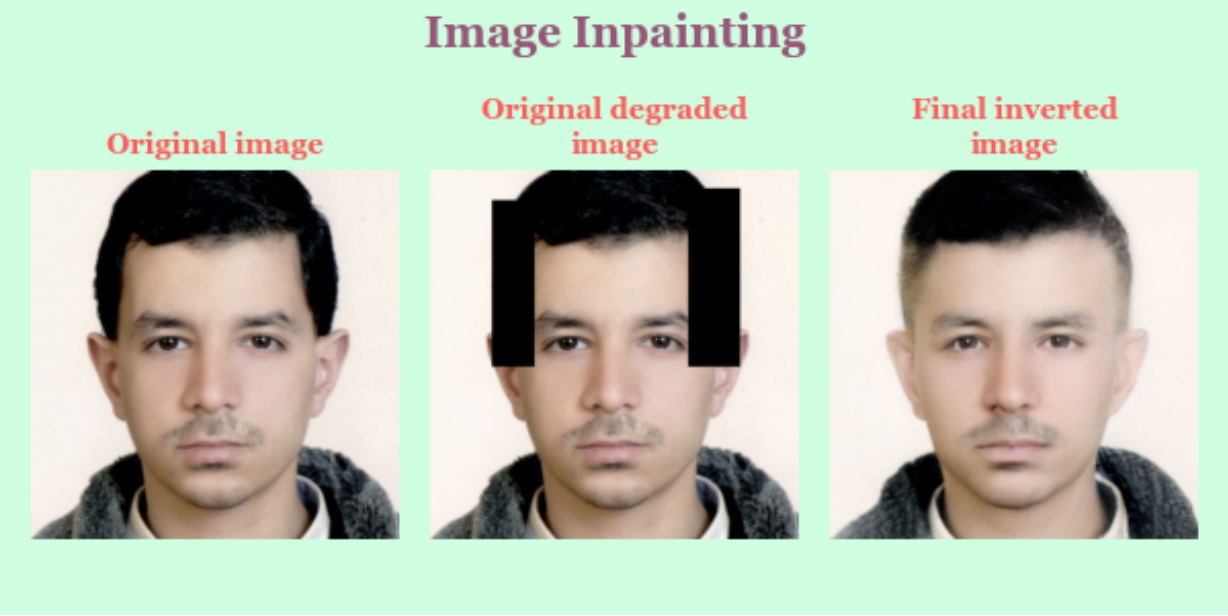
I noticed that a big $latent\_dist\_reg\_weight$ gives more realistic colorization, but with a little deviated image from the original one, I think the bigger the $latent\_dist\_reg\_weight$ the closer the constructed image to the mean of natural images, and vice versa. Moreover, after at most 1000 steps, the loss almost converged, and the result did not improve more.

Image Inpainting:

First image

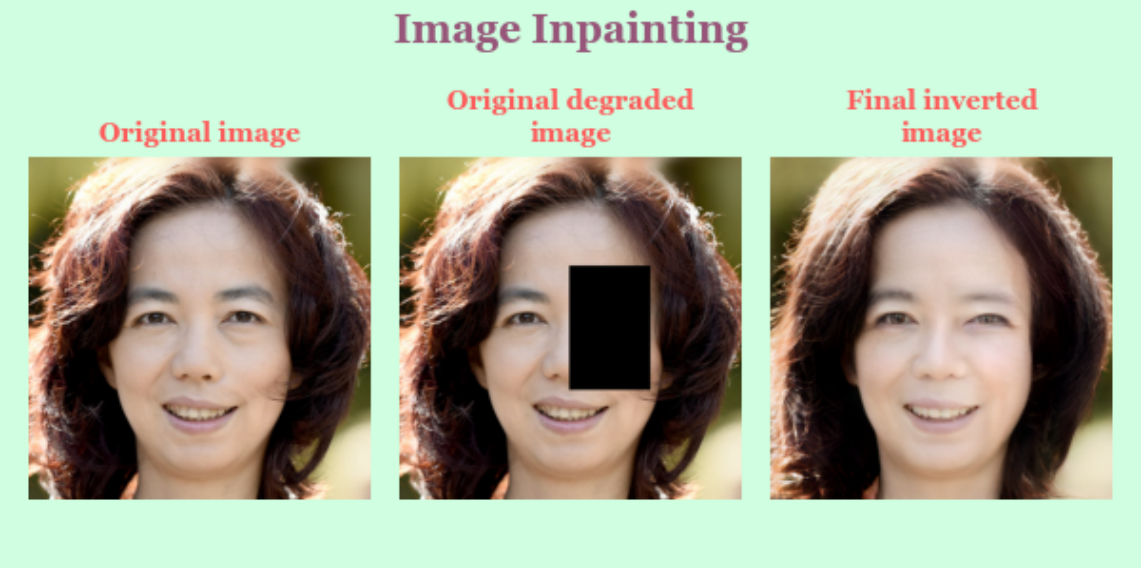


Second image (my personal picture)



* Here I think the difference in the sideburns is because the shape of the mask. *

Fei-Fei Li



Solution Discussion:

Hyperparameters: $num\_steps = 1000$, $latent\_dist\_reg\_weight = 0.2$

I applied the degrading function on the generated image, that is, I multiplied the generated image $G(z)$ with the mask, before computing the loss.

I made some changes to the API, I passed the mask path name ($mask\_fname$) to the $invert\_image$ function, then inside this function I read the mask file, converted it to a tensor, and multiply it with original image, then passed it to the $run\_latent\_optimization$ function and use it to multiply the generated image with it before computing the loss.

I noticed that after at most 1000 steps, the loss almost converged. The optimal $latent\_dist\_reg\_weight$ range for inpainting was between 0.1 – 0.3.

As the previous reconstruction problems, num_steps has improved the result up to 1000 steps, and the bigger the $latent\_dist\_reg\_weight$ the closer the constructed image to the mean of natural images, and vice versa.

