

Contents

1 ex5/README	2
2 ex5/Ex5 results.pdf	3
3 ex5/IMPR Ex5 Deep Style Image Prior 2021 2022.ipynb	11
4 ex5/unaligned images/Christopher Campbell.jpg	36
5 ex5/unaligned images/my face.png	38
6 ex5/Colorization/Alan Turing/alan turing grayscale.png	39
7 ex5/Colorization/Alan Turing/final inverted image.png	41
8 ex5/Colorization/Alan Turing/original degraded image.png	43
9 ex5/Colorization/im1 s900 w1/Christopher Campbell aligned.png	45
10 ex5/Colorization/im1 s900 w1/final inverted image.png	47
11 ex5/Colorization/im1 s900 w1/original degraded image.png	49
12 ex5/Colorization/im2 s900 w1/final inverted image.png	51
13 ex5/Colorization/im2 s900 w1/my face aligned.png	53
14 ex5/Colorization/im2 s900 w1/original degraded image.png	55
15 ex5/Deblurring/Yann LeCun/final inverted image.png	57
16 ex5/Deblurring/Yann LeCun/original degraded image.png	59
17 ex5/Deblurring/Yann LeCun/yann lecun blur.png	61
18 ex5/Deblurring/im1 k85 w0.001/Christopher Campbell aligned.png	63
19 ex5/Deblurring/im1 k85 w0.001/final inverted image.png	65
20 ex5/Deblurring/im1 k85 w0.001/original degraded image.png	67
21 ex5/Deblurring/im2 k75 w0.1/final inverted image.png	69
22 ex5/Deblurring/im2 k75 w0.1/my face aligned.png	71
23 ex5/Deblurring/im2 k75 w0.1/original degraded image.png	73

24 ex5/Inpainting/fei fei li/fei fei li inpainting mask.png	75
25 ex5/Inpainting/fei fei li/fei fei li original.png	76
26 ex5/Inpainting/fei fei li/final inverted image.png	78
27 ex5/Inpainting/fei fei li/original degraded image.png	80
28 ex5/Inpainting/paint im1 s1000 w0.2/Christopher Campbell aligned.png	82
29 ex5/Inpainting/paint im1 s1000 w0.2/Christopher Campbell mask.png	84
30 ex5/Inpainting/paint im1 s1000 w0.2/final inverted image.png	86
31 ex5/Inpainting/paint im1 s1000 w0.2/original degraded image.png	88
32 ex5/Inpainting/paint im2 s1000 w0.2/final inverted image.png	90
33 ex5/Inpainting/paint im2 s1000 w0.2/my face aligned.png	92
34 ex5/Inpainting/paint im2 s1000 w0.2/my face mask.png	94
35 ex5/Inpainting/paint im2 s1000 w0.2/original degraded image.png	96

1 ex5/README

```
1 muaz.abdeen
2
3 files in the ZIP:
4 1- Ex5_results.pdf
5 2- IMPR_Ex5_Deep_Style_Image_Prior_2021_2022.ipynb
6 3- ./Deblurring/im1_k85_w0.001
7   ./Deblurring/im2_k75_w0.1
8   ./Deblurring/Yann LeCun
9 4- ./Colorization/im1_s900_w1
10   ./Colorization/im2_s900_w1
11   ./Colorization/Alan Turing
12 5- ./Inpainting/paint_im1_s1000_w0.2
13   ./Inpainting/paint_im2_s1000_w0.2
14   ./Inpainting/fei_fei_li
15
16 Each one of the 3-5 sub-folders contains 4 files:
17 1- inverted_latent.npz
18 2- original_image.png
19 3- original_degraded_image.png
20 4- final_inverted_image.png
```

67829 – Image Processing – Fall 2021

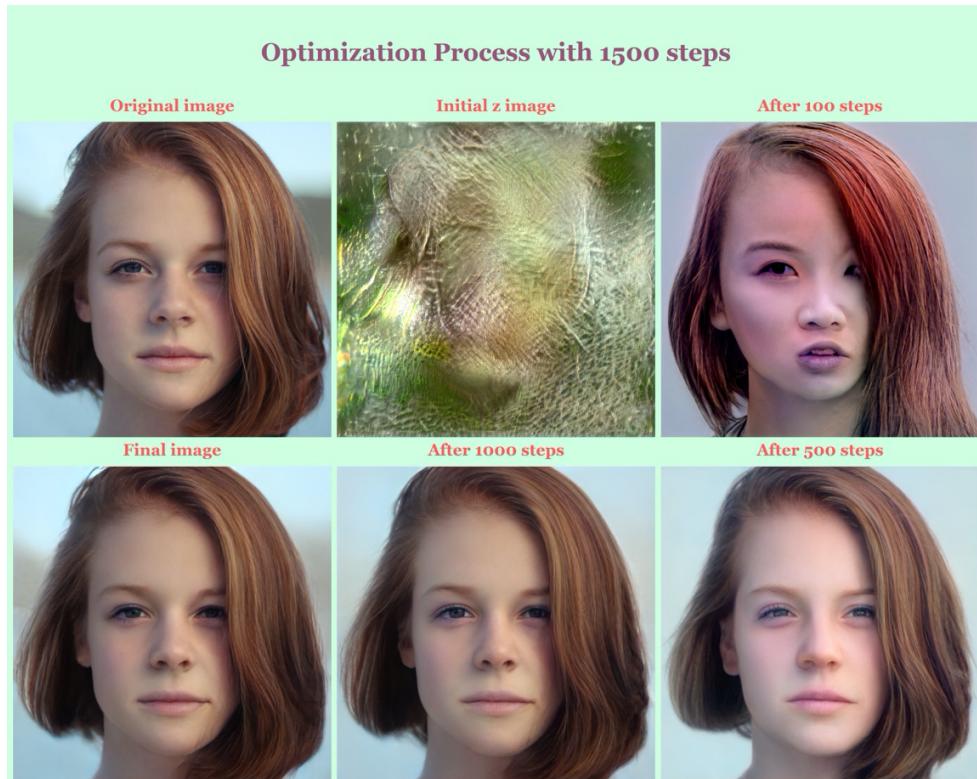
Exercise (5)

Muaz Abdeen – 300575297

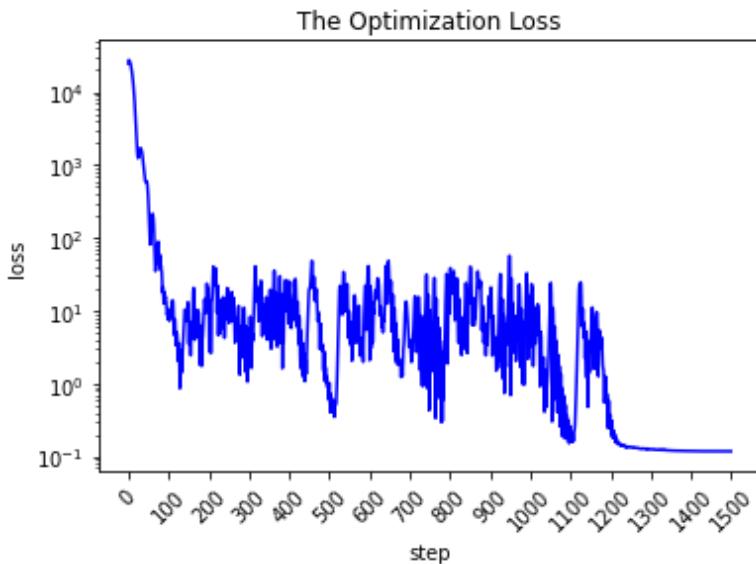
Image Alignment:



The Optimization Progressions:



A Plot of the Optimization Loss:



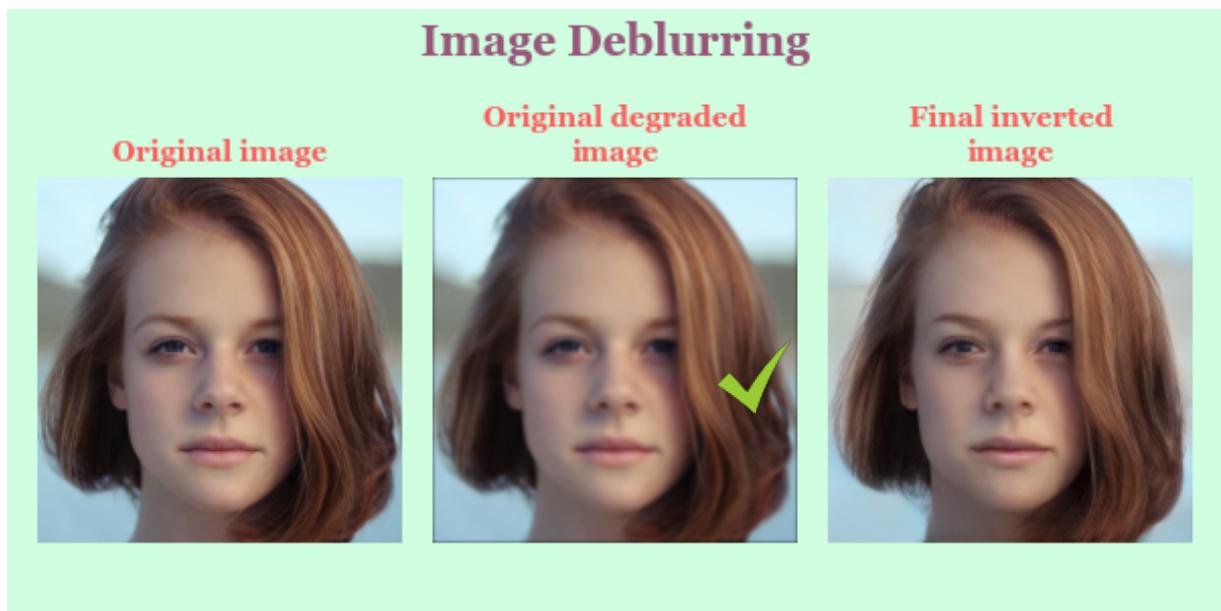
The effect of *latent_dist_reg_weight* and *num_steps* on the results:

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.

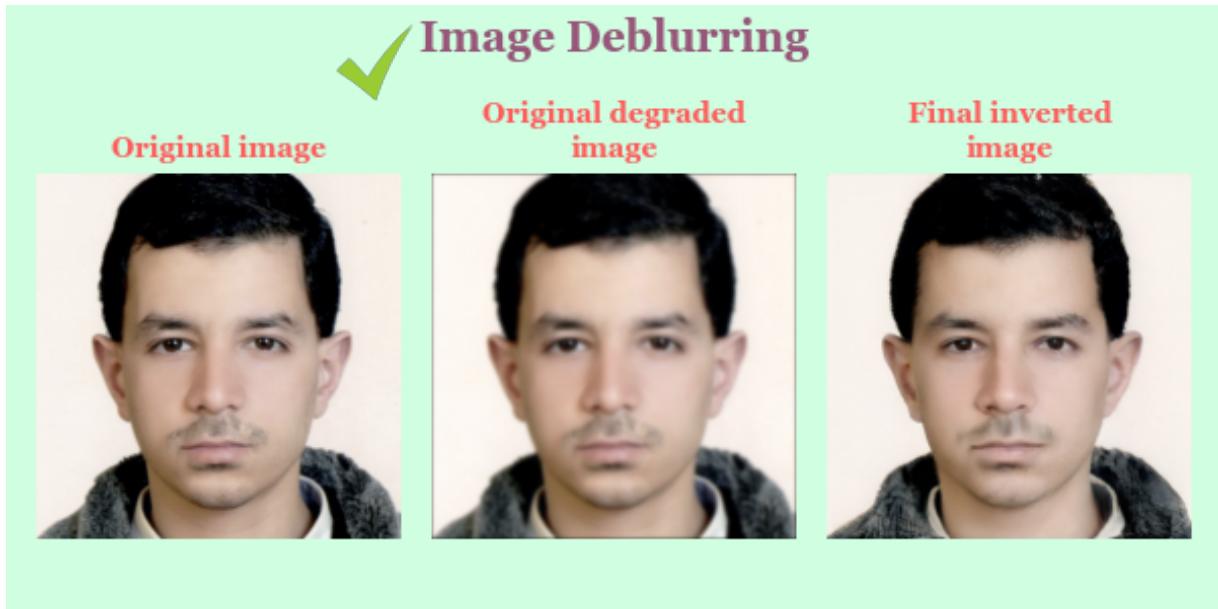
Also, increasing the *num_steps* will give us a better result, to some extent, that is after some threshold, there will be almost no improvement.

Image Deblurring:

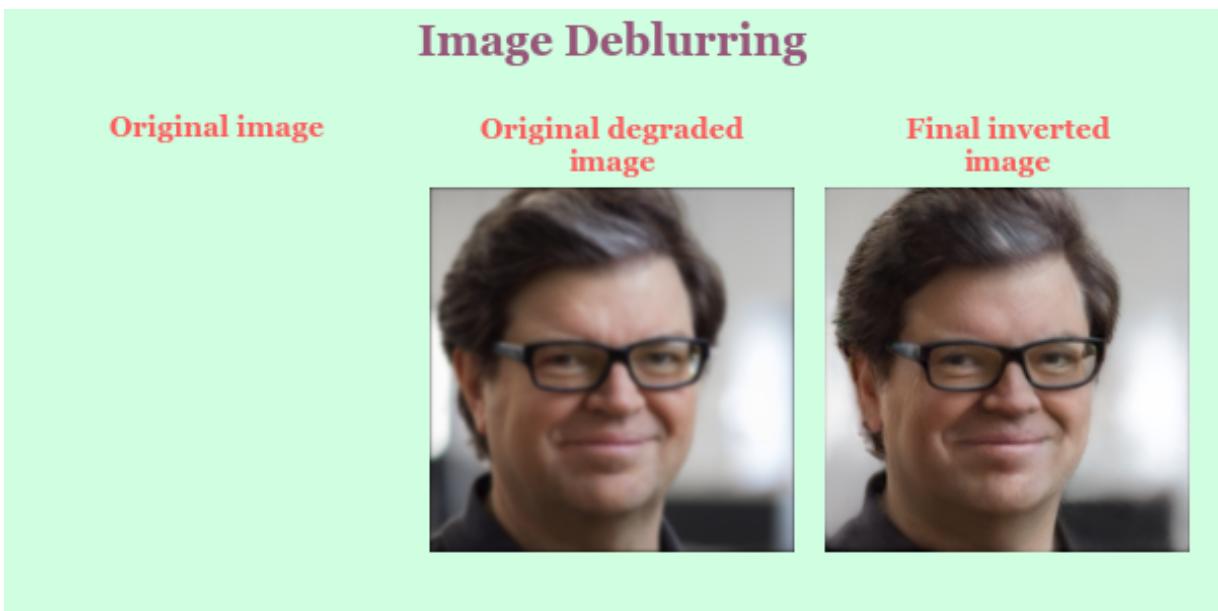
First image



Second image (my personal picture):



Yann LeCun



Solution Discussion:

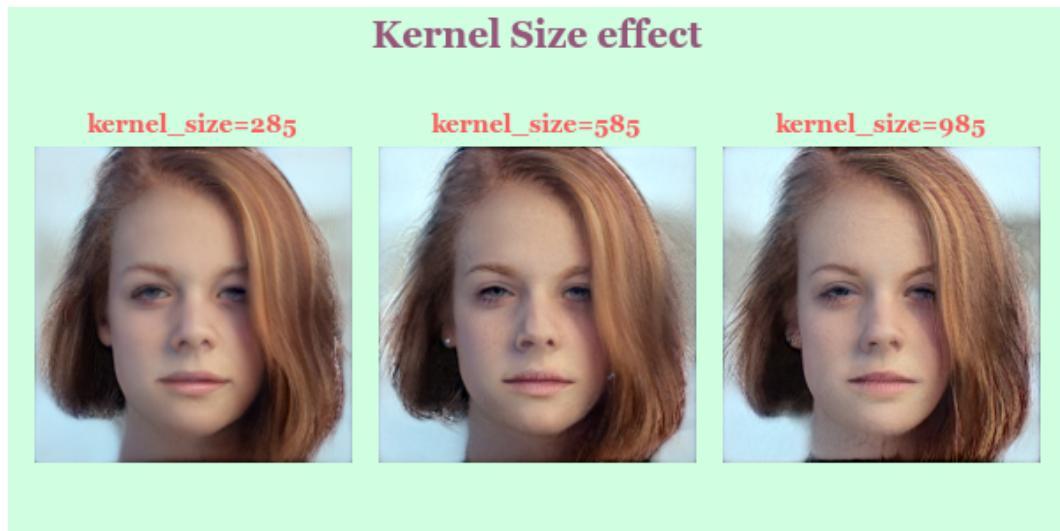
Hyperparameters: $kernel_size = 85$, $num_steps = 1000$,

$latent_dist_reg_weight = 0.001$

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.

