4xRealWebPhoto_v4

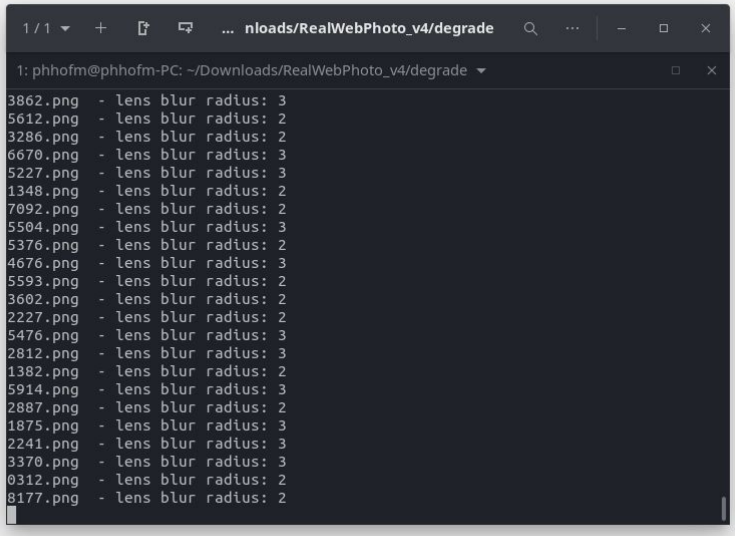
Version 4 Dataset Preparation Workflow

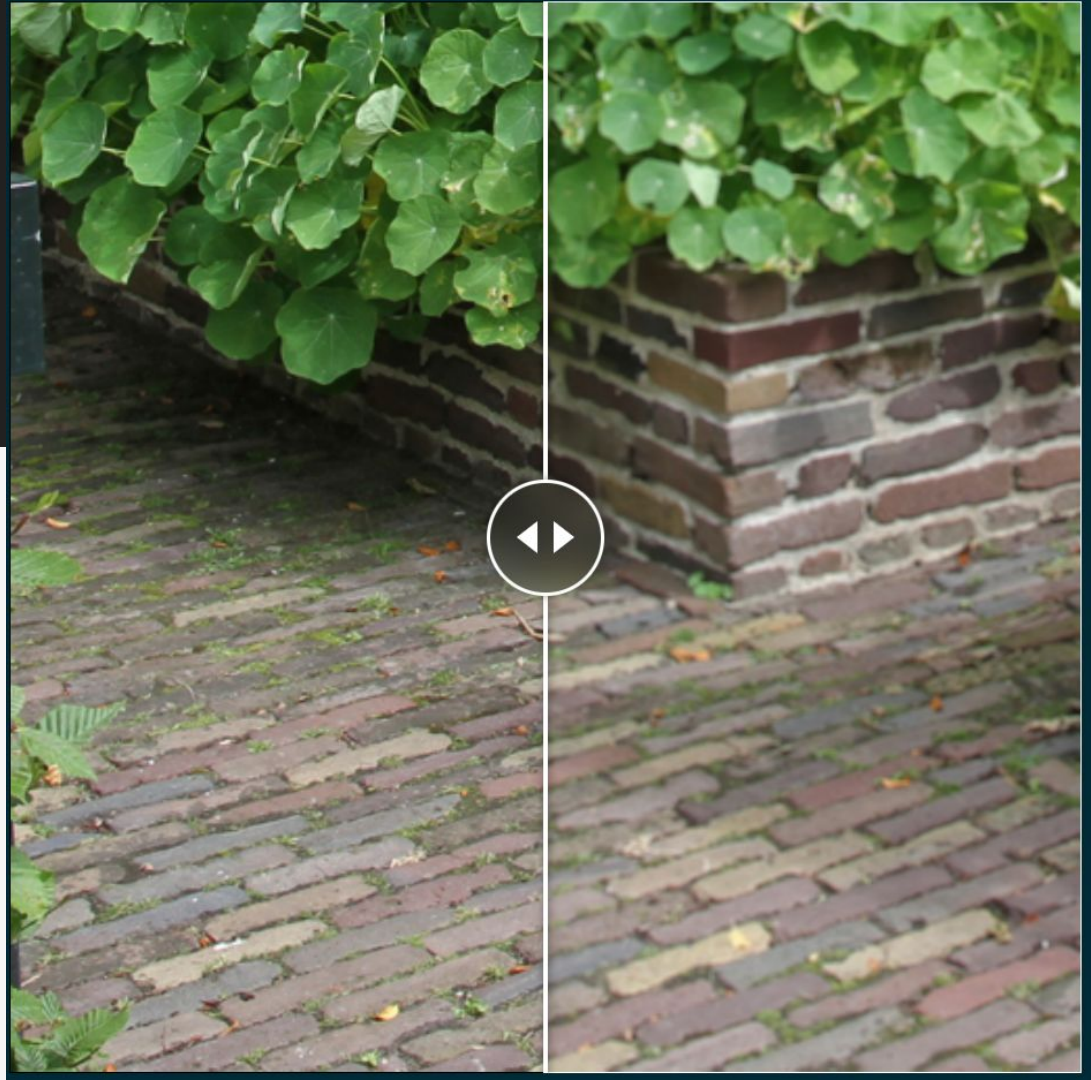
Based on musl’s nomos8k photo dataset

Adding Lens Blur

With radius 2 (weaker) to 3 (stronger) based on previous versions.