3 ex5/IMPR Ex5 Deep Style Image Prior 2021 2022.ipynb


```
{
  "nbformat": 4,
  "nbformat_minor": 0,
  "metadata": {
    "colab": {
      "name": "Copy_of_IMPR_Ex5_Deep_Style_Image_Prior_2021_2022_.ipynb",
      "provenance": [],
      "collapsed_sections": []
    },
    "kernelspec": {
      "name": "python3",
      "display_name": "Python 3"
    },
    "language_info": {
      "name": "python"
    },
    "accelerator": "GPU"
  },
  "cells": [
    {
      "cell_type": "markdown",
      "source": [
        "# Important! \n",
        "When initially opening the notebook there should be a text to the right of the \"Help\" menu saying \"Changes will 
        "![WhatsApp Image 2021-01-02 at 22.02.20.jpeg](
        "\n",
        "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
        "\n",
        "**Failing to do so will result in code loss!**\n",
        "\n",
        " **Note** Make sure you are the only one that has access to it!\n"
      ],
      "metadata": {
        "id": "7PbmNXuh7WuC"
      }
    },
    {
      "cell_type": "code",
      "source": [
        "#@markdown #Image Processing - 67829. { display-mode: \"form\" }\n",
        "#@markdown ##Exercise 5:  Deep Style Image Prior\n",
        "#@markdown ##Due date: 13.01.2022 at 23:59\n",
        "#@title{ display-mode: \"form\" }\n",
        "\n",
        "#@markdown\n",
        "#@markdown This exercise is a bit different than the rest of the exercises in the course.\n",
        "#@markdown The submissions will be a PDF file with your answers and results to the exercise \n",
        "#@markdown as well as some files so that we can verify the authenticity of your results.\n",
        "#@markdown This notebook provides the basic code, but you do not need to adhere to some specific API \n",
        "#@markdown and we will not be running unit tests on your code. \n",
        "#@markdown We will however, be going over your code and running it manually. \n",
        "#@markdown Moreover, we will be running tests to ensure the authenticity of your solution and detect plagiarism\n",
        "#@markdown\n",
        "#@markdown\n",
        "#@markdown Before you start working on the exercise it is recommended that you review the lecture slides covering r
        "#@markdown\n",
```

```
        "#@markdown \n",
        "#@markdown **NOTE**: Neural networks are typically trained on GPUs, without GPUs training takes much longer. \n",
        "#@markdown To enable GPU tranining click on \"Runtime\" -> \"Change runtime type\" -> \"GPU\" -> \"SAVE\".\n",
        "#@markdown\n",
        "#@markdown **NOTE**: A short guide on debugging your code using colab is availble [here](https://colab.research.goo
        "\n",
        "#@markdown But first, we have to download all of the dependencies and install them.\n",
        "#@markdown Play this cell to download it and get everything ready. markdown This may take a few minutes.\n",
        "\n",
        " \n",
        "!mkdir impr_ex5_resources\n",
        "%cd impr_ex5_resources\n",
        "#!wget \"https://www.cs.huji.ac.il/~impr/shape_predictor_68_face_landmarks.dat\" -O shape_predictor_68_face_landmar
        "#!wget \"https://www.cs.huji.ac.il/~impr/align_faces.py\" -O align_faces.py \n",
        "!wget \"https://www.cs.huji.ac.il/~impr/stylegan2-ada-pytorch.tar\" -O stylegan2-ada-pytorch.tar\n",
        "!tar -xvf stylegan2-ada-pytorch.tar\n",
        "!rm -f stylegan2-ada-pytorch.tar\n",
        "\n",
        "import sys\n",
        "ROOT_PATH=\"/content/impr_ex5_resources/stylegan2-ada-pytorch\"\n",
        "sys.path.append(ROOT_PATH)\n",
        "\n",
        "\n",
        "!pip install ninja\n",
        "!pip install mediapy\n",
        "CHECKPOINTS_PATH = \"https://nvlabs-fi-cdn.nvidia.com/stylegan2-ada/pretrained/ffhq.pkl\"\n",
        "\n",
        "\n",
        "\n",
        "\n",
        "import copy\n",
        "import os\n",
        "from time import perf_counter\n",
        "import click\n",
        "import imageio\n",
        "import numpy as np\n",
        "import PIL.Image\n",
        "import torch\n",
        "import torch.nn.functional as F\n",
        "import torchvision.transforms as T\n",
        "\n",
        "import dnnlib\n",
        "import legacy\n",
        "import numpy as np\n",
        "from skimage.draw import line\n",
        "from torch.nn.functional import conv2d,conv1d\n",
        "from torchvision.transforms import Grayscale\n",
        "import mediapy as media\n",
        "from IPython.display import clear_output\n"
      ],
      "metadata": {
        "id": "Jean9HbQ7iq-",
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "outputId": "9b9a6d7d-db5a-425c-b06c-67db9f45c974"
      },
      "execution_count": 3,
      "outputs": [
        {
          "output_type": "stream",
          "name": "stdout",
          "text": [
            "/content/impr_ex5_resources\n",
            "--2022-01-17 11:01:04--  https://www.cs.huji.ac.il/~impr/stylegan2-ada-pytorch.tar\n",
            "Resolving www.cs.huji.ac.il (www.cs.huji.ac.il)... 132.65.118.16\n",
            "Connecting to www.cs.huji.ac.il (www.cs.huji.ac.il)|132.65.118.16|:443... connected.\n",
            "HTTP request sent, awaiting response... 302 Moved Temporarily\n",
            "Location: https://www.cs.huji.ac.il/w~impr/stylegan2-ada-pytorch.tar [following]\n",
```

```
            "--2022-01-17 11:01:05--  https://www.cs.huji.ac.il/w~impr/stylegan2-ada-pytorch.tar\n",
            "Reusing existing connection to www.cs.huji.ac.il:443.\n",
            "HTTP request sent, awaiting response... 200 OK\n",
            "Length: 100364800 (96M) [application/x-tar]\n",
            "Saving to: ‘stylegan2-ada-pytorch.tar’\n",
            "\n",
            "stylegan2-ada-pytor 100%[===================>]  95.71M  16.6MB/s    in 6.7s    \n",
            "\n",
            "2022-01-17 11:01:12 (14.4 MB/s) - ‘stylegan2-ada-pytorch.tar’ saved [100364800/100364800]\n",
            "\n",
            "stylegan2-ada-pytorch/\n",
            "stylegan2-ada-pytorch/metrics/\n",
            "stylegan2-ada-pytorch/._dataset_tool.py\n",
            "tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
            "stylegan2-ada-pytorch/dataset_tool.py\n",
            "stylegan2-ada-pytorch/._.DS_Store\n",
            "tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.FinderInfo'\n",
            "stylegan2-ada-pytorch/.DS_Store\n",
            "stylegan2-ada-pytorch/._style_mixing.py\n",
            "tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
            "stylegan2-ada-pytorch/style_mixing.py\n",
            "stylegan2-ada-pytorch/torch_utils/\n",
            "stylegan2-ada-pytorch/._legacy.py\n",
            "tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
            "stylegan2-ada-pytorch/legacy.py\n",
            "stylegan2-ada-pytorch/._generate.py\n",
            "tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
            "stylegan2-ada-pytorch/generate.py\n",
            "stylegan2-ada-pytorch/training/\n",
            "stylegan2-ada-pytorch/._README.md\n",
            "tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
            "stylegan2-ada-pytorch/README.md\n",
            "stylegan2-ada-pytorch/._projector.py\n",
            "tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
            "stylegan2-ada-pytorch/projector.py\n",
            "stylegan2-ada-pytorch/._train.py\n",
            "tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
            "stylegan2-ada-pytorch/train.py\n",
            "stylegan2-ada-pytorch/._calc_metrics.py\n",
            "tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
            "stylegan2-ada-pytorch/calc_metrics.py\n",
            "stylegan2-ada-pytorch/dnnlib/\n",
            "stylegan2-ada-pytorch/align_faces/\n",
            "stylegan2-ada-pytorch/._LICENSE.txt\n",
            "tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
            "stylegan2-ada-pytorch/LICENSE.txt\n",
            "stylegan2-ada-pytorch/align_faces/._shape_predictor_68_face_landmarks.dat\n",
            "tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
            "stylegan2-ada-pytorch/align_faces/shape_predictor_68_face_landmarks.dat\n",
            "stylegan2-ada-pytorch/align_faces/._align_faces.py\n",
            "tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
            "stylegan2-ada-pytorch/align_faces/align_faces.py\n",
            "stylegan2-ada-pytorch/dnnlib/._util.py\n",
            "tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
            "stylegan2-ada-pytorch/dnnlib/util.py\n",
            "stylegan2-ada-pytorch/dnnlib/.___init__.py\n",
            "tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
            "stylegan2-ada-pytorch/dnnlib/__init__.py\n",
            "stylegan2-ada-pytorch/dnnlib/__pycache__/\n",
            "stylegan2-ada-pytorch/dnnlib/__pycache__/._util.cpython-37.pyc\n",
            "tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
            "stylegan2-ada-pytorch/dnnlib/__pycache__/util.cpython-37.pyc\n",
            "stylegan2-ada-pytorch/dnnlib/__pycache__/.___init__.cpython-37.pyc\n",
            "tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
            "stylegan2-ada-pytorch/dnnlib/__pycache__/__init__.cpython-37.pyc\n",
            "stylegan2-ada-pytorch/training/.___init__.py\n",
            "tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
            "stylegan2-ada-pytorch/training/__init__.py\n",
```

```
"stylegan2-ada-pytorch/training/__pycache__/\n",
"stylegan2-ada-pytorch/training/._augment.py\n",
"tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
"stylegan2-ada-pytorch/training/augment.py\n",
"stylegan2-ada-pytorch/training/._training_loop.py\n",
"tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
"stylegan2-ada-pytorch/training/training_loop.py\n",
"stylegan2-ada-pytorch/training/._dataset.py\n",
"tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
"stylegan2-ada-pytorch/training/dataset.py\n",
"stylegan2-ada-pytorch/training/._networks.py\n",
"tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
"stylegan2-ada-pytorch/training/networks.py\n",
"stylegan2-ada-pytorch/training/._loss.py\n",
"tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
"stylegan2-ada-pytorch/training/loss.py\n",
"stylegan2-ada-pytorch/training/__pycache__/._networks.cpython-37.pyc\n",
"tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
"stylegan2-ada-pytorch/training/__pycache__/networks.cpython-37.pyc\n",
"stylegan2-ada-pytorch/training/__pycache__/.___init__.cpython-37.pyc\n",
"tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
"stylegan2-ada-pytorch/training/__pycache__/__init__.cpython-37.pyc\n",
"stylegan2-ada-pytorch/torch_utils/._misc.py\n",
"tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
"stylegan2-ada-pytorch/torch_utils/misc.py\n",
"stylegan2-ada-pytorch/torch_utils/._persistence.py\n",
"tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
"stylegan2-ada-pytorch/torch_utils/persistence.py\n",
"stylegan2-ada-pytorch/torch_utils/._custom_ops.py\n",
"tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
"stylegan2-ada-pytorch/torch_utils/custom_ops.py\n",
"stylegan2-ada-pytorch/torch_utils/.___init__.py\n",
"tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
"stylegan2-ada-pytorch/torch_utils/__init__.py\n",
"stylegan2-ada-pytorch/torch_utils/__pycache__/\n",
"stylegan2-ada-pytorch/torch_utils/._training_stats.py\n",
"tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
"stylegan2-ada-pytorch/torch_utils/training_stats.py\n",
"stylegan2-ada-pytorch/torch_utils/ops/\n",
"stylegan2-ada-pytorch/torch_utils/ops/._bias_act.cpp\n",
"tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
"stylegan2-ada-pytorch/torch_utils/ops/bias_act.cpp\n",
"stylegan2-ada-pytorch/torch_utils/ops/._upfirdn2d.cpp\n",
"tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
"stylegan2-ada-pytorch/torch_utils/ops/upfirdn2d.cpp\n",
"stylegan2-ada-pytorch/torch_utils/ops/._grid_sample_gradfix.py\n",
"tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
"stylegan2-ada-pytorch/torch_utils/ops/grid_sample_gradfix.py\n",
"stylegan2-ada-pytorch/torch_utils/ops/._bias_act.py\n",
"tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
"stylegan2-ada-pytorch/torch_utils/ops/bias_act.py\n",
"stylegan2-ada-pytorch/torch_utils/ops/._upfirdn2d.cu\n",
"tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
"stylegan2-ada-pytorch/torch_utils/ops/upfirdn2d.cu\n",
"stylegan2-ada-pytorch/torch_utils/ops/._bias_act.cu\n",
"tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
"stylegan2-ada-pytorch/torch_utils/ops/bias_act.cu\n",