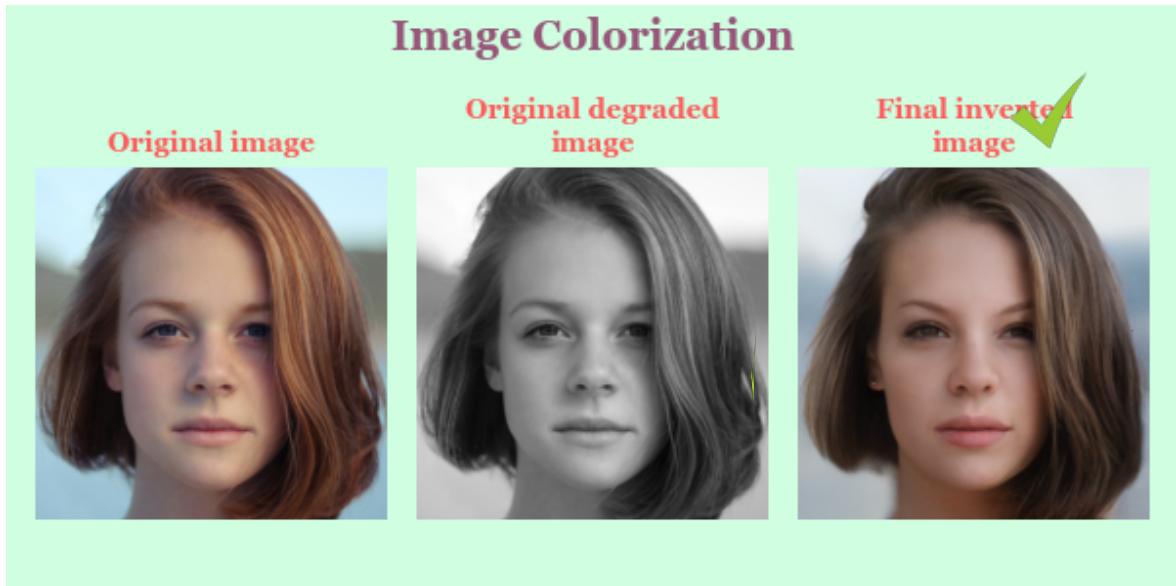


I noticed that after at most 1000 steps, the loss almost converged, and the result did not improve more. Also, and the bigger the *latent_dist_reg_weight* the closer the constructed image to the mean of natural images, and vice versa.

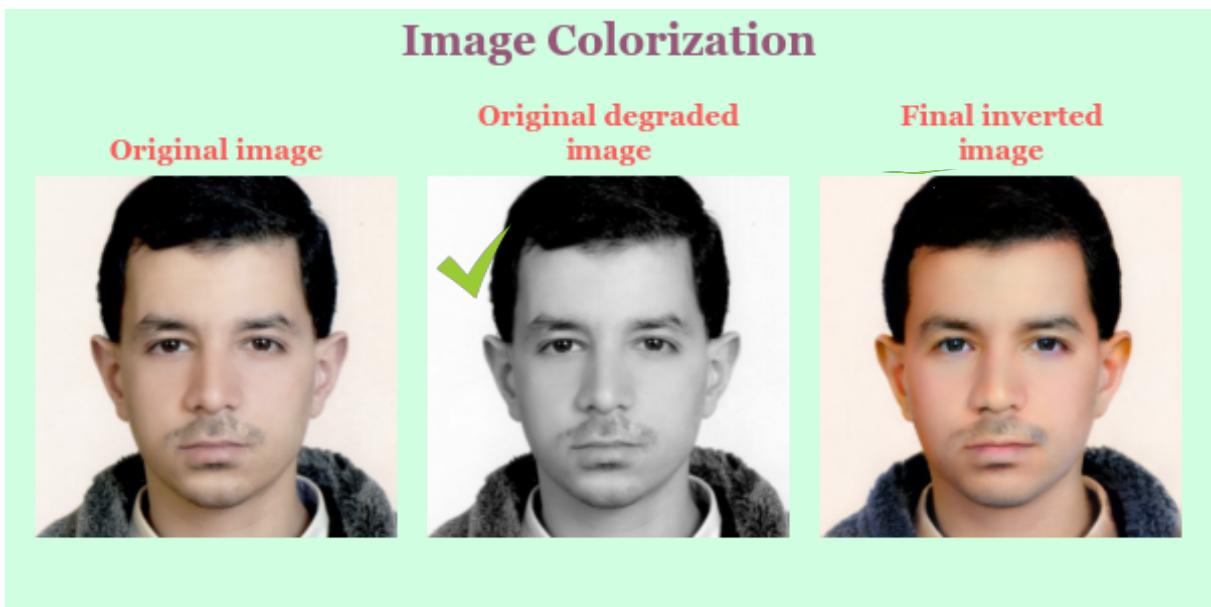
For example, a more blurred image (*kernel_size* = 195) with large *latent_dist_reg_weight* (= 0.2) will give a clear result but deviated relatively from the original one. In the other hand, small *latent_dist_reg_weight* (= 0.01) will give closer result to the original image, but less clear.

Image Colorization:

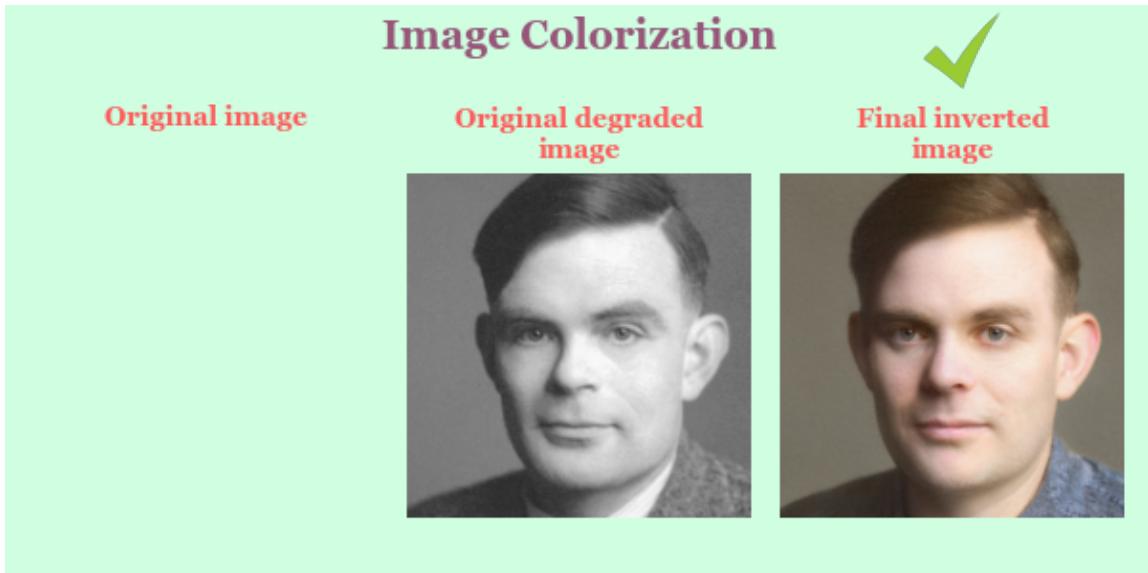
First image



Second image (my personal picture)



Alan Turing



Solution Discussion:

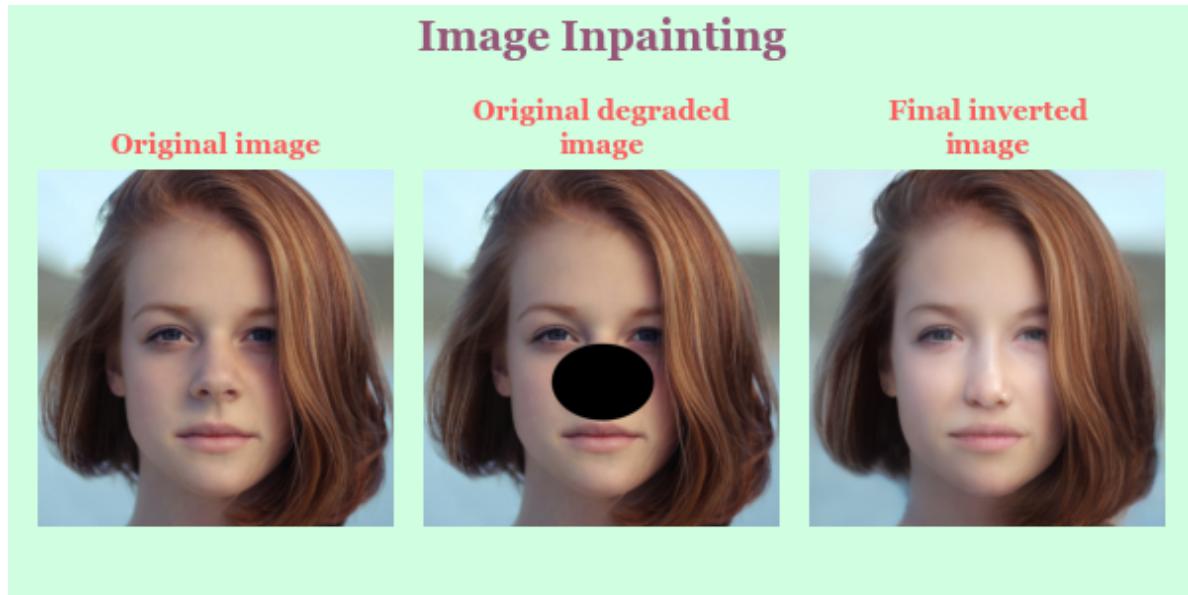
Hyperparameters: `num_steps = 1000, latent_dist_reg_weight = 1`

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.

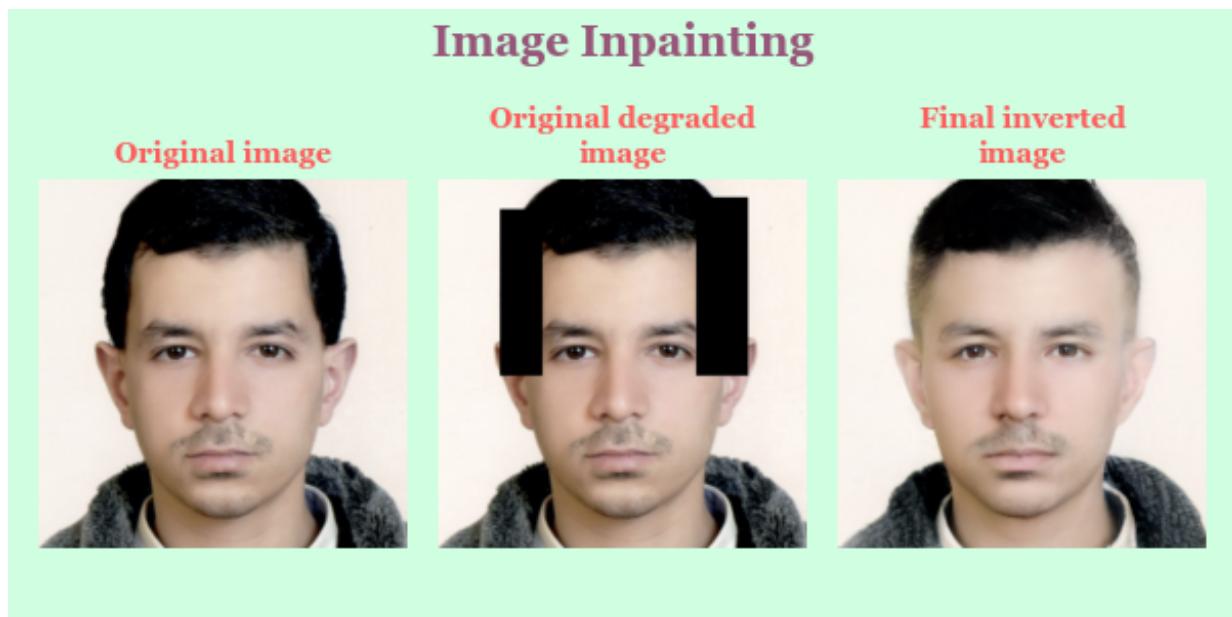
I noticed that a big `latent_dist_reg_weight` gives more realistic colorization, but with a little deviated image from the original one, I think the bigger the `latent_dist_reg_weight` the closer the constructed image to the mean of natural images, and vice versa. Moreover, after at most 1000 steps,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

Image Inpainting:

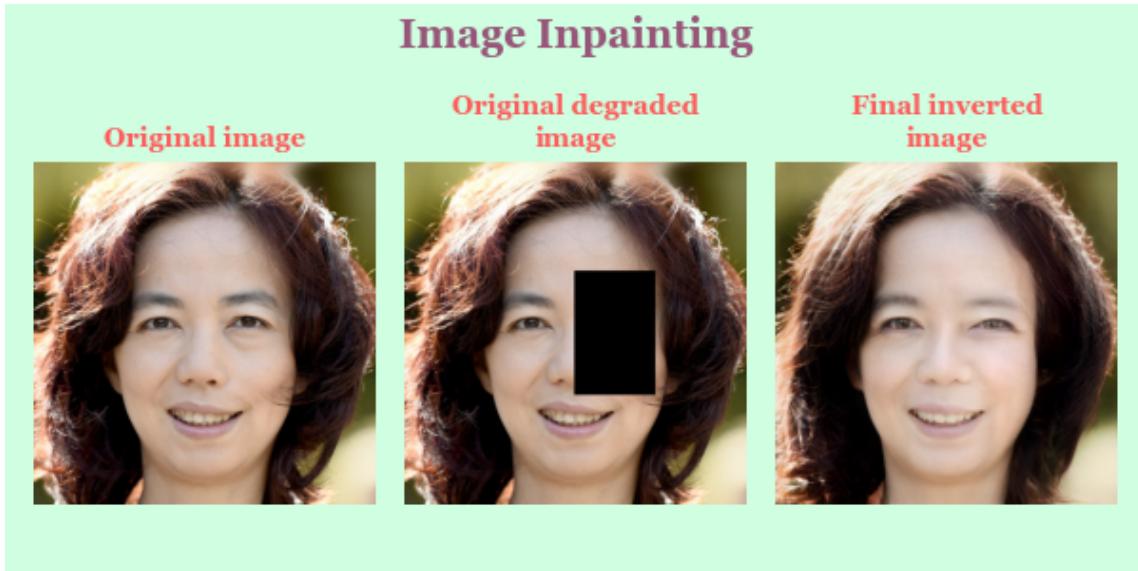
First image



Second image (my personal picture)