The example to the right is radius 2 (so weaker)





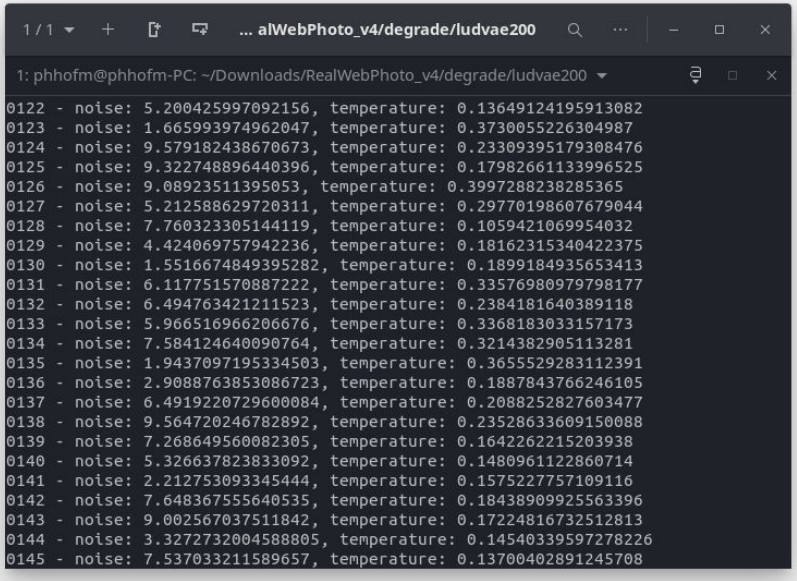
lensblur.py

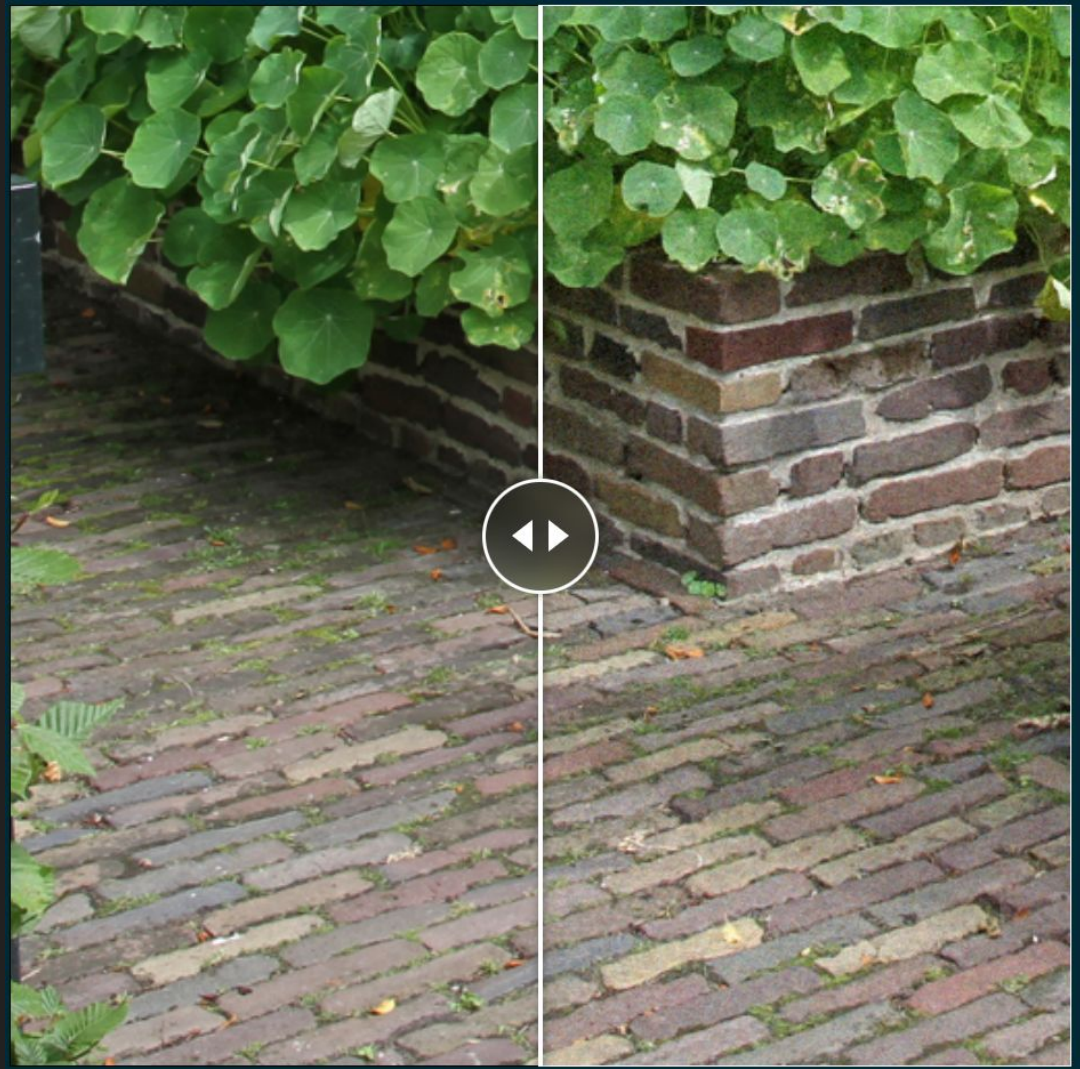
```python
import cv2
from blurgenerator import lens_blur
import os
import random

input_folder_path = "/home/phhofm/Downloads/RealWebPhoto_v4/nomos8k"
output_folder_path = "/home/phhofm/Downloads/RealWebPhoto_v4/nomos8k_lens"
textfile_path = "/home/phhofm/Downloads/RealWebPhoto_v4/degrade"
textfile_name = "lensblur"

def print_text_to_textfile(file_name, text_to_append):
    # Open the file in append & read mode ('a+')
    with open(file_name, "a+") as file_object:
        # Move read cursor to the start of file.
        file_object.seek(0)
        # If file is not empty then append '\n'
        data = file_object.read(100)
        if len(data) > 0:
            file_object.write("\n")
        # Append text at the end of file
        file_object.write(text_to_append)

for filename in os.listdir(input_folder_path):
    if filename.endswith('.png'):
        input_file_path = os.path.join(input_folder_path, filename)
        img=cv2.imread(input_file_path)
        random_lens_blur_radius = random.randint(2,3)
        result = lens_blur(img, radius=random_lens_blur_radius, components=4, exposure_gamma=2)
        output_file_path = os.path.join(output_folder_path, filename)
        cv2.imwrite(output_file_path, result)
        print_text_to_textfile(os.path.join(textfile_path, textfile_name), filename + '-' + str(random_lens_blur_radius))
        print(filename, ' - lens blur radius:', str(random_lens_blur_radius))
```