"stylegan2-ada-pytorch/torch_utils/ops/.___init__.py\n",
"tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
"stylegan2-ada-pytorch/torch_utils/ops/__init__.py\n",
"stylegan2-ada-pytorch/torch_utils/ops/._upfirdn2d.py\n",
"tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
"stylegan2-ada-pytorch/torch_utils/ops/upfirdn2d.py\n",
"stylegan2-ada-pytorch/torch_utils/ops/__pycache__/\n",
"stylegan2-ada-pytorch/torch_utils/ops/._bias_act.h\n",
"tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
"stylegan2-ada-pytorch/torch_utils/ops/bias_act.h\n",
"stylegan2-ada-pytorch/torch_utils/ops/._fma.py\n",
```

```
            "tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
            "stylegan2-ada-pytorch/torch_utils/ops/fma.py\n",
            "stylegan2-ada-pytorch/torch_utils/ops/._conv2d_resample.py\n",
            "tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
            "stylegan2-ada-pytorch/torch_utils/ops/conv2d_resample.py\n",
            "stylegan2-ada-pytorch/torch_utils/ops/._upfirdn2d.h\n",
            "tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
            "stylegan2-ada-pytorch/torch_utils/ops/upfirdn2d.h\n",
            "stylegan2-ada-pytorch/torch_utils/ops/._conv2d_gradfix.py\n",
            "tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
            "stylegan2-ada-pytorch/torch_utils/ops/conv2d_gradfix.py\n",
            "stylegan2-ada-pytorch/torch_utils/ops/__pycache__/._fma.cpython-37.pyc\n",
            "tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
            "stylegan2-ada-pytorch/torch_utils/ops/__pycache__/fma.cpython-37.pyc\n",
            "stylegan2-ada-pytorch/torch_utils/ops/__pycache__/._conv2d_gradfix.cpython-37.pyc\n",
            "tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
            "stylegan2-ada-pytorch/torch_utils/ops/__pycache__/conv2d_gradfix.cpython-37.pyc\n",
            "stylegan2-ada-pytorch/torch_utils/ops/__pycache__/._bias_act.cpython-37.pyc\n",
            "tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
            "stylegan2-ada-pytorch/torch_utils/ops/__pycache__/bias_act.cpython-37.pyc\n",
            "stylegan2-ada-pytorch/torch_utils/ops/__pycache__/._upfirdn2d.cpython-37.pyc\n",
            "tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
            "stylegan2-ada-pytorch/torch_utils/ops/__pycache__/upfirdn2d.cpython-37.pyc\n",
            "stylegan2-ada-pytorch/torch_utils/ops/__pycache__/._conv2d_resample.cpython-37.pyc\n",
            "tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
            "stylegan2-ada-pytorch/torch_utils/ops/__pycache__/conv2d_resample.cpython-37.pyc\n",
            "stylegan2-ada-pytorch/torch_utils/ops/__pycache__/.___init__.cpython-37.pyc\n",
            "tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
            "stylegan2-ada-pytorch/torch_utils/ops/__pycache__/__init__.cpython-37.pyc\n",
            "stylegan2-ada-pytorch/torch_utils/__pycache__/._persistence.cpython-37.pyc\n",
            "tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
            "stylegan2-ada-pytorch/torch_utils/__pycache__/persistence.cpython-37.pyc\n",
            "stylegan2-ada-pytorch/torch_utils/__pycache__/._custom_ops.cpython-37.pyc\n",
            "tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
            "stylegan2-ada-pytorch/torch_utils/__pycache__/custom_ops.cpython-37.pyc\n",
            "stylegan2-ada-pytorch/torch_utils/__pycache__/.___init__.cpython-37.pyc\n",
            "tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
            "stylegan2-ada-pytorch/torch_utils/__pycache__/__init__.cpython-37.pyc\n",
            "stylegan2-ada-pytorch/torch_utils/__pycache__/._misc.cpython-37.pyc\n",
            "tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
            "stylegan2-ada-pytorch/torch_utils/__pycache__/misc.cpython-37.pyc\n",
            "stylegan2-ada-pytorch/metrics/._precision_recall.py\n",
            "tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
            "stylegan2-ada-pytorch/metrics/precision_recall.py\n",
            "stylegan2-ada-pytorch/metrics/._frechet_inception_distance.py\n",
            "tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
            "stylegan2-ada-pytorch/metrics/frechet_inception_distance.py\n",
            "stylegan2-ada-pytorch/metrics/.___init__.py\n",
            "tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
            "stylegan2-ada-pytorch/metrics/__init__.py\n",
            "stylegan2-ada-pytorch/metrics/._metric_utils.py\n",
            "tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
            "stylegan2-ada-pytorch/metrics/metric_utils.py\n",
            "stylegan2-ada-pytorch/metrics/._metric_main.py\n",
            "tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
            "stylegan2-ada-pytorch/metrics/metric_main.py\n",
            "stylegan2-ada-pytorch/metrics/._perceptual_path_length.py\n",
            "tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
            "stylegan2-ada-pytorch/metrics/perceptual_path_length.py\n",
            "stylegan2-ada-pytorch/metrics/._inception_score.py\n",
            "tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
            "stylegan2-ada-pytorch/metrics/inception_score.py\n",
            "stylegan2-ada-pytorch/metrics/._kernel_inception_distance.py\n",
            "tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
            "stylegan2-ada-pytorch/metrics/kernel_inception_distance.py\n",
            "Collecting ninja\n",
            "  Downloading ninja-1.10.2.3-py2.py3-none-manylinux_2_5_x86_64.manylinux1_x86_64.whl (108 kB)\n",
            "\u001b[K     |                | 108 kB 5.4 MB/s \n",
```

```
            "\u001b[?25hInstalling collected packages: ninja\n",
            "Successfully installed ninja-1.10.2.3\n",
            "Collecting mediapy\n",
            "  Downloading mediapy-1.0.3-py3-none-any.whl (24 kB)\n",
            "Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages (from mediapy) (1.19.5)\n",
            "Requirement already satisfied: Pillow in /usr/local/lib/python3.7/dist-packages (from mediapy) (7.1.2)\n",
            "Requirement already satisfied: matplotlib in /usr/local/lib/python3.7/dist-packages (from mediapy) (3.2.2)\n",
            "Requirement already satisfied: ipython in /usr/local/lib/python3.7/dist-packages (from mediapy) (5.5.0)\n",
            "Requirement already satisfied: setuptools>=18.5 in /usr/local/lib/python3.7/dist-packages (from ipython->mediap
            "Requirement already satisfied: pygments in /usr/local/lib/python3.7/dist-packages (from ipython->mediapy) (2.6.
            "Requirement already satisfied: pickleshare in /usr/local/lib/python3.7/dist-packages (from ipython->mediapy) (0
            "Requirement already satisfied: decorator in /usr/local/lib/python3.7/dist-packages (from ipython->mediapy) (4.4
            "Requirement already satisfied: prompt-toolkit<2.0.0,>=1.0.4 in /usr/local/lib/python3.7/dist-packages (from ipy
            "Requirement already satisfied: simplegeneric>0.8 in /usr/local/lib/python3.7/dist-packages (from ipython->media
            "Requirement already satisfied: traitlets>=4.2 in /usr/local/lib/python3.7/dist-packages (from ipython->mediapy)
            "Requirement already satisfied: pexpect in /usr/local/lib/python3.7/dist-packages (from ipython->mediapy) (4.8.0
            "Requirement already satisfied: wcwidth in /usr/local/lib/python3.7/dist-packages (from prompt-toolkit<2.0.0,>=1
            "Requirement already satisfied: six>=1.9.0 in /usr/local/lib/python3.7/dist-packages (from prompt-toolkit<2.0.0,
            "Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local/lib/python3.7/dist-packag
            "Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.7/dist-packages (from matplotlib->mediapy
            "Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib-
            "Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib->me
            "Requirement already satisfied: ptyprocess>=0.5 in /usr/local/lib/python3.7/dist-packages (from pexpect->ipython
            "Installing collected packages: mediapy\n",
            "Successfully installed mediapy-1.0.3\n"
          ]
        }
      ]
    },
    {
      "cell_type": "markdown",
      "source": [
        "# Mounting Google Drive"
      ],
      "metadata": {
        "id": "LY3SOM7bKggZ"
      }
    },
    {
      "cell_type": "code",
      "source": [
        "#@markdown **NOTE**: It is strongly advised you save your results to Google \n",
        "#@markdown Drive as they will be deleted from Colab once it restarts. \n",
        "#@markdown To connect Google Drive run this cell. \n",
        "from google.colab import drive\n",
        "drive.mount('/content/gdrive/')"
      ],
      "metadata": {
        "id": "E8dlOxYdKhyi",
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "outputId": "64ac6ddb-bcd9-4b15-99e9-a6caca52e8bf"
      },
      "execution_count": 4,
      "outputs": [
        {
          "output_type": "stream",
          "name": "stdout",
          "text": [
            "Mounted at /content/gdrive/\n"
          ]
        }
      ]
    },
    {
      "cell_type": "markdown",
      "source": [
```

```
          "Below is the root dir of your Google Drive. To choose the destenation of the dir to save and read from, create it i
        ],
        "metadata": {
          "id": "Lp4YuqH2Lmku"
        }
      },
      {
        "cell_type": "code",
        "source": [
          "ROOT_GDRIVE_PATH=\"/content/gdrive/MyDrive/\"\n",
          "GDRIVE_SAVE_REL_PATH = \"IMPRO_EX5_2021\"\n",
          "FULL_GDRIVE_SAVE_PATH = ROOT_GDRIVE_PATH + GDRIVE_SAVE_REL_PATH"
        ],
        "metadata": {
          "id": "LAjRCCKLLBI5"
        },
        "execution_count": 5,
        "outputs": []
      },
      {
        "cell_type": "markdown",
        "source": [
          "# General Variables"
        ],
        "metadata": {
          "id": "mNEXRTie-E93"
        }
      },
      {
        "cell_type": "code",
        "source": [
          "GAUSSIAN_BLUR_DEGRADATION= 'GAUSSIAN_BLUR_DEGRADATION'\n",
          "GRAYSCALE_DEGRADATION = 'GRAYSCALE_DEGRADATION'\n",
          "INPAINTING_DEGRADATION = 'INPAINTING_DEGRADATION'\n",
          "DENOISING_DEGRADATION = 'DENOISING_DEGRADATION'\n",
          "NO_DEGRADATION= 'NO_DEGRADATION'\n"
        ],
        "metadata": {
          "id": "RFt1dKcO95Qi"
        },
        "execution_count": 6,
        "outputs": []
      },
      {
        "cell_type": "markdown",
        "source": [
          "# Image Alignment"
        ],
        "metadata": {
          "id": "NlMqcLzq9lHv"
        }
      },
      {
        "cell_type": "code",
        "metadata": {
          "id": "cO2O4E5oXn-5",
          "colab": {
            "base_uri": "https://localhost:8080/"
          },
          "outputId": "0716c153-8f73-4662-8887-b578e63c26f6"
        },
        "source": [
          "# The align_faces.py script takes in an input image path, an output image path, and a dat file path. The dat file i
          "# It is advised that you save the files to google drive as restarting Colab will erase them.\n",
          "!python \"$ROOT_PATH/align_faces/align_faces.py\" '/content/gdrive/MyDrive/IMPRO_EX5_2021/input_images/my_face.png'
        ],
        "execution_count": null,
        "outputs": [
```

```
        {
          "output_type": "stream",
          "name": "stdout",
          "text": [
            "Number of faces detected: 1\n",
            "Detection 0: Left: 617 Top: 937 Right: 1575 Bottom: 1895\n",
            "Part 0: (614, 1205), Part 1: (624, 1335) ...\n"
          ]
        }
      ]
    },
    {
      "cell_type": "markdown",
      "source": [
        "# Degradation Functions "
      ],
      "metadata": {
        "id": "Z7GO7LxPC19b"
      }
    },
    {
      "cell_type": "code",
      "source": [
        "    # ********************************************************************************************************\n",
        "    # ******************                  NEED TO ADD DEGRADATION FUNCTIONS               ******************\n",
        "    # ********************************************************************************************************\n",
        "\n",
        "    # I. GAUSSIAN_BLUR_DEGRADATION\n",
        "\n",
        "    def gaussian_kernel(filter_size):\n",
        "      basis = filter_vec = torch.tensor([0.5, 0.5], dtype=torch.float32).reshape(1, 1, 2)\n",
        "      for i in range(filter_size - 2):\n",
        "          filter_vec = conv1d(filter_vec, basis, padding=1)\n",