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



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

```
1  {
2      "nbformat": 4,
3      "nbformat_minor": 0,
4      "metadata": {
5          "colab": {
6              "name": "Copy_of_IMPR_Ex5_Deep_Style_Image_Prior_2021_2022_.ipynb",
7              "provenance": [],
8              "collapsed_sections": []
9          },
10         "kernelspec": {
11             "name": "python3",
12             "display_name": "Python 3"
13         },
14         "language_info": {
15             "name": "python"
16         },
17         "accelerator": "GPU"
18     },
19     "cells": [
20         {
21             "cell_type": "markdown",
22             "source": [
23                 "# Important! \n",
24                 "When initially opening the notebook there should be a text to the right of the \"Help\" menu saying \"Changes will\n",
25                 "! [WhatsApp Image 2021-01-02 at 22.02.20.jpeg] (data:image/jpeg;base64,/9j/4AAQSkZJRgABAQAAAQABAAAD/2wBDAAMCAgICAgMCAg\n",
26                 "\n",
27                 "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",
28                 "\n",
29                 "**Failing to do so will result in code loss!**\n",
30                 "\n",
31                 " **Note** Make sure you are the only one that has access to it!\n"
32             ],
33             "metadata": {
34                 "id": "7PbmNXuh7WuC"
35             }
36         },
37         {
38             "cell_type": "code",
39             "source": [
40                 "#@markdown #Image Processing - 67829. { display-mode: \"form\" }\n",
41                 "#@markdown ##Exercise 5: Deep Style Image Prior\n",
42                 "#@markdown ##Due date: 13.01.2022 at 23:59\n",
43                 "#@title{ display-mode: \"form\" }\n",
44                 "\n",
45                 "#@markdown\n",
46                 "#@markdown This exercise is a bit different than the rest of the exercises in the course.\n",
47                 "#@markdown The submissions will be a PDF file with your answers and results to the exercise\n",
48                 "#@markdown as well as some files so that we can verify the authenticity of your results.\n",
49                 "#@markdown This notebook provides the basic code, but you do not need to adhere to some specific API\n",
50                 "#@markdown and we will not be running unit tests on your code.\n",
51                 "#@markdown We will however, be going over your code and running it manually.\n",
52                 "#@markdown Moreover, we will be running tests to ensure the authenticity of your solution and detect plagiarism\n",
53                 "#@markdown\n",
54                 "#@markdown\n",
55                 "#@markdown Before you start working on the exercise it is recommended that you review the lecture slides covering\n",
56                 "#@markdown\n"
```

```

57 "#@markdown \n",
58 "#@markdown **NOTE**: Neural networks are typically trained on GPUs, without GPUs training takes much longer. \n",
59 "#@markdown To enable GPU tranining click on \"Runtime\" -> \"Change runtime type\" -> \"GPU\" -> \"SAVE\".\n",
60 "#@markdown\n",
61 "#@markdown **NOTE**: A short guide on debugging your code using colab is available [here](https://colab.research.google.com/notebooks/intro.ipynb#scrollTo=VJzWzXGgDwI).
62 "\n",
63 "#@markdown But first, we have to download all of the dependencies and install them.\n",
64 "#@markdown Play this cell to download it and get everything ready. markdown This may take a few minutes.\n",
65 "\n",
66 "\n",
67 "!mkdir impr_ex5_resources\n",
68 "%cd impr_ex5_resources\n",
69 "#!wget \"https://www.cs.huji.ac.il/~impr/shape_predictor_68_face_landmarks.dat\" -O shape_predictor_68_face_landmarks.dat
70 "#!wget \"https://www.cs.huji.ac.il/~impr/align_faces.py\" -O align_faces.py \n",
71 "#!wget \"https://www.cs.huji.ac.il/~impr/stylegan2-ada-pytorch.tar\" -O stylegan2-ada-pytorch.tar\n",
72 "#!tar -xvf stylegan2-ada-pytorch.tar\n",
73 "#!rm -f stylegan2-ada-pytorch.tar\n",
74 "\n",
75 "import sys\n",
76 "ROOT_PATH=\"/content/impr_ex5_resources/stylegan2-ada-pytorch\"\n",
77 "sys.path.append(ROOT_PATH)\n",
78 "\n",
79 "\n",
80 "!pip install ninja\n",
81 "!pip install mediapy\n",
82 "CHECKPOINTS_PATH = \"https://nvlabs-fi-cdn.nvidia.com/stylegan2-ada/pretrained/ffhq.pkl\"\n",
83 "\n",
84 "\n",
85 "\n",
86 "import copy\n",
87 "import os\n",
88 "from time import perf_counter\n",
89 "import click\n",
90 "import imageio\n",
91 "import numpy as np\n",
92 "import PIL.Image\n",
93 "import torch\n",
94 "import torch.nn.functional as F\n",
95 "import torchvision.transforms as T\n",
96 "\n",
97 "import dnnlib\n",
98 "import legacy\n",
99 "import numpy as np\n",
100 "from skimage.draw import line\n",
101 "from torch.nn.functional import conv2d, conv1d\n",
102 "from torchvision.transforms import Grayscale\n",
103 "import mediapy as media\n",
104 "from IPython.display import clear_output\n"
105 ],
106 "metadata": {
107     "id": "Jean9HbQ7iq-",
108     "colab": {
109         "base_uri": "https://localhost:8080/"
110     },
111     "outputId": "9b9a6d7d-db5a-425c-b06c-67db9f45c974"
112 },
113 "execution_count": 3,
114 "outputs": [
115     {
116         "output_type": "stream",
117         "name": "stdout",
118         "text": [
119             "/content/impr_ex5_resources\n",
120             "--2022-01-17 11:01:04-- https://www.cs.huji.ac.il/~impr/stylegan2-ada-pytorch.tar\n",
121             "Resolving www.cs.huji.ac.il (www.cs.huji.ac.il)... 132.65.118.16\n",
122             "Connecting to www.cs.huji.ac.il (www.cs.huji.ac.il)|132.65.118.16|:443... connected.\n",
123             "HTTP request sent, awaiting response... 302 Moved Temporarily\n",
124             "Location: https://www.cs.huji.ac.il/~impr/stylegan2-ada-pytorch.tar [following]\n",

```

```

125
126     "--2022-01-17 11:01:05-- https://www.cs.huji.ac.il/w-impr/stylegan2-ada-pytorch.tar\n",
127     "Reusing existing connection to www.cs.huji.ac.il:443.\n",
128     "HTTP request sent, awaiting response... 200 OK\n",
129     "Length: 100364800 (96M) [application/x-tar]\n",
130     "Saving to: 'stylegan2-ada-pytorch.tar'\n",
131     "\n",
132     "stylegan2-ada-pytor 100%[=====] 95.71M 16.6MB/s in 6.7s \n",
133     "\n",
134     "2022-01-17 11:01:12 (14.4 MB/s) - 'stylegan2-ada-pytorch.tar' saved [100364800/100364800]\n",
135     "\n",
136     "stylegan2-ada-pytorch/\n",
137     "stylegan2-ada-pytorch/metrics/\n",
138     "stylegan2-ada-pytorch/.dataset_tool.py\n",
139     "tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
140     "stylegan2-ada-pytorch/dataset_tool.py\n",
141     "stylegan2-ada-pytorch/.DS_Store\n",
142     "tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.FinderInfo'\n",
143     "stylegan2-ada-pytorch/.DS_Store\n",
144     "stylegan2-ada-pytorch/.style_mixing.py\n",
145     "tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
146     "stylegan2-ada-pytorch/style_mixing.py\n",
147     "stylegan2-ada-pytorch/torch_utils/\n",
148     "stylegan2-ada-pytorch/.legacy.py\n",
149     "tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
150     "stylegan2-ada-pytorch/legacy.py\n",
151     "stylegan2-ada-pytorch/.generate.py\n",
152     "tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
153     "stylegan2-ada-pytorch/generate.py\n",
154     "stylegan2-ada-pytorch/training/\n",
155     "stylegan2-ada-pytorch/.README.md\n",
156     "tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
157     "stylegan2-ada-pytorch/README.md\n",
158     "stylegan2-ada-pytorch/.projector.py\n",
159     "tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
160     "stylegan2-ada-pytorch/.train.py\n",
161     "tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
162     "stylegan2-ada-pytorch/train.py\n",
163     "stylegan2-ada-pytorch/.calc_metrics.py\n",
164     "tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
165     "stylegan2-ada-pytorch/calc_metrics.py\n",
166     "stylegan2-ada-pytorch/dnnlib/\n",
167     "stylegan2-ada-pytorch/align_faces/\n",
168     "stylegan2-ada-pytorch/.LICENSE.txt\n",
169     "tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
170     "stylegan2-ada-pytorch/LICENSE.txt\n",
171     "stylegan2-ada-pytorch/align_faces/.shape_predictor_68_face_landmarks.dat\n",
172     "tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
173     "stylegan2-ada-pytorch/align_faces/shape_predictor_68_face_landmarks.dat\n",
174     "stylegan2-ada-pytorch/align_faces/.align_faces.py\n",
175     "tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
176     "stylegan2-ada-pytorch/align_faces/align_faces.py\n",
177     "stylegan2-ada-pytorch/dnnlib/.util.py\n",
178     "tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
179     "stylegan2-ada-pytorch/dnnlib/util.py\n",
180     "stylegan2-ada-pytorch/dnnlib/.__init__.py\n",
181     "tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
182     "stylegan2-ada-pytorch/dnnlib/__init__.py\n",
183     "stylegan2-ada-pytorch/dnnlib/__pycache_/\n",
184     "stylegan2-ada-pytorch/dnnlib/__pycache_/.util.cpython-37.pyc\n",
185     "tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
186     "stylegan2-ada-pytorch/dnnlib/__pycache_/_util.cpython-37.pyc\n",
187     "stylegan2-ada-pytorch/dnnlib/__pycache_/_/.__init__.cpython-37.pyc\n",
188     "tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
189     "stylegan2-ada-pytorch/dnnlib/__pycache_/_/__init__.cpython-37.pyc\n",
190     "stylegan2-ada-pytorch/training/.__init__.py\n",
191     "tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
192     "stylegan2-ada-pytorch/training/__init__.py\n",

```

```

193 "stylegan2-ada-pytorch/training/__pycache__/\n",
194 "stylegan2-ada-pytorch/training/_augment.py\n",
195 "tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
196 "stylegan2-ada-pytorch/training/augment.py\n",
197 "stylegan2-ada-pytorch/training/_training_loop.py\n",
198 "tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
199 "stylegan2-ada-pytorch/training/training_loop.py\n",
200 "stylegan2-ada-pytorch/training/_dataset.py\n",
201 "tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
202 "stylegan2-ada-pytorch/training/dataset.py\n",
203 "stylegan2-ada-pytorch/training/_networks.py\n",
204 "tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
205 "stylegan2-ada-pytorch/training/networks.py\n",
206 "stylegan2-ada-pytorch/training/_loss.py\n",
207 "tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
208 "stylegan2-ada-pytorch/training/loss.py\n",
209 "stylegan2-ada-pytorch/training/__pycache__/_networks.cpython-37.pyc\n",
210 "tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
211 "stylegan2-ada-pytorch/training/__pycache__/_networks.cpython-37.pyc\n",
212 "stylegan2-ada-pytorch/training/__pycache__/_init__.cpython-37.pyc\n",
213 "tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
214 "stylegan2-ada-pytorch/training/__pycache__/_init_.cpython-37.pyc\n",
215 "stylegan2-ada-pytorch/torch_utils/_misc.py\n",
216 "tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
217 "stylegan2-ada-pytorch/torch_utils/misc.py\n",
218 "stylegan2-ada-pytorch/torch_utils/_persistence.py\n",
219 "tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
220 "stylegan2-ada-pytorch/torch_utils/persistence.py\n",
221 "stylegan2-ada-pytorch/torch_utils/_custom_ops.py\n",
222 "tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
223 "stylegan2-ada-pytorch/torch_utils/custom_ops.py\n",
224 "stylegan2-ada-pytorch/torch_utils/_init_.py\n",
225 "tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
226 "stylegan2-ada-pytorch/torch_utils/_init_.py\n",
227 "stylegan2-ada-pytorch/torch_utils/__pycache__/\n",
228 "stylegan2-ada-pytorch/torch_utils/_training_stats.py\n",
229 "tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
230 "stylegan2-ada-pytorch/torch_utils/training_stats.py\n",
231 "stylegan2-ada-pytorch/torch_utils/ops/\n",
232 "stylegan2-ada-pytorch/torch_utils/ops/_bias_act.cpp\n",
233 "tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
234 "stylegan2-ada-pytorch/torch_utils/ops/bias_act.cpp\n",
235 "stylegan2-ada-pytorch/torch_utils/ops/_upfirdn2d.cpp\n",
236 "tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
237 "stylegan2-ada-pytorch/torch_utils/ops/upfirdn2d.cpp\n",
238 "stylegan2-ada-pytorch/torch_utils/ops/_grid_sample_gradfix.py\n",
239 "tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
240 "stylegan2-ada-pytorch/torch_utils/ops/grid_sample_gradfix.py\n",
241 "stylegan2-ada-pytorch/torch_utils/ops/_bias_act.py\n",
242 "tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
243 "stylegan2-ada-pytorch/torch_utils/ops/bias_act.py\n",
244 "stylegan2-ada-pytorch/torch_utils/ops/_upfirdn2d.cu\n",
245 "tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
246 "stylegan2-ada-pytorch/torch_utils/ops/upfirdn2d.cu\n",
247 "stylegan2-ada-pytorch/torch_utils/ops/_bias_act.cu\n",
248 "tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
249 "stylegan2-ada-pytorch/torch_utils/ops/bias_act.cu\n",
250 "stylegan2-ada-pytorch/torch_utils/ops/_init_.py\n",
251 "tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
252 "stylegan2-ada-pytorch/torch_utils/ops/_init_.py\n",
253 "stylegan2-ada-pytorch/torch_utils/ops/_upfirdn2d.py\n",
254 "tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
255 "stylegan2-ada-pytorch/torch_utils/ops/upfirdn2d.py\n",
256 "stylegan2-ada-pytorch/torch_utils/ops/__pycache__/\n",
257 "stylegan2-ada-pytorch/torch_utils/ops/_bias_act.h\n",
258 "tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
259 "stylegan2-ada-pytorch/torch_utils/ops/bias_act.h\n",
260 "stylegan2-ada-pytorch/torch_utils/ops/_fma.py\n",

```

```

261 "tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
262 "stylegan2-ada-pytorch/torch_utils/ops/fma.py\n",
263 "stylegan2-ada-pytorch/torch_utils/ops/_conv2d_resample.py\n",
264 "tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
265 "stylegan2-ada-pytorch/torch_utils/ops/conv2d_resample.py\n",
266 "stylegan2-ada-pytorch/torch_utils/ops/_upfirdn2d.h\n",
267 "tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
268 "stylegan2-ada-pytorch/torch_utils/ops/upfirdn2d.h\n",
269 "stylegan2-ada-pytorch/torch_utils/ops/_conv2d_gradfix.py\n",
270 "tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
271 "stylegan2-ada-pytorch/torch_utils/ops/conv2d_gradfix.py\n",
272 "stylegan2-ada-pytorch/torch_utils/ops/_pycache_/_fma.cpython-37.pyc\n",
273 "tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
274 "stylegan2-ada-pytorch/torch_utils/ops/_pycache_/_fma.cpython-37.pyc\n",
275 "stylegan2-ada-pytorch/torch_utils/ops/_pycache_/_conv2d_gradfix.cpython-37.pyc\n",
276 "tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
277 "stylegan2-ada-pytorch/torch_utils/ops/_pycache_/_conv2d_gradfix.cpython-37.pyc\n",
278 "stylegan2-ada-pytorch/torch_utils/ops/_pycache_/_bias_act.cpython-37.pyc\n",
279 "tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
280 "stylegan2-ada-pytorch/torch_utils/ops/_pycache_/_bias_act.cpython-37.pyc\n",
281 "stylegan2-ada-pytorch/torch_utils/ops/_pycache_/_upfirdn2d.cpython-37.pyc\n",
282 "tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
283 "stylegan2-ada-pytorch/torch_utils/ops/_pycache_/_upfirdn2d.cpython-37.pyc\n",
284 "stylegan2-ada-pytorch/torch_utils/ops/_pycache_/_conv2d_resample.cpython-37.pyc\n",
285 "tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
286 "stylegan2-ada-pytorch/torch_utils/ops/_pycache_/_conv2d_resample.cpython-37.pyc\n",
287 "stylegan2-ada-pytorch/torch_utils/ops/_pycache_/_init_.cpython-37.pyc\n",
288 "tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
289 "stylegan2-ada-pytorch/torch_utils/ops/_pycache_/_init_.cpython-37.pyc\n",
290 "stylegan2-ada-pytorch/torch_utils/_pycache_/_persistance.cpython-37.pyc\n",
291 "tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
292 "stylegan2-ada-pytorch/torch_utils/_pycache_/_persistance.cpython-37.pyc\n",
293 "stylegan2-ada-pytorch/torch_utils/_pycache_/_custom_ops.cpython-37.pyc\n",
294 "tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
295 "stylegan2-ada-pytorch/torch_utils/_pycache_/_custom_ops.cpython-37.pyc\n",
296 "stylegan2-ada-pytorch/torch_utils/_pycache_/_init_.cpython-37.pyc\n",
297 "tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
298 "stylegan2-ada-pytorch/torch_utils/_pycache_/_init_.cpython-37.pyc\n",
299 "stylegan2-ada-pytorch/torch_utils/_pycache_/_misc.cpython-37.pyc\n",
300 "tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
301 "stylegan2-ada-pytorch/torch_utils/_pycache_/_misc.cpython-37.pyc\n",
302 "stylegan2-ada-pytorch/metrics/_precision_recall.py\n",
303 "tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
304 "stylegan2-ada-pytorch/metrics/precision_recall.py\n",
305 "stylegan2-ada-pytorch/metrics/_frechet_inception_distance.py\n",
306 "tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
307 "stylegan2-ada-pytorch/metrics/frechet_inception_distance.py\n",
308 "stylegan2-ada-pytorch/metrics/_init_.py\n",
309 "tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
310 "stylegan2-ada-pytorch/metrics/_init_.py\n",
311 "stylegan2-ada-pytorch/metrics/_metric_utils.py\n",
312 "tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
313 "stylegan2-ada-pytorch/metrics/metric_utils.py\n",
314 "stylegan2-ada-pytorch/metrics/_metric_main.py\n",
315 "tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
316 "stylegan2-ada-pytorch/metrics/metric_main.py\n",
317 "stylegan2-ada-pytorch/metrics/_perceptual_path_length.py\n",
318 "tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
319 "stylegan2-ada-pytorch/metrics/perceptual_path_length.py\n",
320 "stylegan2-ada-pytorch/metrics/_inception_score.py\n",
321 "tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
322 "stylegan2-ada-pytorch/metrics/inception_score.py\n",
323 "stylegan2-ada-pytorch/metrics/_kernel_inception_distance.py\n",
324 "tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'\n",
325 "stylegan2-ada-pytorch/metrics/kernel_inception_distance.py\n",
326 "Collecting ninja\n",
327 "  Downloading ninja-1.10.2.3-py2.py3-none-manylinux_2_5_x86_64.manylinux1_x86_64.whl (108 kB)\n",
328 "\u0001b[K    | 108 kB 5.4 MB/s \n",

```

```

329     "\u001b[?25hInstalling collected packages: ninja\n",
330     "Successfully installed ninja-1.10.2.3\n",
331     "Collecting mediapy\n",
332     "  Downloading mediapy-1.0.3-py3-none-any.whl (24 kB)\n",
333     "Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages (from mediapy) (1.19.5)\n",
334     "Requirement already satisfied: Pillow in /usr/local/lib/python3.7/dist-packages (from mediapy) (7.1.2)\n",
335     "Requirement already satisfied: matplotlib in /usr/local/lib/python3.7/dist-packages (from mediapy) (3.2.2)\n",
336     "Requirement already satisfied: ipython in /usr/local/lib/python3.7/dist-packages (from mediapy) (5.5.0)\n",
337     "Requirement already satisfied: setuptools>=18.5 in /usr/local/lib/python3.7/dist-packages (from ipython->mediapy)
338     "Requirement already satisfied: pygments in /usr/local/lib/python3.7/dist-packages (from ipython->mediapy) (2.6
339     "Requirement already satisfied: pickleshare in /usr/local/lib/python3.7/dist-packages (from ipython->mediapy) (0
340     "Requirement already satisfied: decorator in /usr/local/lib/python3.7/dist-packages (from ipython->mediapy) (4.4
341     "Requirement already satisfied: prompt-toolkit<2.0.0,>=1.0.4 in /usr/local/lib/python3.7/dist-packages (from ipy
342     "Requirement already satisfied: simplegeneric>0.8 in /usr/local/lib/python3.7/dist-packages (from ipython->media
343     "Requirement already satisfied: traitlets>=4.2 in /usr/local/lib/python3.7/dist-packages (from ipython->mediapy)
344     "Requirement already satisfied: pexpect in /usr/local/lib/python3.7/dist-packages (from ipython->mediapy) (4.8.0
345     "Requirement already satisfied: wcwidth in /usr/local/lib/python3.7/dist-packages (from prompt-toolkit<2.0.0,>=1
346     "Requirement already satisfied: six>=1.9.0 in /usr/local/lib/python3.7/dist-packages (from prompt-toolkit<2.0.0,
347     "Requirement already satisfied: pyparsing!=2.0.4,!!=2.1.2,!!=2.1.6,>=2.0.1 in /usr/local/lib/python3.7/dist-packag
348     "Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.7/dist-packages (from matplotlib->mediap
349     "Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib->me
350     "Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib->me
351     "Requirement already satisfied: ptyprocess>=0.5 in /usr/local/lib/python3.7/dist-packages (from pexpect->ipython
352     "Installing collected packages: mediapy\n",
353     "Successfully installed mediapy-1.0.3\n"
354   ],
355 ]
356 ],
357 },
358 {
359   "cell_type": "markdown",
360   "source": [
361     "# Mounting Google Drive"
362   ],
363   "metadata": {
364     "id": "LY3S0M7bKggZ"
365   },
366 },
367 {
368   "cell_type": "code",
369   "source": [
370     "#@markdown **NOTE**: It is strongly advised you save your results to Google \n",
371     "#@markdown Drive as they will be deleted from Colab once it restarts. \n",
372     "#@markdown To connect Google Drive run this cell. \n",
373     "from google.colab import drive\n",
374     "drive.mount('/content/gdrive/')"
375   ],
376   "metadata": {
377     "id": "E8d10xYdKhyi",
378     "colab": {
379       "base_uri": "https://localhost:8080/"
380     },
381     "outputId": "64ac6ddb-bcd9-4b15-99e9-a6caca52e8bf"
382   },
383   "execution_count": 4,
384   "outputs": [
385     {
386       "output_type": "stream",
387       "name": "stdout",
388       "text": [
389         "Mounted at /content/gdrive/\n"
390       ]
391     }
392   ],
393 },
394 {
395   "cell_type": "markdown",
396   "source": [

```

```

397     "Below is the root dir of your Google Drive. To choose the destination of the dir to save and read from, create it i
398     ],
399     "metadata": {
400       "id": "Lp4YuqH2Lmku"
401     }
402   },
403   {
404     "cell_type": "code",
405     "source": [
406       "ROOT_GDRIVE_PATH=\\\"/content/gdrive/MyDrive/\\\"\\n",
407       "GDRIVE_SAVE_REL_PATH = \\\"IMPRO_EX5_2021\\\"\\n",
408       "FULL_GDRIVE_SAVE_PATH = ROOT_GDRIVE_PATH + GDRIVE_SAVE_REL_PATH"
409     ],
410     "metadata": {
411       "id": "LAjRCCKLLBI5"
412     },
413     "execution_count": 5,
414     "outputs": []
415   },
416   {
417     "cell_type": "markdown",
418     "source": [
419       "# General Variables"
420     ],
421     "metadata": {
422       "id": "mNEXRTie-E93"
423     }
424   },
425   {
426     "cell_type": "code",
427     "source": [
428       "GAUSSIAN_BLUR_DEGRADATION= 'GAUSSIAN_BLUR_DEGRADATION'\\n",
429       "GRAYSCALE_DEGRADATION = 'GRAYSCALE_DEGRADATION'\\n",
430       "INPAINTING_DEGRADATION = 'INPAINTING_DEGRADATION'\\n",
431       "DENOISING_DEGRADATION = 'DENOISING_DEGRADATION'\\n",
432       "NO_DEGRADATION= 'NO_DEGRADATION'\\n"
433     ],
434     "metadata": {
435       "id": "RFt1dKc095Qi"
436     },
437     "execution_count": 6,
438     "outputs": []
439   },
440   {
441     "cell_type": "markdown",
442     "source": [
443       "# Image Alignment"
444     ],
445     "metadata": {
446       "id": "N1MqcLzq9lHv"
447     }
448   },
449   {
450     "cell_type": "code",
451     "metadata": {
452       "id": "c0204E5oXn-5",
453       "colab": {
454         "base_uri": "https://localhost:8080/"
455       },
456       "outputId": "0716c153-8f73-4662-8887-b578e63c26f6"
457     },
458     "source": [
459       "# The align_faces.py script takes in an input image path, an output image path, and a dat file path. The dat file i
460       "# It is advised that you save the files to google drive as restarting Colab will erase them.\\n",
461       "!python \\\"$ROOT_PATH/align_faces/align_faces.py\\\" '/content/gdrive/MyDrive/IMPRO_EX5_2021/input_images/my_face.png"
462     ],
463     "execution_count": null,
464     "outputs": [

```

```

465     {
466         "output_type": "stream",
467         "name": "stdout",
468         "text": [
469             "Number of faces detected: 1\n",
470             "Detection 0: Left: 617 Top: 937 Right: 1575 Bottom: 1895\n",
471             "Part 0: (614, 1205), Part 1: (624, 1335) ...\n"
472         ]
473     }
474   ],
475 },
476 {
477     "cell_type": "markdown",
478     "source": [
479         "# Degradation Functions "
480     ],
481     "metadata": {
482         "id": "Z7G07LxPC19b"
483     }
484 },
485 {
486     "cell_type": "code",
487     "source": [
488         "    # ****\n489         # ***** NEED TO ADD DEGRADATION FUNCTIONS *****\n490         # ****\n491     \"\n492     # I. GAUSSIAN_BLUR_DEGRADATION\n493     \"\n494     def gaussian_kernel(filter_size):\n495         basis = filter_vec = torch.tensor([0.5, 0.5], dtype=torch.float32).reshape(1, 1, 2)\n496         for i in range(filter_size - 2):\n497             filter_vec = conv1d(filter_vec, basis, padding=1)\n498         return filter_vec\n499         # return filter_vec / torch.sum(filter_vec),\n500     \"\n501     \"\n502     def blur_spatial(im, filter_vec):\n503         filter_vec = filter_vec.repeat(3, 1, 1, 1)\n504         padding = filter_vec.shape[3]//2\n505         blurred_im = conv2d(im, filter_vec, padding=[0,padding], groups=3)\n506         blurred_im = conv2d(blurred_im, filter_vec.permute(0, 1, 3, 2), padding=[padding,0], groups=3)\n507     \"\n508         return blurred_im\n509     \"\n510     # II. GRayscale_Degradation\n511     \"\n512     # III. Inpainting_Degradation"
513 ],
514     "metadata": {
515         "id": "PY7gxd63zqnK"
516     },
517     "execution_count": 7,
518     "outputs": []
519 },
520 {
521     "cell_type": "markdown",
522     "source": [
523         "# GAN Inversion"
524     ],
525     "metadata": {
526         "id": "j2EbeYsb9u7A"
527     }
528 },
529 {
530     "cell_type": "code",
531     "source": [
532         "def run_latent_optimization(outdir,\n
```

```

533     " degradation_mode,\n",
534     " G,\n",
535     " imgs_to_disply_dict,\n",
536     " target: torch.Tensor, # [C,H,W] and dynamic range [0,255], W & H must match G output resolution\n",
537     " *,\n",
538     " num_steps = 1000,\n",
539     " w_avg_samples = 10000,\n",
540     " initial_learning_rate = 0.1,\n",
541     " initial_noise_factor = 0.05,\n",
542     " lr_rampdown_length = 0.25,\n",
543     " lr_rampup_length = 0.05,\n",
544     " noise_ramp_length = 0.75,\n",
545     " regularize_noise_weight = 1e5,\n",
546     " latent_dist_reg_weight = 0.001,\n",
547     " blur_kernel = None,\n",
548     " mask = None,\n",
549     " device: torch.device,\n",
550     "\n",
551   "):\n",
552     " assert target.shape == (G.img_channels, G.img_resolution, G.img_resolution)\n",
553   "\n",
554     " if degradation_mode == GAUSSIAN_BLUR_DEGRADATION:\n",
555       " assert blur_kernel is not None\n",
556     " if degradation_mode == INPAINTING_DEGRADATION:\n",
557       " assert mask is not None\n",
558   "\n",
559     " G = copy.deepcopy(G).eval().requires_grad_(False).to(device) # type: ignore\n",
560   "\n",
561     " # Compute w stats.\n",
562     " print(f'Computing W midpoint and stddev using {w_avg_samples} samples...')\n",
563     " z_samples = np.random.RandomState(123).randn(w_avg_samples, G.z_dim)\n",
564     " w_samples = G.mapping(torch.from_numpy(z_samples).to(device), None) # [N, L, C]\n",
565     " w_samples = w_samples.cpu().numpy().astype(np.float32)\n",
566     " w_avg = np.mean(w_samples, axis=0, keepdims=True) # [1, 18, C]\n",
567     " w_avg_original = torch.from_numpy(w_avg).to(device).float()\n",
568     " w_std = (np.sum((w_samples - w_avg) ** 2) / w_avg_samples) ** 0.5\n",
569   "\n",
570     " # Setup noise inputs.\n",
571     " noise_bufs = { name: buf for (name, buf) in G.synthesis.named_buffers() if 'noise_const' in name }\n",
572   "\n",
573     " # Load VGG16 feature detector.\n",
574     " url = 'https://nvlabs-fi-cdn.nvidia.com/stylegan2-ada-pytorch/pretrained/metrics/vgg16.pt'\n",
575     " with dnnlib.util.open_url(url) as f:\n",
576       vgg16 = torch.jit.load(f).eval().to(device)\n",
577   "\n",
578     " # Features for target image.\n",
579     " target_images = target.unsqueeze(0).to(device).to(torch.float32)\n",
580   "\n",
581     " if target_images.shape[2] > 256:\n",
582       target_images = F.interpolate(target_images, size=(256, 256), mode='area')\n",
583     target_features = vgg16(target_images, resize_images=False, return_lpips=True)\n",
584   "\n",
585     w_opt = torch.tensor(w_avg, dtype=torch.float32, device=device, requires_grad=True)\n",
586     w_out = torch.zeros([num_steps] + list(w_opt.shape[1:])), dtype=torch.float32, device=device)\n",
587     loss_out = torch.zeros([num_steps], dtype=torch.float32, device=device)\n",
588     optimizer = torch.optim.Adam([w_opt] + list(noise_bufs.values()), betas=(0.9, 0.999), lr=initial_learning_rate)\n",
589   "\n",
590     " # Init noise.\n",
591     " for buf in noise_bufs.values():\n",
592       buf[:] = torch.randn_like(buf)\n",
593       buf.requires_grad = True\n",
594   "\n",
595     for step in range(num_steps):\n",
596       " # Learning rate schedule.\n",
597       t = step / num_steps\n",
598       w_noise_scale = w_std * initial_noise_factor * max(0.0, 1.0 - t / noise_ramp_length) ** 2\n",
599       lr_ramp = min(1.0, (1.0 - t) / lr_rampdown_length)\n",
600       lr_ramp = 0.5 - 0.5 * np.cos(lr_ramp * np.pi)\n",

```

```

601
602     "lr_ramp = lr_ramp * min(1.0, t / lr_rampup_length)\n",
603     "lr = initial_learning_rate * lr_ramp\n",
604     "for param_group in optimizer.param_groups:\n",
605         "param_group['lr'] = lr\n",
606     "\n",
607     "# Synth image from opt_w\n",
608     "w_noise = torch.randn_like(w_opt) * w_noise_scale\n",
609     "ws = w_opt + w_noise\n",
610     "synth_images = G.synthesis(ws, noise_mode='const')\n",
611     "\n",
612     "# Prep to save synth image\n",
613     "synth_image_save = (synth_images + 1) * (255/2)\n",
614     "synth_image_save = synth_image_save.permute(0, 2, 3, 1).clamp(0, 255).to(torch.uint8)[0].cpu().numpy()\n",
615     "\n",
616     "# ****\n617     "# ***** NEED TO FILL IN THE FOLLOWING CODE *****\n618     "# ****\n619     "if degradation_mode == INPAINTING_DEGRADATION:\n",
620         "assert mask.shape == synth_images.shape\n",
621         "synth_images *= mask\n",
622     "elif degradation_mode == GRAYSCALE_DEGRADATION:\n",
623         "synth_images = Grayscale(num_output_channels=3)(synth_images)\n",
624     "elif degradation_mode == GAUSSIAN_BLUR_DEGRADATION:\n",
625         "synth_images = blur_spatial(synth_images, blur_kernel)\n",
626     "\n",
627     "# ****\n628     "# ***** END CODE TO ADD SECTION *****\n629     "# ****\n630     "\n",
631     "\n",
632     "# Prep to save and show images\n",
633     "synth_image_degraded_save = (synth_images + 1) * (255/2)\n",
634     "synth_image_degraded_save = synth_image_degraded_save.permute(0, 2, 3, 1).clamp(0, 255).to(torch.uint8)[0]\n",
635     "\n",
636     "if step % 20 == 0:\n",
637         "imgs_to_disply_dict["Generated Image"] = synth_image_save\n",
638         "imgs_to_disply_dict["Generated Degraded Image"] = synth_image_degraded_save\n",
639         "clear_output(wait=True)\n",
640         "media.show_images(imgs_to_disply_dict, height=256)\n",
641     if step % 100 == 0:\n",
642         "PIL.Image.fromarray(synth_image_save, 'RGB').save(f'{outdir}/intermidiate_{step}_not_degraded.png')\n",
643         "PIL.Image.fromarray(synth_image_degraded_save, 'RGB').save(f'{outdir}/intermidiate_{step}_degraded.png')\n",
644     "\n",
645     "\n",
646     "# Noise regularization.\n",
647     "reg_loss = 0.0\n",
648     "for v in noise_bufs.values():\n",
649         "noise = v[None, None, :, :] # must be [1,1,H,W] for F.avg_pool2d()\n",
650         "while True:\n",
651             "reg_loss += (noise * torch.roll(noise, shifts=1, dims=3)).mean()**2\n",
652             "reg_loss += (noise * torch.roll(noise, shifts=1, dims=2)).mean()**2\n",
653             "if noise.shape[2] <= 8:\n",
654                 "break\n",
655             "noise = F.avg_pool2d(noise, kernel_size=2)\n",
656     "\n",
657     "# Downsample image to 256x256 if it's larger than that. VGG was built for 224x224 images.\n",
658     "synth_images = (synth_images + 1) * (255/2)\n",
659     "if synth_images.shape[2] > 256:\n",
660         "synth_images = F.interpolate(synth_images, size=(256, 256), mode='area')\n",
661     "\n",
662     "# Features for synth images.\n",
663     "synth_features = vgg16(synth_images, resize_images=False, return_lpips=True)\n",
664     "\n",
665     "# Compute loss\n",
666     "percep_loss = (target_features - synth_features).square().sum()\n",
667     "latent_dist_reg = F.l1_loss(w_avg_original, w_opt)\n",
668     "loss = percep_loss + reg_loss * regularize_noise_weight + latent_dist_reg_weight * latent_dist_reg\n",