        "      return filter_vec\n",
        "      # return filter_vec / torch.sum(filter_vec)\n",
        "\n",
        "\n",
        "    def blur_spatial(im, filter_vec):\n",
        "        filter_vec = filter_vec.repeat(3, 1, 1, 1)\n",
        "        padding = filter_vec.shape[3]//2\n",
        "        blurred_im = conv2d(im, filter_vec , padding=[0,padding], groups=3)\n",
        "        blurred_im = conv2d(blurred_im, filter_vec.permute(0, 1, 3, 2), padding=[padding,0], groups=3)\n",
        "\n",
        "        return blurred_im\n",
        "\n",
        "    # II. GRAYSCALE_DEGRADATION\n",
        "\n",
        "    # III. INPAINTING_DEGRADATION"
      ],
      "metadata": {
        "id": "PY7gxd63zqnK"
      },
      "execution_count": 7,
      "outputs": []
    },
    {
      "cell_type": "markdown",
      "source": [
        "# GAN Inversion"
      ],
      "metadata": {
        "id": "j2EbeYsb9u7A"
      }
    },
    {
      "cell_type": "code",
      "source": [
        "def run_latent_optimization(outdir,\n",
```

```
"    degradation_mode,\n",
"    G,\n",
"    imgs_to_disply_dict,\n",
"    target: torch.Tensor, # [C,H,W] and dynamic range [0,255], W & H must match G output resolution\n",
"    *,\n",
"    num_steps                 = 1000,\n",
"    w_avg_samples             = 10000,\n",
"    initial_learning_rate     = 0.1,\n",
"    initial_noise_factor      = 0.05,\n",
"    lr_rampdown_length        = 0.25,\n",
"    lr_rampup_length          = 0.05,\n",
"    noise_ramp_length         = 0.75,\n",
"    regularize_noise_weight   = 1e5,\n",
"    latent_dist_reg_weight    = 0.001,\n",
"    blur_kernel               = None,\n",
"    mask                      = None,\n",
"    device: torch.device\n",
"    \n",
"):\n",
"    assert target.shape == (G.img_channels, G.img_resolution, G.img_resolution)\n",
"\n",
"    if degradation_mode == GAUSSIAN_BLUR_DEGRADATION:\n",
"      assert blur_kernel is not None\n",
"    if degradation_mode == INPAINTING_DEGRADATION:\n",
"      assert mask is not None\n",
"\n",
"    G = copy.deepcopy(G).eval().requires_grad_(False).to(device) # type: ignore\n",
"  \n",
"    # Compute w stats.\n",
"    print(f'Computing W midpoint and stddev using {w_avg_samples} samples...')\n",
"    z_samples = np.random.RandomState(123).randn(w_avg_samples, G.z_dim)\n",
"    w_samples = G.mapping(torch.from_numpy(z_samples).to(device), None)  # [N, L, C]\n",
"    w_samples = w_samples.cpu().numpy().astype(np.float32)\n",
"    w_avg = np.mean(w_samples, axis=0, keepdims=True)      # [1, 18, C]\n",
"    w_avg_original = torch.from_numpy(w_avg).to(device).float()\n",
"    w_std = (np.sum((w_samples - w_avg) ** 2) / w_avg_samples) ** 0.5\n",
"\n",
"    # Setup noise inputs.\n",
"    noise_bufs = { name: buf for (name, buf) in G.synthesis.named_buffers() if 'noise_const' in name }\n",
"\n",
"    # Load VGG16 feature detector.\n",
"    url = 'https://nvlabs-fi-cdn.nvidia.com/stylegan2-ada-pytorch/pretrained/metrics/vgg16.pt'\n",
"    with dnnlib.util.open_url(url) as f:\n",
"        vgg16 = torch.jit.load(f).eval().to(device)\n",
"    \n",
"    # Features for target image.\n",
"    target_images = target.unsqueeze(0).to(device).to(torch.float32)\n",
"\n",
"    if target_images.shape[2] > 256:\n",
"        target_images = F.interpolate(target_images, size=(256, 256), mode='area')\n",
"    target_features = vgg16(target_images, resize_images=False, return_lpips=True)\n",
"\n",
"    w_opt = torch.tensor(w_avg, dtype=torch.float32, device=device, requires_grad=True) \n",
"    w_out = torch.zeros([num_steps] + list(w_opt.shape[1:]), dtype=torch.float32, device=device)\n",
"    loss_out = torch.zeros([num_steps], dtype=torch.float32, device=device)\n",
"    optimizer = torch.optim.Adam([w_opt] + list(noise_bufs.values()), betas=(0.9, 0.999), lr=initial_learning_rate)\n",
"    \n",
"    # Init noise.\n",
"    for buf in noise_bufs.values():\n",
"        buf[:] = torch.randn_like(buf)\n",
"        buf.requires_grad = True\n",
"\n",
"    for step in range(num_steps):\n",
"        # Learning rate schedule.\n",
"        t = step / num_steps\n",
"        w_noise_scale = w_std * initial_noise_factor * max(0.0, 1.0 - t / noise_ramp_length) ** 2\n",
"        lr_ramp = min(1.0, (1.0 - t) / lr_rampdown_length)\n",
"        lr_ramp = 0.5 - 0.5 * np.cos(lr_ramp * np.pi)\n",
```

```
"        lr_ramp = lr_ramp * min(1.0, t / lr_rampup_length)\n",
"        lr = initial_learning_rate * lr_ramp\n",
"        for param_group in optimizer.param_groups:\n",
"            param_group['lr'] = lr\n",
"\n",
"        # Synth image from opt_w\n",
"        w_noise = torch.randn_like(w_opt) * w_noise_scale\n",
"        ws = w_opt + w_noise\n",
"        synth_images = G.synthesis(ws, noise_mode='const')\n",
"        \n",
"        # Prep to save synth image\n",
"        synth_image_save = (synth_images + 1) * (255/2)\n",
"        synth_image_save = synth_image_save.permute(0, 2, 3, 1).clamp(0, 255).to(torch.uint8)[0].cpu().numpy()\n",
"    \n",
"    \n",
"        # ************************************************************************************************************\
"        # ******************                    NEED TO FILL IN THE FOLLOWING CODE              ******************\
"        # ************************************************************************************************************\
"        if  degradation_mode == INPAINTING_DEGRADATION:\n",
"          assert mask.shape == synth_images.shape\n",
"          synth_images *= mask\n",
"        elif degradation_mode == GRAYSCALE_DEGRADATION:\n",
"          synth_images = Grayscale(num_output_channels=3)(synth_images)\n",
"        elif degradation_mode == GAUSSIAN_BLUR_DEGRADATION:\n",
"          synth_images = blur_spatial(synth_images, blur_kernel)\n",
"\n",
"        # ************************************************************************************************************\
"        # ******************                                END CODE TO ADD SECTION               ******************\
"        # ************************************************************************************************************\
"         \n",
"        \n",
"        # Prep to save and show images\n",
"        synth_image_degraded_save = (synth_images + 1) * (255/2)\n",
"        synth_image_degraded_save = synth_image_degraded_save.permute(0, 2, 3, 1).clamp(0, 255).to(torch.uint8)[0].
"        \n",
"        if step % 20 == 0:\n",
"          imgs_to_disply_dict[\"Generated Image\"]=synth_image_save\n",
"          imgs_to_disply_dict[\"Generated Degraded Image\"]=synth_image_degraded_save\n",
"          clear_output(wait=True)\n",
"          media.show_images(imgs_to_disply_dict,height=256)\n",
"        if step % 100 == 0:\n",
"          PIL.Image.fromarray(synth_image_save, 'RGB').save(f'{outdir}/intermidiate_%d_not_degraded.png'%step)\n",
"          PIL.Image.fromarray(synth_image_degraded_save, 'RGB').save(f'{outdir}/intermidiate_%d_degraded.png'%step)
"\n",
"\n",
"        # Noise regularization.\n",
"        reg_loss = 0.0\n",
"        for v in noise_bufs.values():\n",
"            noise = v[None,None,:,:] # must be [1,1,H,W] for F.avg_pool2d()\n",
"            while True:\n",
"                reg_loss += (noise*torch.roll(noise, shifts=1, dims=3)).mean()**2\n",
"                reg_loss += (noise*torch.roll(noise, shifts=1, dims=2)).mean()**2\n",
"                if noise.shape[2] <= 8:\n",
"                    break\n",
"                noise = F.avg_pool2d(noise, kernel_size=2)\n",
"\n",
"        # Downsample image to 256x256 if it's larger than that. VGG was built for 224x224 images.\n",
"        synth_images = (synth_images + 1) * (255/2)\n",
"        if synth_images.shape[2] > 256:\n",
"            synth_images = F.interpolate(synth_images, size=(256, 256), mode='area')\n",
"\n",
"        # Features for synth images.\n",
"        synth_features = vgg16(synth_images, resize_images=False, return_lpips=True)\n",
"        \n",
"        # Compute loss\n",
"        percep_loss = (target_features - synth_features).square().sum()\n",
"        latent_dist_reg = F.l1_loss(w_avg_original, w_opt)\n",
"        loss = percep_loss + reg_loss * regularize_noise_weight  + latent_dist_reg_weight * latent_dist_reg\n",
```

```
        "\n",
        "\n",
        "        # Step\n",
        "        optimizer.zero_grad(set_to_none=True)\n",
        "        loss.backward()\n",
        "        optimizer.step()\n",
        "\n",
        "        print(f'step {step+1:>4d}/{num_steps}: percep_loss {percep_loss:<4.2f} latent_dist_reg {latent_dist_reg:<4.
        "\n",
        "        # Save inverted latent for each optimization step.\n",
        "        w_out[step] = w_opt.detach()[0]\n",
        "        loss_out[step] = loss\n",
        "\n",
        "        # Normalize noise.\n",
        "        with torch.no_grad():\n",
        "            for buf in noise_bufs.values():\n",
        "                buf -= buf.mean()\n",
        "                buf *= buf.square().mean().rsqrt()\n",
        "    \n",
        "    return w_out, loss_out"
      ],
      "metadata": {
        "id": "daYIThcfi9PP"
      },
      "execution_count": 14,
      "outputs": []
    },
    {
      "cell_type": "code",
      "metadata": {
        "id": "mdFYp5sBor2U"
      },
      "source": [
        "def invert_image(degradation_mode,\n",
        "                 target_fname,\n",
        "                 outdir,\n",
        "                 mask_fname = None,\n",
        "                 seed=303,\n",
        "                 num_steps=1000,\n",
        "                 latent_dist_reg_weight=0.001):\n",
        "    np.random.seed(seed)\n",
        "    torch.manual_seed(seed)\n",
        "\n",
        "    # Load networks.\n",
        "    print('Loading networks from \"%s\"...' % CHECKPOINTS_PATH)\n",
        "    device = torch.device('cuda')\n",
        "    with dnnlib.util.open_url(CHECKPOINTS_PATH) as fp:\n",
        "        networks = legacy.load_network_pkl(fp)\n",
        "        G = networks['G_ema'].requires_grad_(False).to(device)\n",
        "        \n",
        "\n",
        "    # Load target image.\n",
        "    if not os.path.exists(outdir):\n",
        "      os.makedirs(outdir)\n",
        "    target_pil = PIL.Image.open(target_fname).convert('RGB')\n",
        "    w, h = target_pil.size\n",
        "    s = min(w, h)\n",
        "    target_pil = target_pil.crop(((w - s) // 2, (h - s) // 2, (w + s) // 2, (h + s) // 2))\n",
        "    target_pil = target_pil.resize((G.img_resolution, G.img_resolution), PIL.Image.LANCZOS)\n",
        "    target_uint8 = np.array(target_pil, dtype=np.uint8)\n",
        "    target = torch.tensor(target_uint8.transpose([2, 0, 1]), device=device), \n",
        "    target_images = target[0].unsqueeze(0).to(device).to(torch.float32)\n",
        "\n",
        "    # ************************************************************************************************\n",
        "    # ******************                    NEED TO FILL IN THE FOLLOWING CODE              ******************\n",
        "    # ************************************************************************************************\n",
        "    kernel = None\n",
        "    mask = None\n",
```

```
        "    if  degradation_mode == INPAINTING_DEGRADATION:\n",
        "      mask = PIL.Image.open(mask_fname).convert('RGB')\n",
        "      mask = torch.tensor(T.ToTensor()(mask), dtype=torch.float32, device=device).unsqueeze(0)\n",
        "      target_images *= mask\n",
        "    elif degradation_mode == GRAYSCALE_DEGRADATION:\n",
        "      target_images = Grayscale(num_output_channels=3)(target_images)\n",
        "    elif degradation_mode == GAUSSIAN_BLUR_DEGRADATION:\n",
        "      kernel = gaussian_kernel(85).to(device)\n",
        "      target_images = blur_spatial(target_images, kernel)\n",
        "      pass\n",
        "      \n",
        "    # ********************************************************************************************************\n",
        "    # ******************                              END CODE TO ADD SECTION                  ******************\n",
        "    # ********************************************************************************************************\n",
        "\n",
        "    #Save target image\n",
        "    target_to_save = target_images.permute(0, 2, 3, 1).clamp(0, 255).to(torch.uint8)[0].cpu().numpy()\n",
        "    PIL.Image.fromarray(target_to_save, 'RGB').save(f'{outdir}/original_degraded_image.png')\n",
        "    imgs_to_disply_dict = {\n",