```

```

669     "\n",
670     "\n",
671     " # Step\n",
672     " optimizer.zero_grad(set_to_none=True)\n",
673     " loss.backward()\n",
674     " optimizer.step()\n",
675     "\n",
676     print(f'step {step+1:>4d}/{num_steps}: percep_loss {percep_loss:<4.2f} latent_dist_reg {latent_dist_reg:<4.2f}\n',
677     "\n",
678     " # Save inverted latent for each optimization step.\n",
679     " w_out[step] = w_opt.detach()[0]\n",
680     " loss_out[step] = loss\n",
681     "\n",
682     " # Normalize noise.\n",
683     " with torch.no_grad():\n",
684     "     for buf in noise_bufs.values():\n",
685     "         buf -= buf.mean()\n",
686     "         buf *= buf.square().mean().rsqrt()\n",
687     "\n",
688     "     return w_out, loss_out"
689 ],
690 "metadata": {
691   "id": "daYITHcfi9PP"
692 },
693 "execution_count": 14,
694 "outputs": []
695 },
696 {
697   "cell_type": "code",
698   "metadata": {
699     "id": "mdFYp5sBor2U"
700   },
701   "source": [
702     "def invert_image(degradation_mode,\n",
703     "                  target_fname,\n",
704     "                  outdir,\n",
705     "                  mask_fname = None,\n",
706     "                  seed=303,\n",
707     "                  num_steps=1000,\n",
708     "                  latent_dist_reg_weight=0.001):\n",
709     "    np.random.seed(seed)\n",
710     "    torch.manual_seed(seed)\n",
711     "\n",
712     "    # Load networks.\n",
713     "    print('Loading networks from \'%s' % CHECKPOINTS_PATH)\n",
714     "    device = torch.device('cuda')\n",
715     "    with dnnlib.util.open_url(CHECKPOINTS_PATH) as fp:\n",
716     "        networks = legacy.load_network_pkl(fp)\n",
717     "        G = networks['G_ema'].requires_grad_(False).to(device)\n",
718     "        \n",
719     "\n",
720     "    # Load target image.\n",
721     "    if not os.path.exists(outdir):\n",
722     "        os.makedirs(outdir)\n",
723     "    target_pil = PIL.Image.open(target_fname).convert('RGB')\n",
724     "    w, h = target_pil.size\n",
725     "    s = min(w, h)\n",
726     "    target_pil = target_pil.crop(((w - s) // 2, (h - s) // 2, (w + s) // 2, (h + s) // 2))\n",
727     "    target_pil = target_pil.resize((G.img_resolution, G.img_resolution), PIL.Image.LANCZOS)\n",
728     "    target_uint8 = np.array(target_pil, dtype=np.uint8)\n",
729     "    target = torch.tensor(target_uint8.transpose([2, 0, 1]), device=device), \n",
730     "    target_images = target[0].unsqueeze(0).to(device).to(torch.float32)\n",
731     "\n",
732     "    # *****\n",
733     "    # ***** NEED TO FILL IN THE FOLLOWING CODE *****\n",
734     "    # *****\n",
735     "    kernel = None\n",
736     "    mask = None\n",

```

```

737     " if degradation_mode == INPAINTING_DEGRADATION:\n",
738     "     mask = PIL.Image.open(mask_fname).convert('RGB')\n",
739     "     mask = torch.tensor(T.ToTensor()(mask), dtype=torch.float32, device=device).unsqueeze(0)\n",
740     "     target_images *= mask\n",
741     " elif degradation_mode == GRAYSCALE_DEGRADATION:\n",
742     "     target_images = Grayscale(num_output_channels=3)(target_images)\n",
743     " elif degradation_mode == GAUSSIAN_BLUR_DEGRADATION:\n",
744     "     kernel = gaussian_kernel(85).to(device)\n",
745     "     target_images = blur_spatial(target_images, kernel)\n",
746     "     pass\n",
747     " \n",
748     "# ****\n749     # ***** END CODE TO ADD SECTION *****\n750     # ****\n751 \n",
752     "#Save target image\n",
753     target_to_save = target_images.permute(0, 2, 3, 1).clamp(0, 255).to(torch.uint8)[0].cpu().numpy()\n",
754     PIL.Image.fromarray(target_to_save, 'RGB').save(f'{outdir}/original_degraded_image.png')\n",
755     imgs_to_disply_dict = {\n",
756         '\"Original Image\"':target_uint8,\n",
757         '\"Original Degraded Image\"':target_to_save,\n",
758         }\n",
759     "\n",
760     "# Run latent optimization\n",
761     print('Run latent optimization')\n",
762     start_time = perf_counter()\n",
763     optimization_steps, optimization_losses = run_latent_optimization(\n",
764         outdir,\n",
765         degradation_mode,\n",
766         G,\n",
767         imgs_to_disply_dict,\n",
768         target_images[0],\n",
769         num_steps=num_steps,\n",
770         device=device,\n",
771         latent_dist_reg_weight=latent_dist_reg_weight,\n",
772         blur_kernel=kernel,\n",
773         mask=mask\n",
774     )\n",
775     "\n",
776     print (f'Elapsed: {(perf_counter()-start_time):.1f} s')\n",
777     os.makedirs(outdir, exist_ok=True)\n",
778     "\n",
779     # Save final inverted image and latent vector.\n",
780     inverted_latent = optimization_steps[-1]\n",
781     synth_image = G.synthesis(inverted_latent.unsqueeze(0), noise_mode='const')\n",
782     synth_image = (synth_image + 1) * (255/2)\n",
783     synth_image = synth_image.permute(0, 2, 3, 1).clamp(0, 255).to(torch.uint8)[0].cpu().numpy()\n",
784     PIL.Image.fromarray(synth_image, 'RGB').save(f'{outdir}/final_inverted_image.png')\n",
785     np.savez(f'{outdir}/inverted_latent.npz', latent=inverted_latent.unsqueeze(0).cpu().numpy())\n",
786     "\n",
787     " return optimization_losses"
788 ],
789 "execution_count": 20,
790 "outputs": []
791 },
792 {
793     "cell_type": "markdown",
794     "source": [
795         "# GAN Inversion with no degradation"
796     ],
797     "metadata": {
798         "id": "UL3mDbZOGJ2c"
799     }
800 },
801 {
802     "cell_type": "markdown",
803     "source": [
804         "**The optimization progressions with over 5 images**"

```

```

805     ],
806     "metadata": {
807       "id": "wcXL2UYrGp7B"
808     }
809   },
810   {
811     "cell_type": "code",
812     "source": [
813       "opt_losses = invert_image(degradation_mode=NO_DEGRADATION,\n",
814       "                           target_fname='/content/gdrive/MyDrive/image_const/Christopher_Campbell_aligned.png',\n",
815       "                           outdir='/content/gdrive/MyDrive/image_const',\n",
816       "                           seed=303,\n",
817       "                           num_steps=800,\n",
818       "                           latent_dist_reg_weight=0.1)"
819     ],
820     "metadata": {
821       "colab": {
822         "base_uri": "https://localhost:8080/",
823         "height": 661
824       },
825       "id": "CA4iQPyhG8eK",
826       "outputId": "bbdf9f40d-714c-4fdb-ae80-0c9e2f3688b6"
827     },
828     "execution_count": 18,
829     "outputs": [
830       {
831         "output_type": "display_data",
832         "data": {
833           "text/html": [
834             "<table class=\"show_images\" style=\"border-spacing:0px;\"><tr><td style=\"padding:1px;\"><div style=\"display:flex; flex-direction:column; align-items:center;\">\n",
835             "               <div style=\"display:flex; flex-direction:column; align-items:center;\">\n",
836             "                 <div>Original Image</div><div><img width=\"256\" height=\"256\" style=\"image-rendering:pixelated; object-fit:contain;\" alt=\"Original Image\"/>\n",
837             "               <div style=\"display:flex; flex-direction:column; align-items:center;\">\n",
838             "                 <div>Original Degraded Image</div><div><img width=\"256\" height=\"256\" style=\"image-rendering:pixelated; object-fit:contain;\" alt=\"Original Degraded Image\"/>\n",
839             "               <div style=\"display:flex; flex-direction:column; align-items:center;\">\n",
840             "                 <div>Generated Image</div><div><img width=\"256\" height=\"256\" style=\"image-rendering:pixelated; object-fit:contain;\" alt=\"Generated Image\"/>\n",
841             "               <div style=\"display:flex; flex-direction:column; align-items:center;\">\n",
842             "                 <div>Generated Degraded Image</div><div><img width=\"256\" height=\"256\" style=\"image-rendering:pixelated; object-fit:contain;\" alt=\"Generated Degraded Image\"/>\n",
843             "             </td></tr></table>\n",
844             "text/plain": [
845               "<IPython.core.display.HTML object>"
846             ]
847           },
848           "metadata": {}
849         },
850         {
851           "output_type": "stream",
852           "name": "stdout",
853           "text": [
854             "step 781/800: percep_loss 0.10 latent_dist_reg 0.31 loss 0.14 \n",
855             "step 782/800: percep_loss 0.10 latent_dist_reg 0.31 loss 0.14 \n",
856             "step 783/800: percep_loss 0.10 latent_dist_reg 0.31 loss 0.14 \n",
857             "step 784/800: percep_loss 0.10 latent_dist_reg 0.31 loss 0.14 \n",
858             "step 785/800: percep_loss 0.10 latent_dist_reg 0.31 loss 0.14 \n",
859             "step 786/800: percep_loss 0.10 latent_dist_reg 0.31 loss 0.14 \n",
860             "step 787/800: percep_loss 0.10 latent_dist_reg 0.31 loss 0.14 \n",
861             "step 788/800: percep_loss 0.10 latent_dist_reg 0.31 loss 0.14 \n",
862             "step 789/800: percep_loss 0.10 latent_dist_reg 0.31 loss 0.14 \n",
863             "step 790/800: percep_loss 0.10 latent_dist_reg 0.31 loss 0.14 \n",
864             "step 791/800: percep_loss 0.10 latent_dist_reg 0.31 loss 0.14 \n",
865             "step 792/800: percep_loss 0.10 latent_dist_reg 0.31 loss 0.14 \n",
866             "step 793/800: percep_loss 0.10 latent_dist_reg 0.31 loss 0.14 \n",
867             "step 794/800: percep_loss 0.10 latent_dist_reg 0.31 loss 0.14 \n",
868             "step 795/800: percep_loss 0.10 latent_dist_reg 0.31 loss 0.14 \n",
869             "step 796/800: percep_loss 0.10 latent_dist_reg 0.31 loss 0.14 \n",
870             "step 797/800: percep_loss 0.10 latent_dist_reg 0.31 loss 0.14 \n",
871             "step 798/800: percep_loss 0.10 latent_dist_reg 0.31 loss 0.14 \n",
872             "step 799/800: percep_loss 0.10 latent_dist_reg 0.31 loss 0.14 \n",

```

```

873         "step 800/800: percep_loss 0.10 latent_dist_reg 0.31 loss 0.14 \n",
874         "Elapsed: 540.5 s\n"
875     ]
876   ]
877 },
878 {
879   "cell_type": "markdown",
880   "source": [
881     "**A plot of the optimization loss**"
882   ],
883   "metadata": {
884     "id": "BCev1fKdc9U0"
885   }
886 },
887 {
888   "cell_type": "code",
889   "source": [
890     "import matplotlib.pyplot as plt\n",
891     "\n",
892     "def plot_optimization_loss(loss):\n",
893       steps = np.arange(len(loss))\n",
894       plt.plot(steps, loss.cpu().detach().numpy(), 'b-')\n",
895       plt.xlabel('step')\n",
896       plt.xticks(np.arange(0, len(loss) + 1, 100), rotation = (45), fontsize = 10)\n",
897       plt.ylabel('loss')\n",
898       plt.yscale('log')\n",
899       plt.title('The Optimization Loss')\n",
900       plt.tight_layout()\n",
901       plt.show()\n",
902       # plt.savefig('/content/gdrive/MyDrive/IMPRO_EX5_2021/output_images/LossPlot.png')\n"
903   ],
904   "metadata": {
905     "id": "SUU2Tdfuc8VK"
906   },
907   "execution_count": 10,
908   "outputs": []
909 },
910 {
911   "cell_type": "code",
912   "source": [
913     "plot_optimization_loss(opt_losses)"
914   ],
915   "metadata": {
916     "id": "OVTT4H5P-jpb",
917     "colab": {
918       "base_uri": "https://localhost:8080/",
919       "height": 297
920     },
921     "outputId": "4a40cbb7-7815-412a-fd1d-6a2dd42841ac"
922   },
923   "execution_count": 19,
924   "outputs": [
925     {
926       "output_type": "display_data",
927       "data": {
928         "image/png": "iVBORw0KGgoAAAANSUhEUgAAAagAAAEYCAYAAAJeGK1AAAABHNCSVQICAgIfAhkiAAAAAlwSF1zAAALEgAACxIB0t1+/AAAAAD",
929         "text/plain": [
930           "<Figure size 432x288 with 1 Axes>"
931         ]
932       },
933       "metadata": {
934         "needs_background": "light"
935       }
936     }
937   ],
938 },
939 {

```

```

941     "cell_type": "markdown",
942     "source": [
943         "# Image Reconstruction Tasks"
944     ],
945     "metadata": {
946         "id": "uuNq08zWq7f9"
947     },
948 },
949 {
950     "cell_type": "markdown",
951     "source": [
952         "**Image Deblurring**"
953     ],
954     "metadata": {
955         "id": "2DvmVh0jrEEC"
956     },
957 },
958 {
959     "cell_type": "code",
960     "source": [
961         "\n",
962         "opt_losses = invert_image(degradation_mode=GAUSSIAN_BLUR_DEGRADATION,\n",
963         "                           target_fname='/content/gdrive/MyDrive/deblurring/im1/Christopher_Campbell_aligned.png',\n",
964         "                           outdir='/content/gdrive/MyDrive/deblurring/im1',\n",
965         "                           seed=303,\n",
966         "                           num_steps=1000,\n",
967         "                           latent_dist_reg_weight=0.1)"
968     ],
969     "metadata": {
970         "colab": {
971             "base_uri": "https://localhost:8080/",
972             "height": 661
973         },
974         "id": "1lbGRs8prIFd",
975         "outputId": "654c64c8-c66e-43b2-a7ed-034dc8cad524"
976     },
977     "execution_count": 15,
978     "outputs": [
979         {
980             "output_type": "display_data",
981             "data": {
982                 "text/html": [
983                     "<table class=\"show_images\" style=\"border-spacing:0px;\"><tr><td style=\"padding:1px;\"><div style=\"display:flex; flex-direction:column; align-items:center;\">\n",
984                         "<div style=\"display:flex; flex-direction:column; align-items:center;\">\n",
985                             "<div>Original Image</div><div><img width=\"256\" height=\"256\" style=\"image-rendering:pixelated; object-fit: contain;\" alt=\"Original Image\"/>\n",
986                             "<div style=\"display:flex; flex-direction:column; align-items:center;\">\n",
987                                 "<div>Original Degraded Image</div><div><img width=\"256\" height=\"256\" style=\"image-rendering:pixelated; object-fit: contain;\" alt=\"Original Degraded Image\"/>\n",
988                                 "<div style=\"display:flex; flex-direction:column; align-items:center;\">\n",
989                                     "<div>Generated Image</div><div><img width=\"256\" height=\"256\" style=\"image-rendering:pixelated; object-fit: contain;\" alt=\"Generated Image\"/>\n",
990                                     "<div style=\"display:flex; flex-direction:column; align-items:center;\">\n",
991                                         "<div>Generated Degraded Image</div><div><img width=\"256\" height=\"256\" style=\"image-rendering:pixelated; object-fit: contain;\" alt=\"Generated Degraded Image\"/>\n",
992                                         "</div>\n",
993                         "</div>\n",
994                         "<IPython.core.display.HTML object>\n",
995                     ]
996                 },
997                 "metadata": {}
998             },
999             {
1000                 "output_type": "stream",
1001                 "name": "stdout",
1002                 "text": [
1003                     "step 981/1000: percep_loss 0.04 latent_dist_reg 0.25 loss 0.06 \n",
1004                     "step 982/1000: percep_loss 0.04 latent_dist_reg 0.25 loss 0.06 \n",
1005                     "step 983/1000: percep_loss 0.04 latent_dist_reg 0.25 loss 0.06 \n",
1006                     "step 984/1000: percep_loss 0.04 latent_dist_reg 0.25 loss 0.06 \n",
1007                     "step 985/1000: percep_loss 0.04 latent_dist_reg 0.25 loss 0.06 \n",
1008                     "step 986/1000: percep_loss 0.04 latent_dist_reg 0.25 loss 0.06 \n",