        "        \"Original Image\":target_uint8,\n",
        "        \"Original Degraded Image\":target_to_save,\n",
        "              }\n",
        "\n",
        "    # Run latent optimization\n",
        "    print('Run latent optimization')\n",
        "    start_time = perf_counter()\n",
        "    optimization_steps, optimization_losses = run_latent_optimization(\n",
        "        outdir,\n",
        "        degradation_mode,\n",
        "        G,\n",
        "        imgs_to_disply_dict,\n",
        "        target_images[0],\n",
        "        num_steps=num_steps,\n",
        "        device=device,\n",
        "        latent_dist_reg_weight=latent_dist_reg_weight,\n",
        "        blur_kernel=kernel,\n",
        "        mask=mask\n",
        "    )\n",
        "\n",
        "    print (f'Elapsed: {(perf_counter()-start_time):.1f} s')\n",
        "    os.makedirs(outdir, exist_ok=True)\n",
        "\n",
        "    # Save final inverted image and latent vector.\n",
        "    inverted_latent = optimization_steps[-1]\n",
        "    synth_image = G.synthesis(inverted_latent.unsqueeze(0), noise_mode='const')\n",
        "    synth_image = (synth_image + 1) * (255/2)\n",
        "    synth_image = synth_image.permute(0, 2, 3, 1).clamp(0, 255).to(torch.uint8)[0].cpu().numpy()\n",
        "    PIL.Image.fromarray(synth_image, 'RGB').save(f'{outdir}/final_inverted_image.png')\n",
        "    np.savez(f'{outdir}/inverted_latent.npz', latent=inverted_latent.unsqueeze(0).cpu().numpy())\n",
        "\n",
        "    return optimization_losses"
      ],
      "execution_count": 20,
      "outputs": []
    },
    {
      "cell_type": "markdown",
      "source": [
        "# GAN Inversion with no degradation"
      ],
      "metadata": {
        "id": "UL3mDbZOGJ2c"
      }
    },
    {
      "cell_type": "markdown",
      "source": [
        "**The optimization progressions with over 5 images**"
```

```
    ],
    "metadata": {
      "id": "wcXL2UYrGp7B"
    }
  },
  {
    "cell_type": "code",
    "source": [
      "opt_losses = invert_image(degradation_mode=NO_DEGRADATION,\n",
      "                          target_fname='/content/gdrive/MyDrive/image_const/Christopher_Campbell_aligned.png',\n",
      "                          outdir='/content/gdrive/MyDrive/image_const',\n",
      "                          seed=303,\n",
      "                          num_steps=800,\n",
      "                          latent_dist_reg_weight=0.1)"
    ],
    "metadata": {
      "colab": {
        "base_uri": "https://localhost:8080/",
        "height": 661
      },
      "id": "CA4iQPyhG8eK",
      "outputId": "bbd9f40d-714c-4fdb-ae80-0c9e2f3688b6"
    },
    "execution_count": 18,
    "outputs": [
      {
        "output_type": "display_data",
        "data": {
          "text/html": [
            "<table class=\"show_images\" style=\"border-spacing:0px;\"><tr><td style=\"padding:1px;\"><div style=\"displa
            "      <div style=\"display:flex; flex-direction:column; align-items:center;\">\n",
            "      <div>Original Image</div><div><img width=\"256\" height=\"256\" style=\"image-rendering:pixelated; obje
            "      <div style=\"display:flex; flex-direction:column; align-items:center;\">\n",
            "      <div>Original Degraded Image</div><div><img width=\"256\" height=\"256\" style=\"image-rendering:pixela
            "      <div style=\"display:flex; flex-direction:column; align-items:center;\">\n",
            "      <div>Generated Image</div><div><img width=\"256\" height=\"256\" style=\"image-rendering:pixelated; obj
            "      <div style=\"display:flex; flex-direction:column; align-items:center;\">\n",
            "      <div>Generated Degraded Image</div><div><img width=\"256\" height=\"256\" style=\"image-rendering:pixel
          ],
          "text/plain": [
            "<IPython.core.display.HTML object>"
          ]
        },
        "metadata": {}
      },
      {
        "output_type": "stream",
        "name": "stdout",
        "text": [
          "step  781/800: percep_loss 0.10 latent_dist_reg 0.31 loss 0.14 \n",
          "step  782/800: percep_loss 0.10 latent_dist_reg 0.31 loss 0.14 \n",
          "step  783/800: percep_loss 0.10 latent_dist_reg 0.31 loss 0.14 \n",
          "step  784/800: percep_loss 0.10 latent_dist_reg 0.31 loss 0.14 \n",
          "step  785/800: percep_loss 0.10 latent_dist_reg 0.31 loss 0.14 \n",
          "step  786/800: percep_loss 0.10 latent_dist_reg 0.31 loss 0.14 \n",
          "step  787/800: percep_loss 0.10 latent_dist_reg 0.31 loss 0.14 \n",
          "step  788/800: percep_loss 0.10 latent_dist_reg 0.31 loss 0.14 \n",
          "step  789/800: percep_loss 0.10 latent_dist_reg 0.31 loss 0.14 \n",
          "step  790/800: percep_loss 0.10 latent_dist_reg 0.31 loss 0.14 \n",
          "step  791/800: percep_loss 0.10 latent_dist_reg 0.31 loss 0.14 \n",
          "step  792/800: percep_loss 0.10 latent_dist_reg 0.31 loss 0.14 \n",
          "step  793/800: percep_loss 0.10 latent_dist_reg 0.31 loss 0.14 \n",
          "step  794/800: percep_loss 0.10 latent_dist_reg 0.31 loss 0.14 \n",
          "step  795/800: percep_loss 0.10 latent_dist_reg 0.31 loss 0.14 \n",
          "step  796/800: percep_loss 0.10 latent_dist_reg 0.31 loss 0.14 \n",
          "step  797/800: percep_loss 0.10 latent_dist_reg 0.31 loss 0.14 \n",
          "step  798/800: percep_loss 0.10 latent_dist_reg 0.31 loss 0.14 \n",
          "step  799/800: percep_loss 0.10 latent_dist_reg 0.31 loss 0.14 \n",
```

```
            "step  800/800: percep_loss 0.10 latent_dist_reg 0.31 loss 0.14 \n",
            "Elapsed: 540.5 s\n"
          ]
        }
      ]
    },
    {
      "cell_type": "markdown",
      "source": [
        "**A plot of the optimization loss**"
      ],
      "metadata": {
        "id": "BCev1fKdc9UO"
      }
    },
    {
      "cell_type": "code",
      "source": [
        "import matplotlib.pyplot as plt\n",
        "\n",
        "def plot_optimization_loss(loss):\n",
        "    steps = np.arange(len(loss))\n",
        "    plt.plot(steps, loss.cpu().detach().numpy(), 'b-')\n",
        "    plt.xlabel('step')\n",
        "    plt.xticks(np.arange(0, len(loss) + 1, 100), rotation = (45), fontsize = 10)\n",
        "    plt.ylabel('loss')\n",
        "    plt.yscale('log')\n",
        "    plt.title('The Optimization Loss')\n",
        "    plt.tight_layout()\n",
        "    plt.show()\n",
        "    # plt.savefig('/content/gdrive/MyDrive/IMPRO_EX5_2021/output_images/LossPlot.png')\n"
      ],
      "metadata": {
        "id": "SUU2Tdfuc8VK"
      },
      "execution_count": 10,
      "outputs": []
    },
    {
      "cell_type": "code",
      "source": [
        "plot_optimization_loss(opt_losses)"
      ],
      "metadata": {
        "id": "0VTT4H5P-jpb",
        "colab": {
          "base_uri": "https://localhost:8080/",
          "height": 297
        },
        "outputId": "4a40cbb7-7815-412a-fd1d-6a2dd42841ac"
      },
      "execution_count": 19,
      "outputs": [
        {
          "output_type": "display_data",
          "data": {
            "image/png": "iVBORw0KGgoAAAANSUhEUgAAAagAAAEYCAYAAAAJeGK1AAAABHNCSVQICAgIfAhkiAAAAAlwSFlzAAALEgAACxIB0t1+/AAAAD
            "text/plain": [
              "<Figure size 432x288 with 1 Axes>"
            ]
          },
          "metadata": {
            "needs_background": "light"
          }
        }
      ]
    },
    {
```

```
    "cell_type": "markdown",
    "source": [
      "# Image Reconstruction Tasks"
    ],
    "metadata": {
      "id": "uuNq08zWq7f9"
    }
  },
  {
    "cell_type": "markdown",
    "source": [
      "**Image Deblurring**"
    ],
    "metadata": {
      "id": "2DvmVh0jrEEC"
    }
  },
  {
    "cell_type": "code",
    "source": [
      "\n",
      "opt_losses = invert_image(degradation_mode=GAUSSIAN_BLUR_DEGRADATION,\n",
      "                          target_fname='/content/gdrive/MyDrive/deblurring/im1/Christopher_Campbell_aligned.png',\n
      "                          outdir='/content/gdrive/MyDrive/deblurring/im1',\n",
      "                          seed=303,\n",
      "                          num_steps=1000,\n",
      "                          latent_dist_reg_weight=0.1)"
    ],
    "metadata": {
      "colab": {
        "base_uri": "https://localhost:8080/",
        "height": 661
      },
      "id": "11bGRs8prIFd",
      "outputId": "654c64c8-c66e-43b2-a7ed-034dc8cad524"
    },
    "execution_count": 15,
    "outputs": [
      {
        "output_type": "display_data",
        "data": {
          "text/html": [
            "<table class=\"show_images\" style=\"border-spacing:0px;\"><tr><td style=\"padding:1px;\"><div style=\"displa
            "      <div style=\"display:flex; flex-direction:column; align-items:center;\">\n",
            "      <div>Original Image</div><div><img width=\"256\" height=\"256\" style=\"image-rendering:pixelated; obje
            "      <div style=\"display:flex; flex-direction:column; align-items:center;\">\n",
            "      <div>Original Degraded Image</div><div><img width=\"256\" height=\"256\" style=\"image-rendering:pixela
            "      <div style=\"display:flex; flex-direction:column; align-items:center;\">\n",
            "      <div>Generated Image</div><div><img width=\"256\" height=\"256\" style=\"image-rendering:pixelated; obj
            "      <div style=\"display:flex; flex-direction:column; align-items:center;\">\n",
            "      <div>Generated Degraded Image</div><div><img width=\"256\" height=\"256\" style=\"image-rendering:pixel
          ],
          "text/plain": [
            "<IPython.core.display.HTML object>"
          ]
        },
        "metadata": {}
      },
      {
        "output_type": "stream",
        "name": "stdout",
        "text": [
          "step  981/1000: percep_loss 0.04 latent_dist_reg 0.25 loss 0.06 \n",
          "step  982/1000: percep_loss 0.04 latent_dist_reg 0.25 loss 0.06 \n",
          "step  983/1000: percep_loss 0.04 latent_dist_reg 0.25 loss 0.06 \n",
          "step  984/1000: percep_loss 0.04 latent_dist_reg 0.25 loss 0.06 \n",
          "step  985/1000: percep_loss 0.04 latent_dist_reg 0.25 loss 0.06 \n",
          "step  986/1000: percep_loss 0.04 latent_dist_reg 0.25 loss 0.06 \n",
```

```
            "step  987/1000: percep_loss 0.04 latent_dist_reg 0.25 loss 0.06 \n",
            "step  988/1000: percep_loss 0.04 latent_dist_reg 0.25 loss 0.06 \n",
            "step  989/1000: percep_loss 0.04 latent_dist_reg 0.25 loss 0.06 \n",
            "step  990/1000: percep_loss 0.04 latent_dist_reg 0.25 loss 0.06 \n",
            "step  991/1000: percep_loss 0.04 latent_dist_reg 0.25 loss 0.06 \n",
            "step  992/1000: percep_loss 0.04 latent_dist_reg 0.25 loss 0.06 \n",
            "step  993/1000: percep_loss 0.04 latent_dist_reg 0.25 loss 0.06 \n",
            "step  994/1000: percep_loss 0.04 latent_dist_reg 0.25 loss 0.06 \n",
            "step  995/1000: percep_loss 0.04 latent_dist_reg 0.25 loss 0.06 \n",
            "step  996/1000: percep_loss 0.04 latent_dist_reg 0.25 loss 0.06 \n",
            "step  997/1000: percep_loss 0.04 latent_dist_reg 0.25 loss 0.06 \n",
            "step  998/1000: percep_loss 0.04 latent_dist_reg 0.25 loss 0.06 \n",
            "step  999/1000: percep_loss 0.04 latent_dist_reg 0.25 loss 0.06 \n",
            "step 1000/1000: percep_loss 0.04 latent_dist_reg 0.25 loss 0.06 \n",
            "Elapsed: 345.4 s\n"
          ]
        }
      ]
    },
    {
      "cell_type": "code",
      "source": [
        "plot_optimization_loss(opt_losses)"
      ],
      "metadata": {
        "id": "dQ4YubC9esWY",
        "colab": {
          "base_uri": "https://localhost:8080/",
          "height": 297
        },
        "outputId": "435f2163-b21a-4e9f-defd-f68e67f85a1f"
      },
      "execution_count": 12,
      "outputs": [
        {
          "output_type": "display_data",
          "data": {
            "image/png": "iVBORw0KGgoAAAANSUhEUgAAAagAAAEYCAYAAAAJeGK1AAAABHNCSVQICAgIfAhkiAAAAAlwSFlzAAALEgAACxIB0t1+/AAAAD
            "text/plain": [
              "<Figure size 432x288 with 1 Axes>"
            ]
          },
          "metadata": {
            "needs_background": "light"
          }
        }
      ]
    },
    {
      "cell_type": "code",
      "source": [
        "opt_losses = invert_image(degradation_mode=GAUSSIAN_BLUR_DEGRADATION,\n",
        "                          target_fname='/content/gdrive/MyDrive/deblurring/im2/my_face_aligned.png',\n",
        "                          outdir='/content/gdrive/MyDrive/deblurring/im2',\n",