```

```

1009     "step 987/1000: percep_loss 0.04 latent_dist_reg 0.25 loss 0.06 \n",
1010     "step 988/1000: percep_loss 0.04 latent_dist_reg 0.25 loss 0.06 \n",
1011     "step 989/1000: percep_loss 0.04 latent_dist_reg 0.25 loss 0.06 \n",
1012     "step 990/1000: percep_loss 0.04 latent_dist_reg 0.25 loss 0.06 \n",
1013     "step 991/1000: percep_loss 0.04 latent_dist_reg 0.25 loss 0.06 \n",
1014     "step 992/1000: percep_loss 0.04 latent_dist_reg 0.25 loss 0.06 \n",
1015     "step 993/1000: percep_loss 0.04 latent_dist_reg 0.25 loss 0.06 \n",
1016     "step 994/1000: percep_loss 0.04 latent_dist_reg 0.25 loss 0.06 \n",
1017     "step 995/1000: percep_loss 0.04 latent_dist_reg 0.25 loss 0.06 \n",
1018     "step 996/1000: percep_loss 0.04 latent_dist_reg 0.25 loss 0.06 \n",
1019     "step 997/1000: percep_loss 0.04 latent_dist_reg 0.25 loss 0.06 \n",
1020     "step 998/1000: percep_loss 0.04 latent_dist_reg 0.25 loss 0.06 \n",
1021     "step 999/1000: percep_loss 0.04 latent_dist_reg 0.25 loss 0.06 \n",
1022     "step 1000/1000: percep_loss 0.04 latent_dist_reg 0.25 loss 0.06 \n",
1023     "Elapsed: 345.4 s\n"
1024   ]
1025 }
1026 ]
1027 },
1028 {
1029   "cell_type": "code",
1030   "source": [
1031     "plot_optimization_loss(opt_losses)"
1032   ],
1033   "metadata": {
1034     "id": "dQ4YubC9esWY",
1035     "colab": {
1036       "base_uri": "https://localhost:8080/",
1037       "height": 297
1038     },
1039     "outputId": "435f2163-b21a-4e9f-defd-f68e67f85a1f"
1040   },
1041   "execution_count": 12,
1042   "outputs": [
1043     {
1044       "output_type": "display_data",
1045       "data": {
1046         "image/png": "iVBORw0KGgoAAAANSUhEUgAAAagAAAEYCAYAAAJeGK1AAAABHNCVQICAgIfAhkiAAAAAlwSF1zAAALEgAACxIB0t1+/AAAAAD",
1047         "text/plain": [
1048           "<Figure size 432x288 with 1 Axes>"
1049         ]
1050       },
1051       "metadata": {
1052         "needs_background": "light"
1053       }
1054     }
1055   ],
1056 },
1057 {
1058   "cell_type": "code",
1059   "source": [
1060     "opt_losses = invert_image(degradation_mode=GAUSSIAN_BLUR_DEGRADATION,\n",
1061     "                           target_fname='/content/gdrive/MyDrive/deblurring/im2/my_face_aligned.png',\n",
1062     "                           outdir='/content/gdrive/MyDrive/deblurring/im2',\n",
1063     "                           seed=303,\n",
1064     "                           num_steps=1000,\n",
1065     "                           latent_dist_reg_weight=0.1)"
1066   ],
1067   "metadata": {
1068     "id": "bdeXXRLlesQ1",
1069     "colab": {
1070       "base_uri": "https://localhost:8080/",
1071       "height": 653
1072     },
1073     "outputId": "d16ff7d3-5fa5-4a5d-c50a-4f70db05a0b5"
1074   },
1075   "execution_count": null,
1076   "outputs": [

```

```

1077 {
1078     "output_type": "display_data",
1079     "data": {
1080         "text/html": [
1081             "<table class=\"show_images\" style=\"border-spacing:0px;\"><tr><td style=\"padding:1px;\"><div style=\"display:flex; flex-direction:column; align-items:center;\">\n",
1082                 "<div style=\"display:flex; flex-direction:column; align-items:center;\">\n",
1083                     "<div>Original Image</div><div><img width=\"256\" height=\"256\" style=\"image-rendering:pixelated; object-fit: contain;\" alt=\"Original Image\"/>\n",
1084                     "<div>Original Degraded Image</div><div><img width=\"256\" height=\"256\" style=\"image-rendering:pixelated; object-fit: contain;\" alt=\"Original Degraded Image\"/>\n",
1085                     "<div>Generated Image</div><div><img width=\"256\" height=\"256\" style=\"image-rendering:pixelated; object-fit: contain;\" alt=\"Generated Image\"/>\n",
1086                     "<div>Generated Degraded Image</div><div><img width=\"256\" height=\"256\" style=\"image-rendering:pixelated; object-fit: contain;\" alt=\"Generated Degraded Image\"/>\n",
1087             "</div>\n",
1088         ],
1089         "text/plain": [
1090             "<IPython.core.display.HTML object>"
1091         ]
1092     },
1093 },
1094 },
1095 "metadata": {}
1096 },
1097 {
1098     "output_type": "stream",
1099     "name": "stdout",
1100     "text": [
1101         "step 981/1000: percep_loss 0.04 latent_dist_reg 0.22 loss 0.06 \n",
1102         "step 982/1000: percep_loss 0.04 latent_dist_reg 0.22 loss 0.06 \n",
1103         "step 983/1000: percep_loss 0.04 latent_dist_reg 0.22 loss 0.06 \n",
1104         "step 984/1000: percep_loss 0.04 latent_dist_reg 0.22 loss 0.06 \n",
1105         "step 985/1000: percep_loss 0.04 latent_dist_reg 0.22 loss 0.06 \n",
1106         "step 986/1000: percep_loss 0.04 latent_dist_reg 0.22 loss 0.06 \n",
1107         "step 987/1000: percep_loss 0.04 latent_dist_reg 0.22 loss 0.06 \n",
1108         "step 988/1000: percep_loss 0.04 latent_dist_reg 0.22 loss 0.06 \n",
1109         "step 989/1000: percep_loss 0.04 latent_dist_reg 0.22 loss 0.06 \n",
1110         "step 990/1000: percep_loss 0.04 latent_dist_reg 0.22 loss 0.06 \n",
1111         "step 991/1000: percep_loss 0.04 latent_dist_reg 0.22 loss 0.06 \n",
1112         "step 992/1000: percep_loss 0.04 latent_dist_reg 0.22 loss 0.06 \n",
1113         "step 993/1000: percep_loss 0.04 latent_dist_reg 0.22 loss 0.06 \n",
1114         "step 994/1000: percep_loss 0.04 latent_dist_reg 0.22 loss 0.06 \n",
1115         "step 995/1000: percep_loss 0.04 latent_dist_reg 0.22 loss 0.06 \n",
1116         "step 996/1000: percep_loss 0.04 latent_dist_reg 0.22 loss 0.06 \n",
1117         "step 997/1000: percep_loss 0.04 latent_dist_reg 0.22 loss 0.06 \n",
1118         "step 998/1000: percep_loss 0.04 latent_dist_reg 0.22 loss 0.06 \n",
1119         "step 999/1000: percep_loss 0.04 latent_dist_reg 0.22 loss 0.06 \n",
1120         "step 1000/1000: percep_loss 0.04 latent_dist_reg 0.22 loss 0.06 \n",
1121         "Elapsed: 260.9 s\n"
1122     ]
1123 }
1124 ],
1125 },
1126 {
1127     "cell_type": "code",
1128     "source": [
1129         "plot_optimization_loss(opt_losses)"
1130     ],
1131     "metadata": {
1132         "id": "aWtQgNkterlM",
1133         "colab": {
1134             "base_uri": "https://localhost:8080/",
1135             "height": 297
1136         },
1137         "outputId": "6164515a-8531-4564-cd1d-28339d48074d"
1138     },
1139     "execution_count": null,
1140     "outputs": [
1141         {
1142             "output_type": "display_data",
1143             "data": {
1144                 "image/png": "iVBORw0KGgoAAAANSUhEUgAAAAgAAAAEYCAYAAAJeGK1AAABHNCVQICAgIfAhkiAAAAAlwSF1zAAALEgAACxIB0t1+/AAAAAD"
1145             }
1146         }
1147     ]
1148 }

```

```

1145     "text/plain": [
1146         "<Figure size 432x288 with 1 Axes>"
1147     ],
1148 },
1149     "metadata": {
1150         "needs_background": "light"
1151     }
1152 },
1153 ],
1154 },
1155 {
1156     "cell_type": "markdown",
1157     "source": [
1158         "**Deblurring on the given blurry image of Yann LeCun**"
1159     ],
1160     "metadata": {
1161         "id": "_wNVoFb54YMB"
1162     }
1163 },
1164 {
1165     "cell_type": "markdown",
1166     "source": [
1167         "In order to run the next cell, comment the line:\n",
1168         "\n",
1169         "`target_images = blur_spatial(target_images, kernel)`\n",
1170         "\n",
1171         "in the function: `invert_image` under the section:\n",
1172         "\n",
1173         `elif degradation_mode == GAUSSIAN_BLUR_DEGRADATION:`\n",
1174         "\n",
1175         "\n"
1176     ],
1177     "metadata": {
1178         "id": "MphzBo03ayxH"
1179     }
1180 },
1181 {
1182     "cell_type": "code",
1183     "source": [
1184         "opt_losses = invert_image(degradation_mode=GAUSSIAN_BLUR_DEGRADATION,\n",
1185         "                           target_fname='/content/gdrive/MyDrive/deblurring/Yann LeCun/yann_lecun_blur.png',\n",
1186         "                           outdir='/content/gdrive/MyDrive/deblurring/Yann LeCun',\n",
1187         "                           seed=303,\n",
1188         "                           num_steps=2000,\n",
1189         "                           latent_dist_reg_weight=0.005)"
1190     ],
1191     "metadata": {
1192         "colab": {
1193             "base_uri": "https://localhost:8080/",
1194             "height": 661
1195         },
1196         "id": "gNJDp15n4mXg",
1197         "outputId": "26ed2faf-429e-40a1-c360-a9341fcf0943"
1198     },
1199     "execution_count": 15,
1200     "outputs": [
1201         {
1202             "output_type": "display_data",
1203             "data": {
1204                 "text/html": [
1205                     "<table class=\"show_images\" style=\"border-spacing:0px;\"><tr><td style=\"padding:1px;\"><div style=\"display:flex; flex-direction:column; align-items:center;\">\n",
1206                     "        <div style=\"display:flex; flex-direction:column; align-items:center;\">\n",
1207                     "            <div>Original Image</div><div><img width=\"256\" height=\"256\" style=\"image-rendering:pixelated; object-fit:cover;\" alt=\"Original Image\"/>\n",
1208                     "            <div style=\"display:flex; flex-direction:column; align-items:center;\">\n",
1209                     "                <div>Original Degraded Image</div><div><img width=\"256\" height=\"256\" style=\"image-rendering:pixelated; object-fit:cover;\" alt=\"Original Degraded Image\"/>\n",
1210                     "                <div style=\"display:flex; flex-direction:column; align-items:center;\">\n",
1211                     "                    <div>Generated Image</div><div><img width=\"256\" height=\"256\" style=\"image-rendering:pixelated; object-fit:cover;\" alt=\"Generated Image\"/>\n",
1212                     "                    <div style=\"display:flex; flex-direction:column; align-items:center;\">\n"

```

```

1213         "Generated Degraded Image</div><div><img width=\"256\" height=\"256\" style=\"image-rendering:pixelated; object-fit: contain;\" alt=\"Generated Degraded Image\" data-bbox=1213-1214></div>
1214     ],
1215     "text/plain": [
1216         "<IPython.core.display.HTML object>"
1217     ]
1218 },
1219 "metadata": {}
1220 },
1221 {
1222     "output_type": "stream",
1223     "name": "stdout",
1224     "text": [
1225         "step 1981/2000: percep_loss 0.05 latent_dist_reg 1.36 loss 0.06 \n",
1226         "step 1982/2000: percep_loss 0.05 latent_dist_reg 1.36 loss 0.06 \n",
1227         "step 1983/2000: percep_loss 0.05 latent_dist_reg 1.36 loss 0.06 \n",
1228         "step 1984/2000: percep_loss 0.05 latent_dist_reg 1.36 loss 0.06 \n",
1229         "step 1985/2000: percep_loss 0.05 latent_dist_reg 1.36 loss 0.06 \n",
1230         "step 1986/2000: percep_loss 0.05 latent_dist_reg 1.36 loss 0.06 \n",
1231         "step 1987/2000: percep_loss 0.05 latent_dist_reg 1.36 loss 0.06 \n",
1232         "step 1988/2000: percep_loss 0.05 latent_dist_reg 1.36 loss 0.06 \n",
1233         "step 1989/2000: percep_loss 0.05 latent_dist_reg 1.36 loss 0.06 \n",
1234         "step 1990/2000: percep_loss 0.05 latent_dist_reg 1.36 loss 0.06 \n",
1235         "step 1991/2000: percep_loss 0.05 latent_dist_reg 1.36 loss 0.06 \n",
1236         "step 1992/2000: percep_loss 0.05 latent_dist_reg 1.36 loss 0.06 \n",
1237         "step 1993/2000: percep_loss 0.05 latent_dist_reg 1.36 loss 0.06 \n",
1238         "step 1994/2000: percep_loss 0.05 latent_dist_reg 1.36 loss 0.06 \n",
1239         "step 1995/2000: percep_loss 0.05 latent_dist_reg 1.36 loss 0.06 \n",
1240         "step 1996/2000: percep_loss 0.05 latent_dist_reg 1.36 loss 0.06 \n",
1241         "step 1997/2000: percep_loss 0.05 latent_dist_reg 1.36 loss 0.06 \n",
1242         "step 1998/2000: percep_loss 0.05 latent_dist_reg 1.36 loss 0.06 \n",
1243         "step 1999/2000: percep_loss 0.05 latent_dist_reg 1.36 loss 0.06 \n",
1244         "step 2000/2000: percep_loss 0.05 latent_dist_reg 1.36 loss 0.06 \n",
1245         "Elapsed: 1402.8 s\n"
1246     ]
1247 }
1248 ],
1249 },
1250 {
1251     "cell_type": "code",
1252     "source": [
1253         "plot_optimization_loss(opt_losses)"
1254     ],
1255     "metadata": {
1256         "colab": {
1257             "base_uri": "https://localhost:8080/",
1258             "height": 297
1259         },
1260         "id": "EGjBmD39-Y_j",
1261         "outputId": "170d454d-6b11-45c0-cbb4-1d37319339e9"
1262     },
1263     "execution_count": null,
1264     "outputs": [
1265         {
1266             "output_type": "display_data",
1267             "data": {
1268                 "image/png": "iVBORw0KGgoAAAANSUhEUgAAAAgAAAEYCAYAAAJeGK1AAAABHNCVQICAgIfAhkiAAAAAlwSF1zAAALEgAACxIB0t1+/AAAAAD",
1269                 "text/plain": [
1270                     "<Figure size 432x288 with 1 Axes>"
1271                 ]
1272             },
1273             "metadata": {
1274                 "needs_background": "light"
1275             }
1276         }
1277     ],
1278 },
1279 {
1280     "cell_type": "markdown",