        "                          seed=303,\n",
        "                          num_steps=1000,\n",
        "                          latent_dist_reg_weight=0.1)"
      ],
      "metadata": {
        "id": "bdeXXRLlesQl",
        "colab": {
          "base_uri": "https://localhost:8080/",
          "height": 653
        },
        "outputId": "d16ff7d3-5fa5-4a5d-c50a-4f70db05a0b5"
      },
      "execution_count": null,
      "outputs": [
```

```
        {
          "output_type": "display_data",
          "data": {
            "text/html": [
              "<table class=\"show_images\" style=\"border-spacing:0px;\"><tr><td style=\"padding:1px;\"><div style=\"displa
              "      <div style=\"display:flex; flex-direction:column; align-items:center;\">\n",
              "      <div>Original Image</div><div><img width=\"256\" height=\"256\" style=\"image-rendering:pixelated; obje
              "      <div style=\"display:flex; flex-direction:column; align-items:center;\">\n",
              "      <div>Original Degraded Image</div><div><img width=\"256\" height=\"256\" style=\"image-rendering:pixela
              "      <div style=\"display:flex; flex-direction:column; align-items:center;\">\n",
              "      <div>Generated Image</div><div><img width=\"256\" height=\"256\" style=\"image-rendering:pixelated; obj
              "      <div style=\"display:flex; flex-direction:column; align-items:center;\">\n",
              "      <div>Generated Degraded Image</div><div><img width=\"256\" height=\"256\" style=\"image-rendering:pixel
            ],
            "text/plain": [
              "<IPython.core.display.HTML object>"
            ]
          },
          "metadata": {}
        },
        {
          "output_type": "stream",
          "name": "stdout",
          "text": [
            "step  981/1000: percep_loss 0.04 latent_dist_reg 0.22 loss 0.06 \n",
            "step  982/1000: percep_loss 0.04 latent_dist_reg 0.22 loss 0.06 \n",
            "step  983/1000: percep_loss 0.04 latent_dist_reg 0.22 loss 0.06 \n",
            "step  984/1000: percep_loss 0.04 latent_dist_reg 0.22 loss 0.06 \n",
            "step  985/1000: percep_loss 0.04 latent_dist_reg 0.22 loss 0.06 \n",
            "step  986/1000: percep_loss 0.04 latent_dist_reg 0.22 loss 0.06 \n",
            "step  987/1000: percep_loss 0.04 latent_dist_reg 0.22 loss 0.06 \n",
            "step  988/1000: percep_loss 0.04 latent_dist_reg 0.22 loss 0.06 \n",
            "step  989/1000: percep_loss 0.04 latent_dist_reg 0.22 loss 0.06 \n",
            "step  990/1000: percep_loss 0.04 latent_dist_reg 0.22 loss 0.06 \n",
            "step  991/1000: percep_loss 0.04 latent_dist_reg 0.22 loss 0.06 \n",
            "step  992/1000: percep_loss 0.04 latent_dist_reg 0.22 loss 0.06 \n",
            "step  993/1000: percep_loss 0.04 latent_dist_reg 0.22 loss 0.06 \n",
            "step  994/1000: percep_loss 0.04 latent_dist_reg 0.22 loss 0.06 \n",
            "step  995/1000: percep_loss 0.04 latent_dist_reg 0.22 loss 0.06 \n",
            "step  996/1000: percep_loss 0.04 latent_dist_reg 0.22 loss 0.06 \n",
            "step  997/1000: percep_loss 0.04 latent_dist_reg 0.22 loss 0.06 \n",
            "step  998/1000: percep_loss 0.04 latent_dist_reg 0.22 loss 0.06 \n",
            "step  999/1000: percep_loss 0.04 latent_dist_reg 0.22 loss 0.06 \n",
            "step 1000/1000: percep_loss 0.04 latent_dist_reg 0.22 loss 0.06 \n",
            "Elapsed: 260.9 s\n"
          ]
        }
      ]
    },
    {
      "cell_type": "code",
      "source": [
        "plot_optimization_loss(opt_losses)"
      ],
      "metadata": {
        "id": "aWtQgNkterlM",
        "colab": {
          "base_uri": "https://localhost:8080/",
          "height": 297
        },
        "outputId": "6164515a-8531-4564-cd1d-28339d48074d"
      },
      "execution_count": null,
      "outputs": [
        {
          "output_type": "display_data",
          "data": {
            "image/png": "iVBORw0KGgoAAAANSUhEUgAAAagAAAEYCAYAAAAJeGK1AAAABHNCSVQICAgIfAhkiAAAAAlwSFlzAAALEgAACxIB0t1+/AAAAD
```

```
            "text/plain": [
              "<Figure size 432x288 with 1 Axes>"
            ]
          },
          "metadata": {
            "needs_background": "light"
          }
        }
      ]
    },
    {
      "cell_type": "markdown",
      "source": [
        "**Deblurring on the given blurry image of Yann LeCun**"
      ],
      "metadata": {
        "id": "_wNVoFb54YMB"
      }
    },
    {
      "cell_type": "markdown",
      "source": [
        "In order to run the next cell, comment the line:\n",
        "\n",
        "`target_images = blur_spatial(target_images, kernel)`\n",
        "\n",
        "in the function:  `invert_image`  under the section:\n",
        "\n",
        "`elif degradation_mode == GAUSSIAN_BLUR_DEGRADATION:`\n",
        "\n",
        "\n"
      ],
      "metadata": {
        "id": "MphzBo03ayxH"
      }
    },
    {
      "cell_type": "code",
      "source": [
        "opt_losses = invert_image(degradation_mode=GAUSSIAN_BLUR_DEGRADATION,\n",
        "                          target_fname='/content/gdrive/MyDrive/deblurring/Yann LeCun/yann_lecun_blur.png',\n",
        "                          outdir='/content/gdrive/MyDrive/deblurring/Yann LeCun',\n",
        "                          seed=303,\n",
        "                          num_steps=2000,\n",
        "                          latent_dist_reg_weight=0.005)"
      ],
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/",
          "height": 661
        },
        "id": "gNJDp15n4mXg",
        "outputId": "26ed2faf-429e-40a1-c360-a9341fcf0943"
      },
      "execution_count": 15,
      "outputs": [
        {
          "output_type": "display_data",
          "data": {
            "text/html": [
              "<table class=\"show_images\" style=\"border-spacing:0px;\"><tr><td style=\"padding:1px;\"><div style=\"displa
              "      <div style=\"display:flex; flex-direction:column; align-items:center;\">\n",
              "      <div>Original Image</div><div><img width=\"256\" height=\"256\" style=\"image-rendering:pixelated; obje
              "      <div style=\"display:flex; flex-direction:column; align-items:center;\">\n",
              "      <div>Original Degraded Image</div><div><img width=\"256\" height=\"256\" style=\"image-rendering:pixela
              "      <div style=\"display:flex; flex-direction:column; align-items:center;\">\n",
              "      <div>Generated Image</div><div><img width=\"256\" height=\"256\" style=\"image-rendering:pixelated; obj
              "      <div style=\"display:flex; flex-direction:column; align-items:center;\">\n",
```

```
              "      <div>Generated Degraded Image</div><div><img width=\"256\" height=\"256\" style=\"image-rendering:pixel
            ],
            "text/plain": [
              "<IPython.core.display.HTML object>"
            ]
          },
          "metadata": {}
        },
        {
          "output_type": "stream",
          "name": "stdout",
          "text": [
            "step 1981/2000: percep_loss 0.05 latent_dist_reg 1.36 loss 0.06 \n",
            "step 1982/2000: percep_loss 0.05 latent_dist_reg 1.36 loss 0.06 \n",
            "step 1983/2000: percep_loss 0.05 latent_dist_reg 1.36 loss 0.06 \n",
            "step 1984/2000: percep_loss 0.05 latent_dist_reg 1.36 loss 0.06 \n",
            "step 1985/2000: percep_loss 0.05 latent_dist_reg 1.36 loss 0.06 \n",
            "step 1986/2000: percep_loss 0.05 latent_dist_reg 1.36 loss 0.06 \n",
            "step 1987/2000: percep_loss 0.05 latent_dist_reg 1.36 loss 0.06 \n",
            "step 1988/2000: percep_loss 0.05 latent_dist_reg 1.36 loss 0.06 \n",
            "step 1989/2000: percep_loss 0.05 latent_dist_reg 1.36 loss 0.06 \n",
            "step 1990/2000: percep_loss 0.05 latent_dist_reg 1.36 loss 0.06 \n",
            "step 1991/2000: percep_loss 0.05 latent_dist_reg 1.36 loss 0.06 \n",
            "step 1992/2000: percep_loss 0.05 latent_dist_reg 1.36 loss 0.06 \n",
            "step 1993/2000: percep_loss 0.05 latent_dist_reg 1.36 loss 0.06 \n",
            "step 1994/2000: percep_loss 0.05 latent_dist_reg 1.36 loss 0.06 \n",
            "step 1995/2000: percep_loss 0.05 latent_dist_reg 1.36 loss 0.06 \n",
            "step 1996/2000: percep_loss 0.05 latent_dist_reg 1.36 loss 0.06 \n",
            "step 1997/2000: percep_loss 0.05 latent_dist_reg 1.36 loss 0.06 \n",
            "step 1998/2000: percep_loss 0.05 latent_dist_reg 1.36 loss 0.06 \n",
            "step 1999/2000: percep_loss 0.05 latent_dist_reg 1.36 loss 0.06 \n",
            "step 2000/2000: percep_loss 0.05 latent_dist_reg 1.36 loss 0.06 \n",
            "Elapsed: 1402.8 s\n"
          ]
        }
      ]
    },
    {
      "cell_type": "code",
      "source": [
        "plot_optimization_loss(opt_losses)"
      ],
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/",
          "height": 297
        },
        "id": "EGjBmD39-Y_j",
        "outputId": "170d454d-6b11-45c0-cbb4-1d37319339e9"
      },
      "execution_count": null,
      "outputs": [
        {
          "output_type": "display_data",
          "data": {
            "image/png": "iVBORwOKGgoAAAANSUhEUgAAAagAAAEYCAYAAAAJeGK1AAAABHNCSVQICAgIfAhkiAAAAAlwSFlzAAALEgAACxIB0t1+/AAAAD
            "text/plain": [
              "<Figure size 432x288 with 1 Axes>"
            ]
          },
          "metadata": {
            "needs_background": "light"
          }
        }
      ]
    },
    {
      "cell_type": "markdown",
```

```
    "source": [
      "**Image Colorization**"
    ],
    "metadata": {
      "id": "Rzqos7yN-5cf"
    }
  },
  {
    "cell_type": "code",
    "source": [
      "opt_losses = invert_image(degradation_mode=GRAYSCALE_DEGRADATION,\n",
      "                          target_fname='/content/gdrive/MyDrive/Colorization/im1/Christopher_Campbell_aligned.png',
      "                          outdir='/content/gdrive/MyDrive/Colorization/im1',\n",
      "                          seed=613,\n",
      "                          num_steps=900,\n",
      "                          latent_dist_reg_weight=1)"
    ],
    "metadata": {
      "colab": {
        "base_uri": "https://localhost:8080/",
        "height": 653
      },
      "id": "-xf3Slkd--jK",
      "outputId": "07ffd2bb-26fe-4325-b5da-37adea66584b"
    },
    "execution_count": null,
    "outputs": [
      {
        "output_type": "display_data",
        "data": {
          "text/html": [
            "<table class=\"show_images\" style=\"border-spacing:0px;\"><tr><td style=\"padding:1px;\"><div style=\"displa
            "      <div style=\"display:flex; flex-direction:column; align-items:center;\">\n",
            "      <div>Original Image</div><div><img width=\"256\" height=\"256\" style=\"image-rendering:pixelated; obje
            "      <div style=\"display:flex; flex-direction:column; align-items:center;\">\n",
            "      <div>Original Degraded Image</div><div><img width=\"256\" height=\"256\" style=\"image-rendering:pixela
            "      <div style=\"display:flex; flex-direction:column; align-items:center;\">\n",
            "      <div>Generated Image</div><div><img width=\"256\" height=\"256\" style=\"image-rendering:pixelated; obj
            "      <div style=\"display:flex; flex-direction:column; align-items:center;\">\n",
            "      <div>Generated Degraded Image</div><div><img width=\"256\" height=\"256\" style=\"image-rendering:pixel
          ],
          "text/plain": [
            "<IPython.core.display.HTML object>"
          ]
        },
        "metadata": {}
      },
      {
        "output_type": "stream",
        "name": "stdout",
        "text": [
          "step  881/900: percep_loss 0.15 latent_dist_reg 0.06 loss 0.21 \n",
          "step  882/900: percep_loss 0.15 latent_dist_reg 0.06 loss 0.21 \n",
          "step  883/900: percep_loss 0.15 latent_dist_reg 0.06 loss 0.21 \n",
          "step  884/900: percep_loss 0.15 latent_dist_reg 0.06 loss 0.21 \n",
          "step  885/900: percep_loss 0.15 latent_dist_reg 0.06 loss 0.21 \n",
          "step  886/900: percep_loss 0.15 latent_dist_reg 0.06 loss 0.21 \n",
          "step  887/900: percep_loss 0.15 latent_dist_reg 0.06 loss 0.21 \n",
          "step  888/900: percep_loss 0.15 latent_dist_reg 0.06 loss 0.21 \n",
          "step  889/900: percep_loss 0.15 latent_dist_reg 0.06 loss 0.21 \n",
          "step  890/900: percep_loss 0.15 latent_dist_reg 0.06 loss 0.21 \n",
          "step  891/900: percep_loss 0.15 latent_dist_reg 0.06 loss 0.21 \n",
          "step  892/900: percep_loss 0.15 latent_dist_reg 0.06 loss 0.21 \n",
          "step  893/900: percep_loss 0.15 latent_dist_reg 0.06 loss 0.21 \n",
          "step  894/900: percep_loss 0.15 latent_dist_reg 0.06 loss 0.21 \n",
          "step  895/900: percep_loss 0.15 latent_dist_reg 0.06 loss 0.21 \n",
          "step  896/900: percep_loss 0.15 latent_dist_reg 0.06 loss 0.21 \n",
          "step  897/900: percep_loss 0.15 latent_dist_reg 0.06 loss 0.21 \n",
```

```
            "step  898/900: percep_loss 0.15 latent_dist_reg 0.06 loss 0.21 \n",
            "step  899/900: percep_loss 0.15 latent_dist_reg 0.06 loss 0.21 \n",
            "step  900/900: percep_loss 0.15 latent_dist_reg 0.06 loss 0.21 \n",
            "Elapsed: 229.4 s\n"
          ]
        }
      ]
    },
    {
      "cell_type": "code",
      "source": [
        "plot_optimization_loss(opt_losses)"
      ],
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/",
          "height": 297
        },
        "id": "ra0im_ByVdjb",
        "outputId": "92561da3-4bfc-4297-8105-5675440609e9"
      },
      "execution_count": null,
      "outputs": [
        {
          "output_type": "display_data",
          "data": {
            "image/png": "iVBORw0KGgoAAAANSUhEUgAAAagAAAEYCAYAAAAJeGK1AAAABHNCSVQICAgIfAhkiAAAAAlwSFlzAAALEgAACxIB0t1+/AAAAD
            "text/plain": [
              "<Figure size 432x288 with 1 Axes>"
            ]
          },
          "metadata": {
            "needs_background": "light"
          }
        }
      ]
    },
    {
      "cell_type": "code",
      "source": [
        "opt_losses = invert_image(degradation_mode=GRAYSCALE_DEGRADATION,\n",
        "                          target_fname='/content/gdrive/MyDrive/Colorization/im2/my_face_aligned.png',\n",
        "                          outdir='/content/gdrive/MyDrive/Colorization/im2',\n",
        "                          seed=303,\n",
        "                          num_steps=1000,\n",
        "                          latent_dist_reg_weight=2)"
      ],
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/",
          "height": 653
        },
        "id": "sv2q8h45Vdb9",
        "outputId": "c6e6ee4f-f337-4c43-ddcb-edf10b020cab"
      },
      "execution_count": null,
      "outputs": [
        {
          "output_type": "display_data",
          "data": {
            "text/html": [
              "<table class=\"show_images\" style=\"border-spacing:0px;\"><tr><td style=\"padding:1px;\"><div style=\"displa
              "      <div style=\"display:flex; flex-direction:column; align-items:center;\">\n",
              "      <div>Original Image</div><div><img width=\"256\" height=\"256\" style=\"image-rendering:pixelated; obje
              "      <div style=\"display:flex; flex-direction:column; align-items:center;\">\n",
              "      <div>Original Degraded Image</div><div><img width=\"256\" height=\"256\" style=\"image-rendering:pixela
              "      <div style=\"display:flex; flex-direction:column; align-items:center;\">\n",
              "      <div>Generated Image</div><div><img width=\"256\" height=\"256\" style=\"image-rendering:pixelated; obj
```

```
              "      <div style=\"display:flex; flex-direction:column; align-items:center;\">\n",
              "      <div>Generated Degraded Image</div><div><img width=\"256\" height=\"256\" style=\"image-rendering:pixel
            ],
            "text/plain": [
              "<IPython.core.display.HTML object>"
            ]
          },
          "metadata": {}
        },
        {
          "output_type": "stream",
          "name": "stdout",
          "text": [
            "step  981/1000: percep_loss 0.17 latent_dist_reg 0.03 loss 0.22 \n",
            "step  982/1000: percep_loss 0.17 latent_dist_reg 0.03 loss 0.22 \n",
            "step  983/1000: percep_loss 0.17 latent_dist_reg 0.03 loss 0.22 \n",
            "step  984/1000: percep_loss 0.17 latent_dist_reg 0.03 loss 0.22 \n",
            "step  985/1000: percep_loss 0.17 latent_dist_reg 0.03 loss 0.22 \n",
            "step  986/1000: percep_loss 0.17 latent_dist_reg 0.03 loss 0.22 \n",
            "step  987/1000: percep_loss 0.17 latent_dist_reg 0.03 loss 0.22 \n",
            "step  988/1000: percep_loss 0.17 latent_dist_reg 0.03 loss 0.22 \n",
            "step  989/1000: percep_loss 0.17 latent_dist_reg 0.03 loss 0.22 \n",
            "step  990/1000: percep_loss 0.17 latent_dist_reg 0.03 loss 0.22 \n",
            "step  991/1000: percep_loss 0.17 latent_dist_reg 0.03 loss 0.22 \n",
            "step  992/1000: percep_loss 0.17 latent_dist_reg 0.03 loss 0.22 \n",
            "step  993/1000: percep_loss 0.17 latent_dist_reg 0.03 loss 0.22 \n",
            "step  994/1000: percep_loss 0.17 latent_dist_reg 0.03 loss 0.22 \n",
            "step  995/1000: percep_loss 0.17 latent_dist_reg 0.03 loss 0.22 \n",
            "step  996/1000: percep_loss 0.17 latent_dist_reg 0.03 loss 0.22 \n",
            "step  997/1000: percep_loss 0.17 latent_dist_reg 0.03 loss 0.22 \n",
            "step  998/1000: percep_loss 0.17 latent_dist_reg 0.03 loss 0.22 \n",
            "step  999/1000: percep_loss 0.17 latent_dist_reg 0.03 loss 0.22 \n",
            "step 1000/1000: percep_loss 0.17 latent_dist_reg 0.03 loss 0.22 \n",
            "Elapsed: 244.5 s\n"
          ]
        }
      ]
    },
    {
      "cell_type": "code",
      "source": [
        "plot_optimization_loss(opt_losses)"
      ],
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/",
          "height": 297
        },
        "id": "FChkvoxPVc6F",
        "outputId": "738f5b70-aacf-4b73-b509-02b916f56742"
      },
      "execution_count": null,
      "outputs": [
        {
          "output_type": "display_data",
          "data": {
            "image/png": "iVBORw0KGgoAAAANSUhEUgAAAagAAAEYCAYAAAAJeGK1AAAABHNCSVQICAgIfAhkiAAAAAlwSFlzAAALEgAACxIB0t1+/AAAAD
            "text/plain": [
              "<Figure size 432x288 with 1 Axes>"
            ]
          },
          "metadata": {
            "needs_background": "light"
          }
        }
      ]
    },
    {
```

```
    "cell_type": "markdown",
    "source": [
      "**The colorization on the given grayscale image of Alan Turing**"
    ],
    "metadata": {
      "id": "EzV-NOQ6--97"
    }
  },
  {
    "cell_type": "code",
    "source": [
      "opt_losses = invert_image(degradation_mode=GRAYSCALE_DEGRADATION,\n",
      "                          target_fname='/content/gdrive/MyDrive/IMPRO_EX5_2021/Alan Turing/alan_turing_grayscale.pn
      "                          outdir='/content/gdrive/MyDrive/IMPRO_EX5_2021/Alan Turing',\n",
      "                          seed=303,\n",
      "                          num_steps=1500,\n",
      "                          latent_dist_reg_weight=1)"
    ],
    "metadata": {
      "id": "4IdNubSz_EFG",
      "colab": {
        "base_uri": "https://localhost:8080/",
        "height": 653
      },
      "outputId": "8d8206d2-3df9-4630-85d1-103f73285f6a"
    },
    "execution_count": null,
    "outputs": [
      {
        "output_type": "display_data",
        "data": {
          "text/html": [
            "<table class=\"show_images\" style=\"border-spacing:0px;\"><tr><td style=\"padding:1px;\"><div style=\"displa
            "      <div style=\"display:flex; flex-direction:column; align-items:center;\">\n",
            "      <div>Original Image</div><div><img width=\"256\" height=\"256\" style=\"image-rendering:pixelated; obje
            "      <div style=\"display:flex; flex-direction:column; align-items:center;\">\n",
            "      <div>Original Degraded Image</div><div><img width=\"256\" height=\"256\" style=\"image-rendering:pixela
            "      <div style=\"display:flex; flex-direction:column; align-items:center;\">\n",
            "      <div>Generated Image</div><div><img width=\"256\" height=\"256\" style=\"image-rendering:pixelated; obj
            "      <div style=\"display:flex; flex-direction:column; align-items:center;\">\n",
            "      <div>Generated Degraded Image</div><div><img width=\"256\" height=\"256\" style=\"image-rendering:pixel
          ],
          "text/plain": [
            "<IPython.core.display.HTML object>"
          ]
        },
        "metadata": {}
      },
      {
        "output_type": "stream",
        "name": "stdout",
        "text": [
          "step 1481/1500: percep_loss 0.15 latent_dist_reg 0.06 loss 0.20 \n",
          "step 1482/1500: percep_loss 0.15 latent_dist_reg 0.06 loss 0.20 \n",
          "step 1483/1500: percep_loss 0.15 latent_dist_reg 0.06 loss 0.20 \n",
          "step 1484/1500: percep_loss 0.15 latent_dist_reg 0.06 loss 0.20 \n",
          "step 1485/1500: percep_loss 0.15 latent_dist_reg 0.06 loss 0.20 \n",
          "step 1486/1500: percep_loss 0.15 latent_dist_reg 0.06 loss 0.20 \n",
          "step 1487/1500: percep_loss 0.15 latent_dist_reg 0.06 loss 0.20 \n",
          "step 1488/1500: percep_loss 0.15 latent_dist_reg 0.06 loss 0.20 \n",
          "step 1489/1500: percep_loss 0.15 latent_dist_reg 0.06 loss 0.20 \n",
          "step 1490/1500: percep_loss 0.15 latent_dist_reg 0.06 loss 0.20 \n",
          "step 1491/1500: percep_loss 0.15 latent_dist_reg 0.06 loss 0.20 \n",
          "step 1492/1500: percep_loss 0.15 latent_dist_reg 0.06 loss 0.20 \n",
          "step 1493/1500: percep_loss 0.15 latent_dist_reg 0.06 loss 0.20 \n",
          "step 1494/1500: percep_loss 0.15 latent_dist_reg 0.06 loss 0.20 \n",
          "step 1495/1500: percep_loss 0.15 latent_dist_reg 0.06 loss 0.20 \n",
          "step 1496/1500: percep_loss 0.15 latent_dist_reg 0.06 loss 0.20 \n",
```

```
            "step 1497/1500: percep_loss 0.15 latent_dist_reg 0.06 loss 0.20 \n",
            "step 1498/1500: percep_loss 0.15 latent_dist_reg 0.06 loss 0.20 \n",
            "step 1499/1500: percep_loss 0.15 latent_dist_reg 0.06 loss 0.20 \n",
            "step 1500/1500: percep_loss 0.15 latent_dist_reg 0.06 loss 0.20 \n",
            "Elapsed: 389.2 s\n"
          ]
        }
      ]
    },
    {
      "cell_type": "code",
      "source": [
        "plot_optimization_loss(opt_losses)"
      ],
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/",
          "height": 314
        },
        "id": "LdyRP7tc_Du7",
        "outputId": "c5c80024-5eda-4cf0-bd1e-6d67820d7d8c"
      },
      "execution_count": null,
      "outputs": [
        {
          "output_type": "display_data",
          "data": {
            "image/png": "iVBORw0KGgoAAAANSUhEUgAAAagAAAEYCAYAAAAJeGK1AAAABHNCSVQICAgIfAhkiAAAAAlwSFlzAAALEgAACxIB0t1+/AAAAD
            "text/plain": [
              "<Figure size 432x288 with 1 Axes>"
            ]
          },
          "metadata": {
            "needs_background": "light"
          }
        },
        {
          "output_type": "display_data",
          "data": {
            "text/plain": [
              "<Figure size 432x288 with 0 Axes>"
            ]
          },
          "metadata": {}
        }
      ]
    },
    {
      "cell_type": "markdown",
      "source": [
        "**Image Inpainting**"
      ],
      "metadata": {
        "id": "CSJVTuqpzII_"
      }
    },
    {
      "cell_type": "code",
      "source": [
        "opt_losses = invert_image(degradation_mode=INPAINTING_DEGRADATION,\n",
        "                          target_fname='/content/gdrive/MyDrive/inpainting/im1/Christopher_Campbell_aligned.png',\n
        "                          outdir='/content/gdrive/MyDrive/inpainting/im1',\n",
        "                          mask_fname='/content/gdrive/MyDrive/inpainting/im1/Christopher_Campbell_mask.png',\n",
        "                          seed=303,\n",
        "                          num_steps=1000,\n",
        "                          latent_dist_reg_weight=0.2)"
      ],
      "metadata": {
```

```
          "colab": {
            "base_uri": "https://localhost:8080/",
            "height": 653
          },
          "id": "PYdO0jEbNUSQ",
          "outputId": "6c279316-3ac9-4ef7-fa56-a3d61663a5e9"
        },
        "execution_count": null,
        "outputs": [
          {
            "output_type": "display_data",
            "data": {
              "text/html": [
                "<table class=\"show_images\" style=\"border-spacing:0px;\"><tr><td style=\"padding:1px;\"><div style=\"displa
                "      <div style=\"display:flex; flex-direction:column; align-items:center;\">\n",
                "      <div>Original Image</div><div><img width=\"256\" height=\"256\" style=\"image-rendering:pixelated; obje
```


```
**********************************************
*** THIS FILE WAS TOO BIG AND WAS TRUNCATED ***
**********************************************
```