```

```

1281     "source": [
1282         "**Image Colorization**"
1283     ],
1284     "metadata": {
1285         "id": "Rzqos7yN-5cf"
1286     }
1287 },
1288 {
1289     "cell_type": "code",
1290     "source": [
1291         "opt_losses = invert_image(degradation_mode=GRAYSCALE_DEGRADATION,\n",
1292         "                                target_fname='/content/gdrive/MyDrive/Colorization/im1/Christopher_Campbell_aligned.png',\n",
1293         "                                outdir='/content/gdrive/MyDrive/Colorization/im1',\n",
1294         "                                seed=613,\n",
1295         "                                num_steps=900,\n",
1296         "                                latent_dist_reg_weight=1)"
1297     ],
1298     "metadata": {
1299         "colab": {
1300             "base_uri": "https://localhost:8080/",
1301             "height": 653
1302         },
1303         "id": "-xf3Slkd--jK",
1304         "outputId": "07ffd2bb-26fe-4325-b5da-37adea66584b"
1305     },
1306     "execution_count": null,
1307     "outputs": [
1308         {
1309             "output_type": "display_data",
1310             "data": {
1311                 "text/html": [
1312                     "<table class=\"show_images\" style=\"border-spacing:0px;\"><tr><td style=\"padding:1px;\"><div style=\"display:flex; flex-direction:column; align-items:center;\">\n",
1313                     "                         <div style=\"display:flex; flex-direction:column; align-items:center;\">\n",
1314                     "                             <div>Original Image</div><div><img width=\"256\" height=\"256\" style=\"image-rendering:pixelated; object-fit: contain;\" alt=\"Original Image\"/>\n",
1315                     "                         <div style=\"display:flex; flex-direction:column; align-items:center;\">\n",
1316                     "                             <div>Original Degraded Image</div><div><img width=\"256\" height=\"256\" style=\"image-rendering:pixelated; object-fit: contain;\" alt=\"Original Degraded Image\"/>\n",
1317                     "                         <div style=\"display:flex; flex-direction:column; align-items:center;\">\n",
1318                     "                             <div>Generated Image</div><div><img width=\"256\" height=\"256\" style=\"image-rendering:pixelated; object-fit: contain;\" alt=\"Generated Image\"/>\n",
1319                     "                         <div style=\"display:flex; flex-direction:column; align-items:center;\">\n",
1320                     "                             <div>Generated Degraded Image</div><div><img width=\"256\" height=\"256\" style=\"image-rendering:pixelated; object-fit: contain;\" alt=\"Generated Degraded Image\"/>\n",
1321                 ],
1322                 "text/plain": [
1323                     "<IPython.core.display.HTML object>"
1324                 ]
1325             },
1326             "metadata": {}
1327         },
1328         {
1329             "output_type": "stream",
1330             "name": "stdout",
1331             "text": [
1332                 "step 881/900: percep_loss 0.15 latent_dist_reg 0.06 loss 0.21 \n",
1333                 "step 882/900: percep_loss 0.15 latent_dist_reg 0.06 loss 0.21 \n",
1334                 "step 883/900: percep_loss 0.15 latent_dist_reg 0.06 loss 0.21 \n",
1335                 "step 884/900: percep_loss 0.15 latent_dist_reg 0.06 loss 0.21 \n",
1336                 "step 885/900: percep_loss 0.15 latent_dist_reg 0.06 loss 0.21 \n",
1337                 "step 886/900: percep_loss 0.15 latent_dist_reg 0.06 loss 0.21 \n",
1338                 "step 887/900: percep_loss 0.15 latent_dist_reg 0.06 loss 0.21 \n",
1339                 "step 888/900: percep_loss 0.15 latent_dist_reg 0.06 loss 0.21 \n",
1340                 "step 889/900: percep_loss 0.15 latent_dist_reg 0.06 loss 0.21 \n",
1341                 "step 890/900: percep_loss 0.15 latent_dist_reg 0.06 loss 0.21 \n",
1342                 "step 891/900: percep_loss 0.15 latent_dist_reg 0.06 loss 0.21 \n",
1343                 "step 892/900: percep_loss 0.15 latent_dist_reg 0.06 loss 0.21 \n",
1344                 "step 893/900: percep_loss 0.15 latent_dist_reg 0.06 loss 0.21 \n",
1345                 "step 894/900: percep_loss 0.15 latent_dist_reg 0.06 loss 0.21 \n",
1346                 "step 895/900: percep_loss 0.15 latent_dist_reg 0.06 loss 0.21 \n",
1347                 "step 896/900: percep_loss 0.15 latent_dist_reg 0.06 loss 0.21 \n",
1348                 "step 897/900: percep_loss 0.15 latent_dist_reg 0.06 loss 0.21 \n",

```

```

1349     "step 898/900: percep_loss 0.15 latent_dist_reg 0.06 loss 0.21 \n",
1350     "step 899/900: percep_loss 0.15 latent_dist_reg 0.06 loss 0.21 \n",
1351     "step 900/900: percep_loss 0.15 latent_dist_reg 0.06 loss 0.21 \n",
1352     "Elapsed: 229.4 s\n"
1353   ]
1354 }
1355 ]
1356 },
1357 {
1358   "cell_type": "code",
1359   "source": [
1360     "plot_optimization_loss(opt_losses)"
1361   ],
1362   "metadata": {
1363     "colab": {
1364       "base_uri": "https://localhost:8080/",
1365       "height": 297
1366     },
1367     "id": "ra0im_ByVdjb",
1368     "outputId": "92561da3-4bfc-4297-8105-5675440609e9"
1369   },
1370   "execution_count": null,
1371   "outputs": [
1372     {
1373       "output_type": "display_data",
1374       "data": {
1375         "image/png": "iVBORw0KGgoAAAANSUhEUgAAAagAAAECAYAAAAJeGK1AAAABHNCVQICAgIfAhkiAAAAAlwSF1zAAALEgAACxIB0t1+/AAAAAD",
1376         "text/plain": [
1377           "<Figure size 432x288 with 1 Axes>"
1378         ]
1379       },
1380       "metadata": {
1381         "needs_background": "light"
1382       }
1383     }
1384   ],
1385 },
1386 {
1387   "cell_type": "code",
1388   "source": [
1389     "opt_losses = invert_image(degradation_mode=GRAYSCALE_DEGRADATION,\n",
1390     "                           target_fname='/content/gdrive/MyDrive/Colorization/im2/my_face_aligned.png',\n",
1391     "                           outdir='/content/gdrive/MyDrive/Colorization/im2',\n",
1392     "                           seed=303,\n",
1393     "                           num_steps=1000,\n",
1394     "                           latent_dist_reg_weight=2)"
1395   ],
1396   "metadata": {
1397     "colab": {
1398       "base_uri": "https://localhost:8080/",
1399       "height": 653
1400     },
1401     "id": "sv2q8h45Vdb9",
1402     "outputId": "c6e6ee4f-f337-4c43-ddcb-edf10b020cab"
1403   },
1404   "execution_count": null,
1405   "outputs": [
1406     {
1407       "output_type": "display_data",
1408       "data": {
1409         "text/html": [
1410           "<table class=\"show_images\" style=\"border-spacing:0px;\"><tr><td style=\"padding:1px;\"><div style=\"display:flex; flex-direction:column; align-items:center;\">\n",
1411             "<div style=\"display:flex; flex-direction:column; align-items:center;\">\n",
1412               "<div>Original Image</div><div><img width=\"256\" height=\"256\" style=\"image-rendering:pixelated; object-fit:cover;\" alt=\"Original Image\"/>\n",
1413             "<div style=\"display:flex; flex-direction:column; align-items:center;\">\n",
1414               "<div>Original Degraded Image</div><div><img width=\"256\" height=\"256\" style=\"image-rendering:pixelated; object-fit:cover;\" alt=\"Original Degraded Image\"/>\n",
1415             "<div style=\"display:flex; flex-direction:column; align-items:center;\">\n",
1416               "<div>Generated Image</div><div><img width=\"256\" height=\"256\" style=\"image-rendering:pixelated; object-fit:cover;\" alt=\"Generated Image\"/>\n"

```

```

1417     "      <div style=\"display:flex; flex-direction:column; align-items:center;\">\n",
1418     "        <div>Generated Degraded Image</div><div><img width=\"256\" height=\"256\" style=\"image-rendering:pixel\n",
1419     ],
1420     "text/plain": [
1421       "<IPython.core.display.HTML object>"
1422     ]
1423   },
1424   "metadata": {}
1425 },
1426 {
1427   "output_type": "stream",
1428   "name": "stdout",
1429   "text": [
1430     "step 981/1000: percep_loss 0.17 latent_dist_reg 0.03 loss 0.22 \n",
1431     "step 982/1000: percep_loss 0.17 latent_dist_reg 0.03 loss 0.22 \n",
1432     "step 983/1000: percep_loss 0.17 latent_dist_reg 0.03 loss 0.22 \n",
1433     "step 984/1000: percep_loss 0.17 latent_dist_reg 0.03 loss 0.22 \n",
1434     "step 985/1000: percep_loss 0.17 latent_dist_reg 0.03 loss 0.22 \n",
1435     "step 986/1000: percep_loss 0.17 latent_dist_reg 0.03 loss 0.22 \n",
1436     "step 987/1000: percep_loss 0.17 latent_dist_reg 0.03 loss 0.22 \n",
1437     "step 988/1000: percep_loss 0.17 latent_dist_reg 0.03 loss 0.22 \n",
1438     "step 989/1000: percep_loss 0.17 latent_dist_reg 0.03 loss 0.22 \n",
1439     "step 990/1000: percep_loss 0.17 latent_dist_reg 0.03 loss 0.22 \n",
1440     "step 991/1000: percep_loss 0.17 latent_dist_reg 0.03 loss 0.22 \n",
1441     "step 992/1000: percep_loss 0.17 latent_dist_reg 0.03 loss 0.22 \n",
1442     "step 993/1000: percep_loss 0.17 latent_dist_reg 0.03 loss 0.22 \n",
1443     "step 994/1000: percep_loss 0.17 latent_dist_reg 0.03 loss 0.22 \n",
1444     "step 995/1000: percep_loss 0.17 latent_dist_reg 0.03 loss 0.22 \n",
1445     "step 996/1000: percep_loss 0.17 latent_dist_reg 0.03 loss 0.22 \n",
1446     "step 997/1000: percep_loss 0.17 latent_dist_reg 0.03 loss 0.22 \n",
1447     "step 998/1000: percep_loss 0.17 latent_dist_reg 0.03 loss 0.22 \n",
1448     "step 999/1000: percep_loss 0.17 latent_dist_reg 0.03 loss 0.22 \n",
1449     "step 1000/1000: percep_loss 0.17 latent_dist_reg 0.03 loss 0.22 \n",
1450     "Elapsed: 244.5 s\n"
1451   ]
1452 }
1453 ],
1454 },
1455 {
1456   "cell_type": "code",
1457   "source": [
1458     "plot_optimization_loss(opt_losses)"
1459   ],
1460   "metadata": {
1461     "colab": {
1462       "base_uri": "https://localhost:8080/",
1463       "height": 297
1464     },
1465     "id": "FChkvoxPVc6F",
1466     "outputId": "738f5b70-aacf-4b73-b509-02b916f56742"
1467   },
1468   "execution_count": null,
1469   "outputs": [
1470     {
1471       "output_type": "display_data",
1472       "data": {
1473         "image/png": "iVBORw0KGgoAAAANSUhEUgAAAagAAAECAYAAAAJeGK1AAAABHNCVQICAgIfAhkiAAAAAlwSF1zAAALEgAACxIB0t1+/AAAAAD",
1474         "text/plain": [
1475           "<Figure size 432x288 with 1 Axes>"
1476         ]
1477       },
1478       "metadata": {
1479         "needs_background": "light"
1480       }
1481     }
1482   ],
1483 },
1484 {

```

```

1485 "cell_type": "markdown",
1486 "source": [
1487     "**The colorization on the given grayscale image of Alan Turing**"
1488 ],
1489 "metadata": {
1490     "id": "EzV-NOQ6--97"
1491 }
1492 },
1493 {
1494     "cell_type": "code",
1495     "source": [
1496         "opt_losses = invert_image(degradation_mode=GRAYSCALE_DEGRADATION,\n",
1497         "                           target_fname='/content/gdrive/MyDrive/IMPRO_EX5_2021/Alan Turing/alan_turing_grayscale.pr",
1498         "                           outdir='/content/gdrive/MyDrive/IMPRO_EX5_2021/Alan Turing',\n",
1499         "                           seed=303,\n",
1500         "                           num_steps=1500,\n",
1501         "                           latent_dist_reg_weight=1)"
1502 ],
1503     "metadata": {
1504         "id": "4IdNubSz_EFG",
1505         "colab": {
1506             "base_uri": "https://localhost:8080/",
1507             "height": 653
1508         },
1509         "outputId": "8d8206d2-3df9-4630-85d1-103f73285f6a"
1510     },
1511     "execution_count": null,
1512     "outputs": [
1513         {
1514             "output_type": "display_data",
1515             "data": {
1516                 "text/html": [
1517                     "<table class=\"show_images\" style=\"border-spacing:0px;\"><tr><td style=\"padding:1px;\"><div style=\"display:flex; flex-direction:column; align-items:center;\">\n",
1518                     "                         <div style=\"display:flex; flex-direction:column; align-items:center;\">\n",
1519                     "                             <div>Original Image</div><div><img width=\"256\" height=\"256\" style=\"image-rendering:pixelated; object-fit: contain;\" alt=\"Original Image\"/>\n",
1520                     "                             <div>Original Degraded Image</div><div><img width=\"256\" height=\"256\" style=\"image-rendering:pixelated; object-fit: contain;\" alt=\"Original Degraded Image\"/>\n",
1521                     "                             <div>Generated Image</div><div><img width=\"256\" height=\"256\" style=\"image-rendering:pixelated; object-fit: contain;\" alt=\"Generated Image\"/>\n",
1522                     "                             <div>Generated Degraded Image</div><div><img width=\"256\" height=\"256\" style=\"image-rendering:pixelated; object-fit: contain;\" alt=\"Generated Degraded Image\"/>\n",
1523                     "                         </div>\n",
1524                     "                     </td>\n",
1525                     "                 </tr></table>\n",
1526                     "<text/plain>:\n",
1527                     "<IPython.core.display.HTML object>\n",
1528                 ]
1529             },
1530             "metadata": {}
1531         },
1532         {
1533             "output_type": "stream",
1534             "name": "stdout",
1535             "text": [
1536                 "step 1481/1500: percep_loss 0.15 latent_dist_reg 0.06 loss 0.20 \n",
1537                 "step 1482/1500: percep_loss 0.15 latent_dist_reg 0.06 loss 0.20 \n",
1538                 "step 1483/1500: percep_loss 0.15 latent_dist_reg 0.06 loss 0.20 \n",
1539                 "step 1484/1500: percep_loss 0.15 latent_dist_reg 0.06 loss 0.20 \n",
1540                 "step 1485/1500: percep_loss 0.15 latent_dist_reg 0.06 loss 0.20 \n",
1541                 "step 1486/1500: percep_loss 0.15 latent_dist_reg 0.06 loss 0.20 \n",
1542                 "step 1487/1500: percep_loss 0.15 latent_dist_reg 0.06 loss 0.20 \n",
1543                 "step 1488/1500: percep_loss 0.15 latent_dist_reg 0.06 loss 0.20 \n",
1544                 "step 1489/1500: percep_loss 0.15 latent_dist_reg 0.06 loss 0.20 \n",
1545                 "step 1490/1500: percep_loss 0.15 latent_dist_reg 0.06 loss 0.20 \n",
1546                 "step 1491/1500: percep_loss 0.15 latent_dist_reg 0.06 loss 0.20 \n",
1547                 "step 1492/1500: percep_loss 0.15 latent_dist_reg 0.06 loss 0.20 \n",
1548                 "step 1493/1500: percep_loss 0.15 latent_dist_reg 0.06 loss 0.20 \n",
1549                 "step 1494/1500: percep_loss 0.15 latent_dist_reg 0.06 loss 0.20 \n",
1550                 "step 1495/1500: percep_loss 0.15 latent_dist_reg 0.06 loss 0.20 \n",
1551                 "step 1496/1500: percep_loss 0.15 latent_dist_reg 0.06 loss 0.20 \n",
1552             ]
1553         }
1554     ]
1555 
```

```

1553     "step 1497/1500: percep_loss 0.15 latent_dist_reg 0.06 loss 0.20 \n",
1554     "step 1498/1500: percep_loss 0.15 latent_dist_reg 0.06 loss 0.20 \n",
1555     "step 1499/1500: percep_loss 0.15 latent_dist_reg 0.06 loss 0.20 \n",
1556     "step 1500/1500: percep_loss 0.15 latent_dist_reg 0.06 loss 0.20 \n",
1557     "Elapsed: 389.2 s\n"
1558   ]
1559 }
1560 ]
1561 },
1562 {
1563   "cell_type": "code",
1564   "source": [
1565     "plot_optimization_loss(opt_losses)"
1566   ],
1567   "metadata": {
1568     "colab": {
1569       "base_uri": "https://localhost:8080/",
1570       "height": 314
1571     },
1572     "id": "LdyRP7tc_Du7",
1573     "outputId": "c5c80024-5eda-4cf0-bd1e-6d67820d7d8c"
1574   },
1575   "execution_count": null,
1576   "outputs": [
1577     {
1578       "output_type": "display_data",
1579       "data": {
1580         "image/png": "iVBORw0KGgoAAAANSUhEUgAAAAgAAAECAYAAQJeGK1AAAABHNCVQICAgIfAhkiAAAAAlwSF1zAAALEgAACxIB0t1+/AAAAA",
1581         "text/plain": [
1582           "<Figure size 432x288 with 1 Axes>"
1583         ]
1584       },
1585       "metadata": {
1586         "needs_background": "light"
1587       }
1588     },
1589     {
1590       "output_type": "display_data",
1591       "data": {
1592         "text/plain": [
1593           "<Figure size 432x288 with 0 Axes>"
1594         ]
1595       },
1596       "metadata": {}
1597     }
1598   ],
1599 },
1600 {
1601   "cell_type": "markdown",
1602   "source": [
1603     "***Image Inpainting***"
1604   ],
1605   "metadata": {
1606     "id": "CSJVTuqpzII_"
1607   },
1608 },
1609 {
1610   "cell_type": "code",
1611   "source": [
1612     "opt_losses = invert_image(degradation_mode=INPAINTING_DEGRADATION,\n",
1613     "                           target_fname='/content/gdrive/MyDrive/inpainting/im1/Christopher_Campbell_aligned.png',\n",
1614     "                           outdir='/content/gdrive/MyDrive/inpainting/im1',\n",
1615     "                           mask_fname='/content/gdrive/MyDrive/inpainting/im1/Christopher_Campbell_mask.png',\n",
1616     "                           seed=303,\n",
1617     "                           num_steps=1000,\n",
1618     "                           latent_dist_reg_weight=0.2)"
1619   ],
1620   "metadata": {