4 ex5/unaligned images/Christopher Campbell.jpg

5 ex5/unaligned images/my face.png

6 ex5/Colorization/Alan Turing/alan turing grayscale.png

7 ex5/Colorization/Alan Turing/final inverted image.png

8 ex5/Colorization/Alan Turing/original degraded image.png

9 ex5/Colorization/im1 s900 w1/Christopher Campbell aligned.png

10 ex5/Colorization/im1 s900 w1/final inverted image.png

11 ex5/Colorization/im1 s900 w1/original degraded image.png

12 ex5/Colorization/im2 s900 w1/final inverted image.png

13 ex5/Colorization/im2 s900 w1/my face aligned.png

14 ex5/Colorization/im2 s900 w1/original degraded image.png

15 ex5/Deblurring/Yann LeCun/final inverted image.png

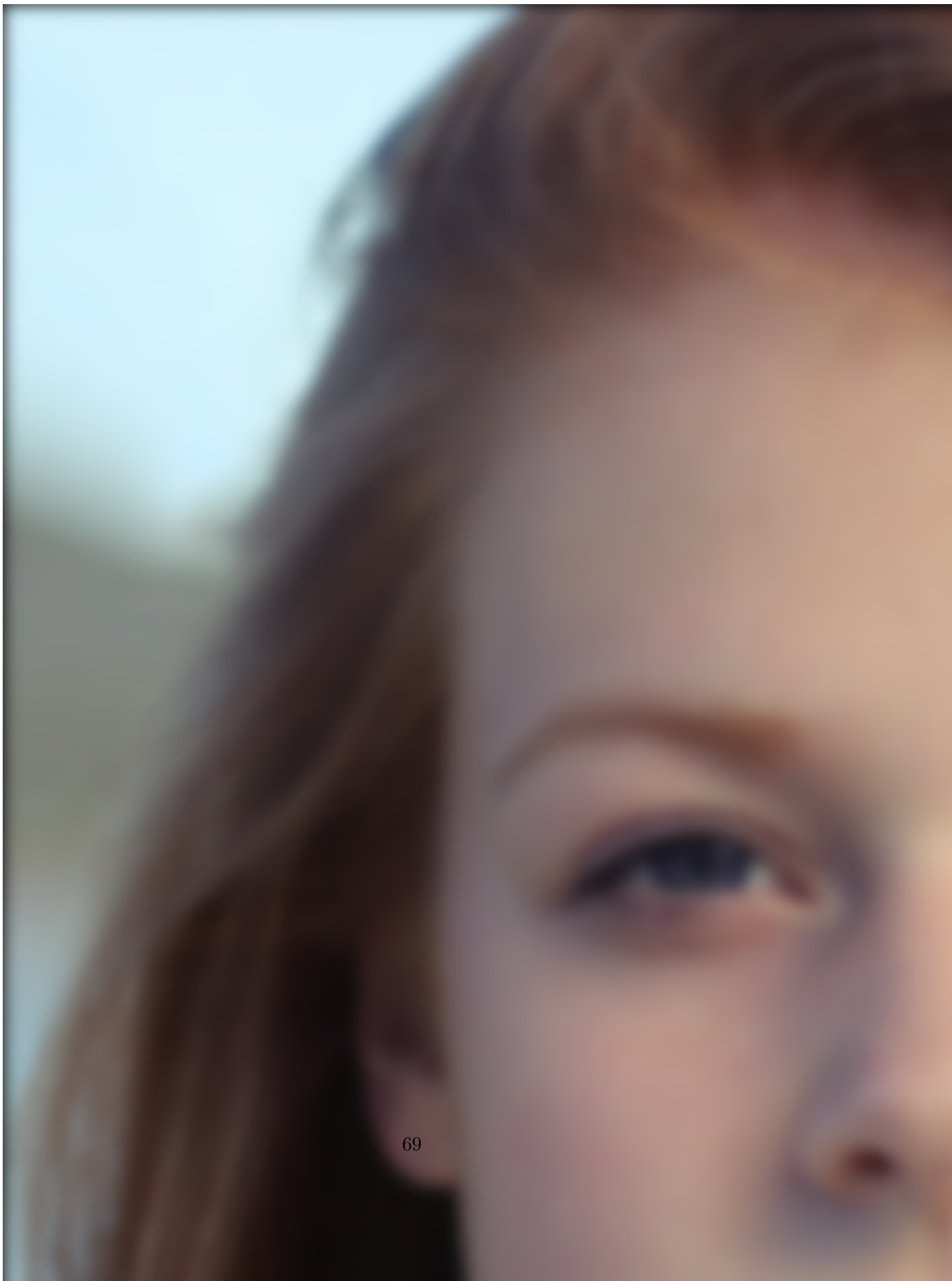
16 ex5/Deblurring/Yann LeCun/original degraded image.png

17 ex5/Deblurring/Yann LeCun/yann lecun blur.png

18 ex5/Deblurring/im1 k85 w0.001/Christopher Campbell aligned.png

19 ex5/Deblurring/im1 k85 w0.001/final inverted image.png

20 ex5/Deblurring/im1 k85 w0.001/original degraded image.png

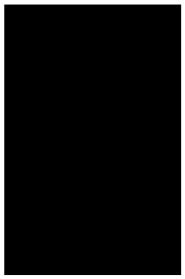


21 ex5/Deblurring/im2 k75 w0.1/final inverted image.png

22 ex5/Deblurring/im2 k75 w0.1/my face aligned.png

23 ex5/Deblurring/im2 k75 w0.1/original degraded image.png

24 ex5/Inpainting/fei fei li/fei fei li inpainting mask.png



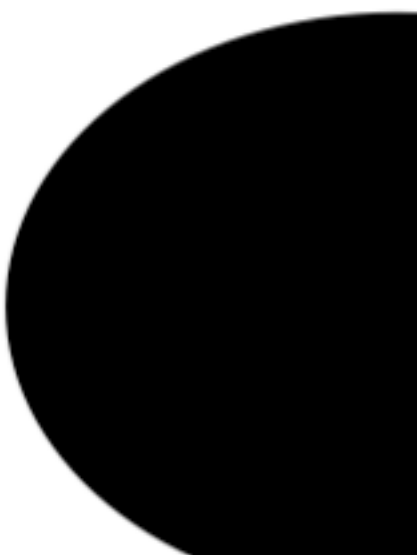
25 ex5/Inpainting/fei fei li/fei fei li original.png

26 ex5/Inpainting/fei fei li/final inverted image.png

27 ex5/Inpainting/fei fei li/original degraded image.png

28 ex5/Inpainting/paint im1 s1000 w0.2/Christopher Campbell aligned.png

29 ex5/Inpainting/paint im1 s1000 w0.2/Christopher Campbell mask.png



30 ex5/Inpainting/paint im1 s1000 w0.2/final inverted image.png

31 ex5/Inpainting/paint im1 s1000 w0.2/original degraded image.png

32 ex5/Inpainting/paint im2 s1000 w0.2/final inverted image.png

33 ex5/Inpainting/paint im2 s1000 w0.2/my face aligned.png

34 ex5/Inpainting/paint im2 s1000 w0.2/my face mask.png

35 ex5/Inpainting/paint im2 s1000 w0.2/original degraded image.png