```

```
1621     "colab": {
1622         "base_uri": "https://localhost:8080/",
1623         "height": 653
1624     },
1625     "id": "PYd00jEbNUSQ",
1626     "outputId": "6c279316-3ac9-4ef7-fa56-a3d61663a5e9"
1627 },
1628 "execution_count": null,
1629 "outputs": [
1630     {
1631         "output_type": "display_data",
1632         "data": {
1633             "text/html": [
1634                 "<table class=\"show_images\" style=\"border-spacing:0px;\"><tr><td style=\"padding:1px;\"><div style=\"display:flex; flex-direction:column; align-items:center;\">\n",
1635                 "        <div>Original Image</div><div><img width=\"256\" height=\"256\" style=\"image-rendering:pixelated; object-fit: contain;\" alt=\"Original Image\"/></div>\n",
1636             ],
1637             "text/plain": [
1638                 "*****\n",
1639                 "*** THIS FILE WAS TOO BIG AND WAS TRUNCATED ***\n",
1640                 "*****\n"
1641             ]
1642         }
1643     }
1644 ]
1645 }
```


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



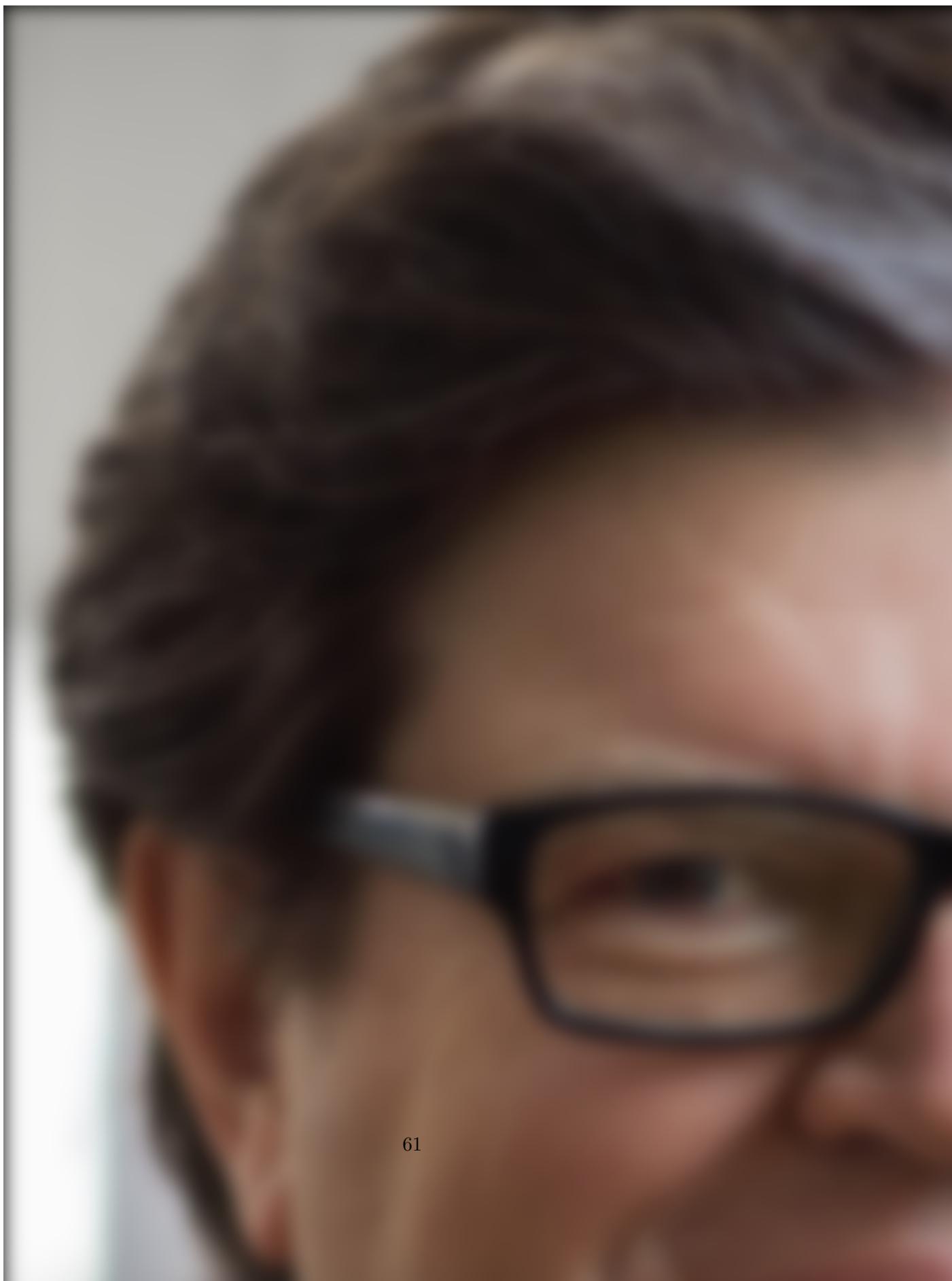
14 ex5/Colorization/im2 s900 w1/original
degraded image.png



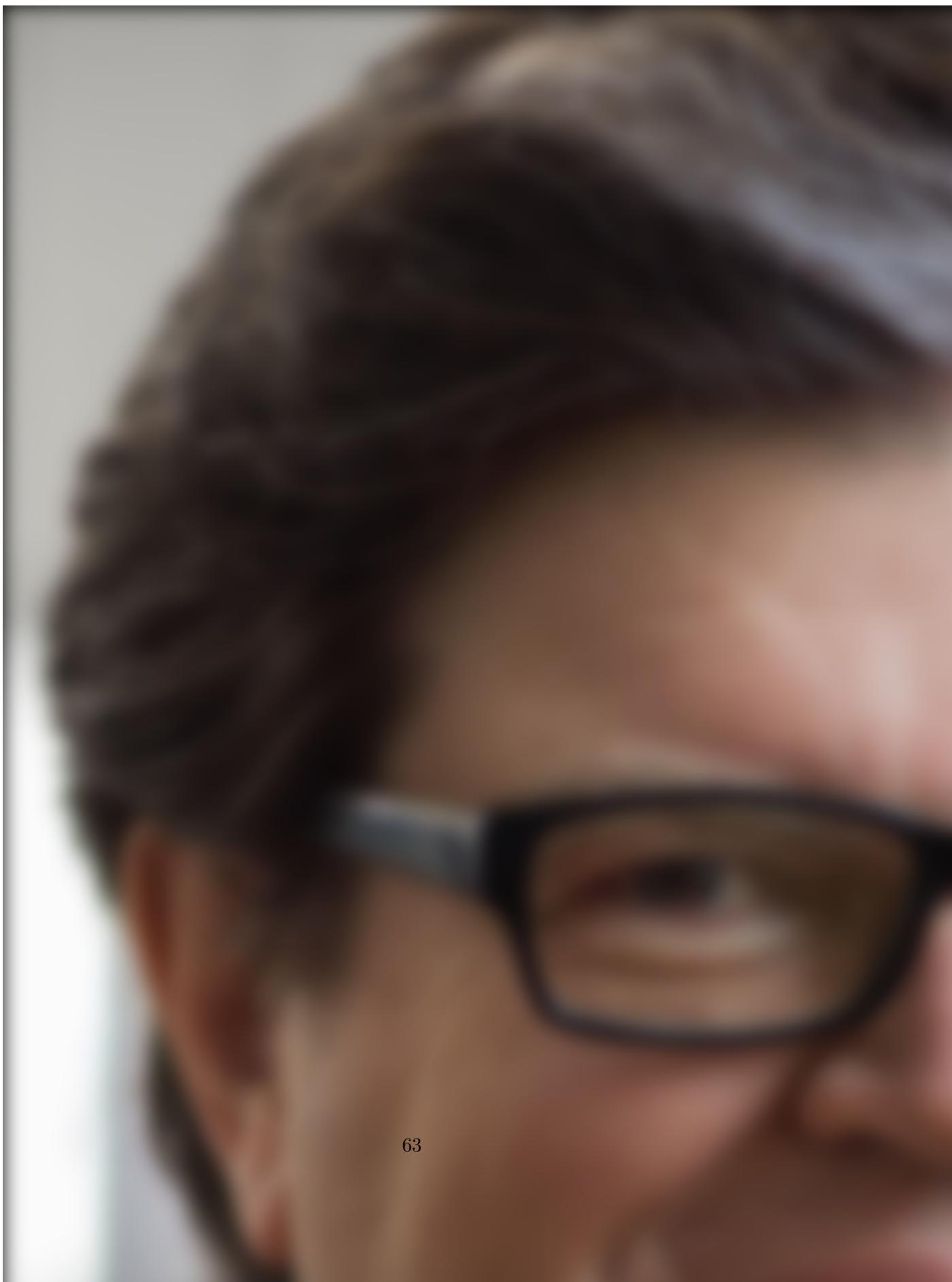
15 ex5/Deblurring/Yann LeCun/final inverted image.png



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



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



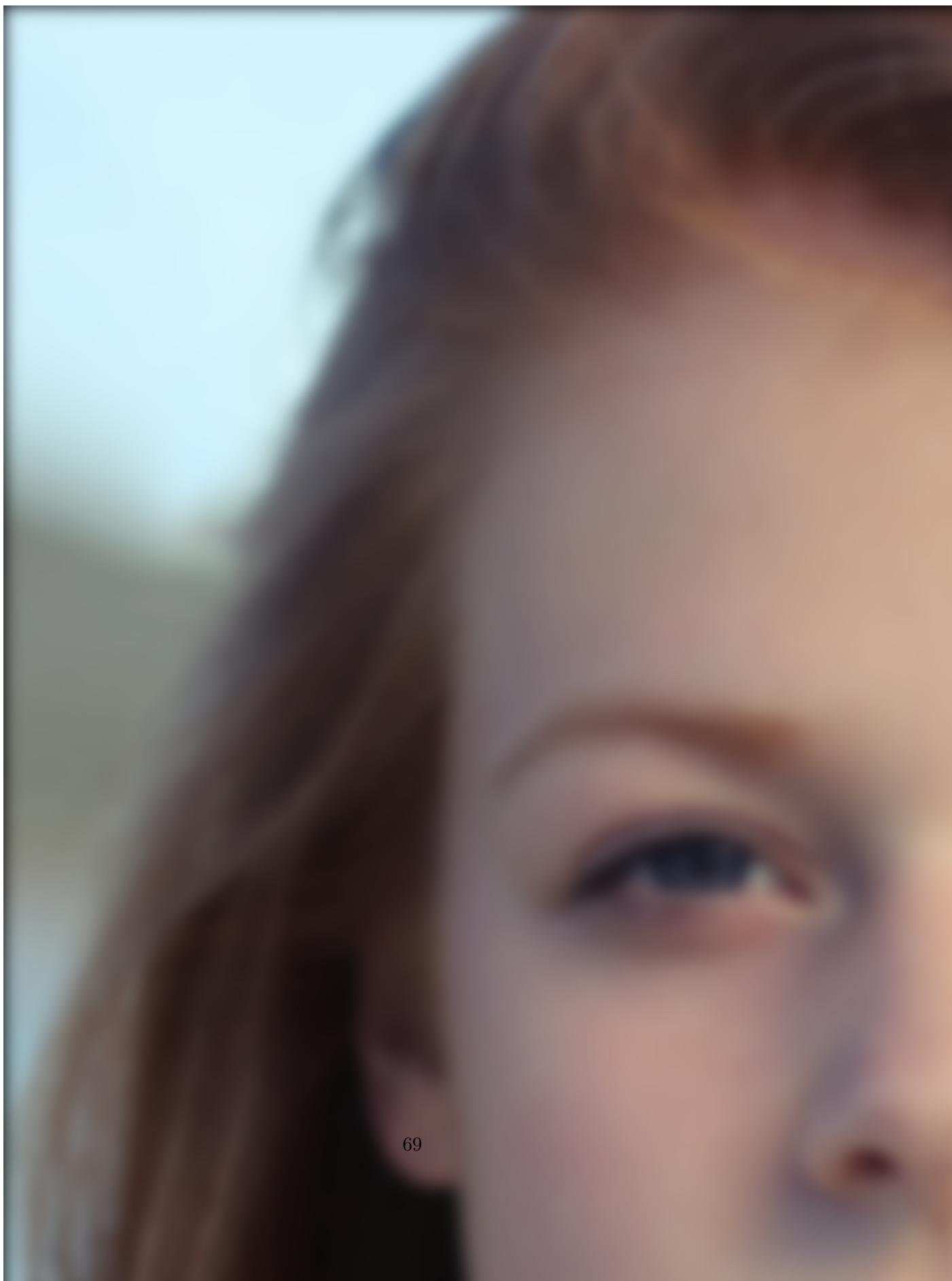
**18 ex5/Deblurring/im1 k85
w0.001/Christopher Campbell aligned.png**



**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



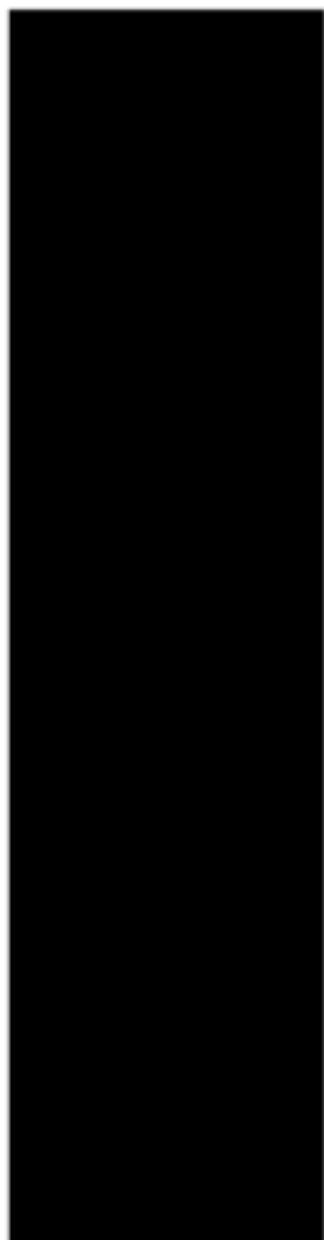
32 ex5/Inpainting/paint im2 s1000 w0.2/final
inverted image.png



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