Adding Realistic Noise

With my ludvae200 model, at different/randomized strengths on the spectrum from noise 1 temperature 0.1 to noise 10 temperature 0.4

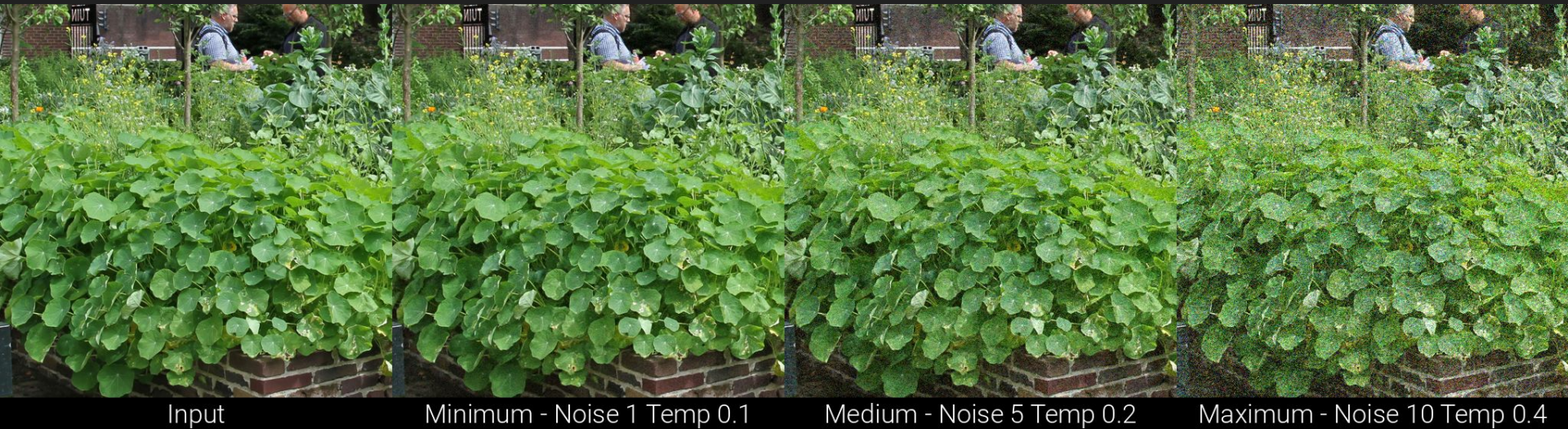
Example right: noise: 2.102180071606248, temperature: 0.13628391090182937





Ludvae200 Inference Script Settings

Visualization of the spectrum I used with my inference script for the 4xRealWebPhoto_v4 dataset when using my ludvae200 model.



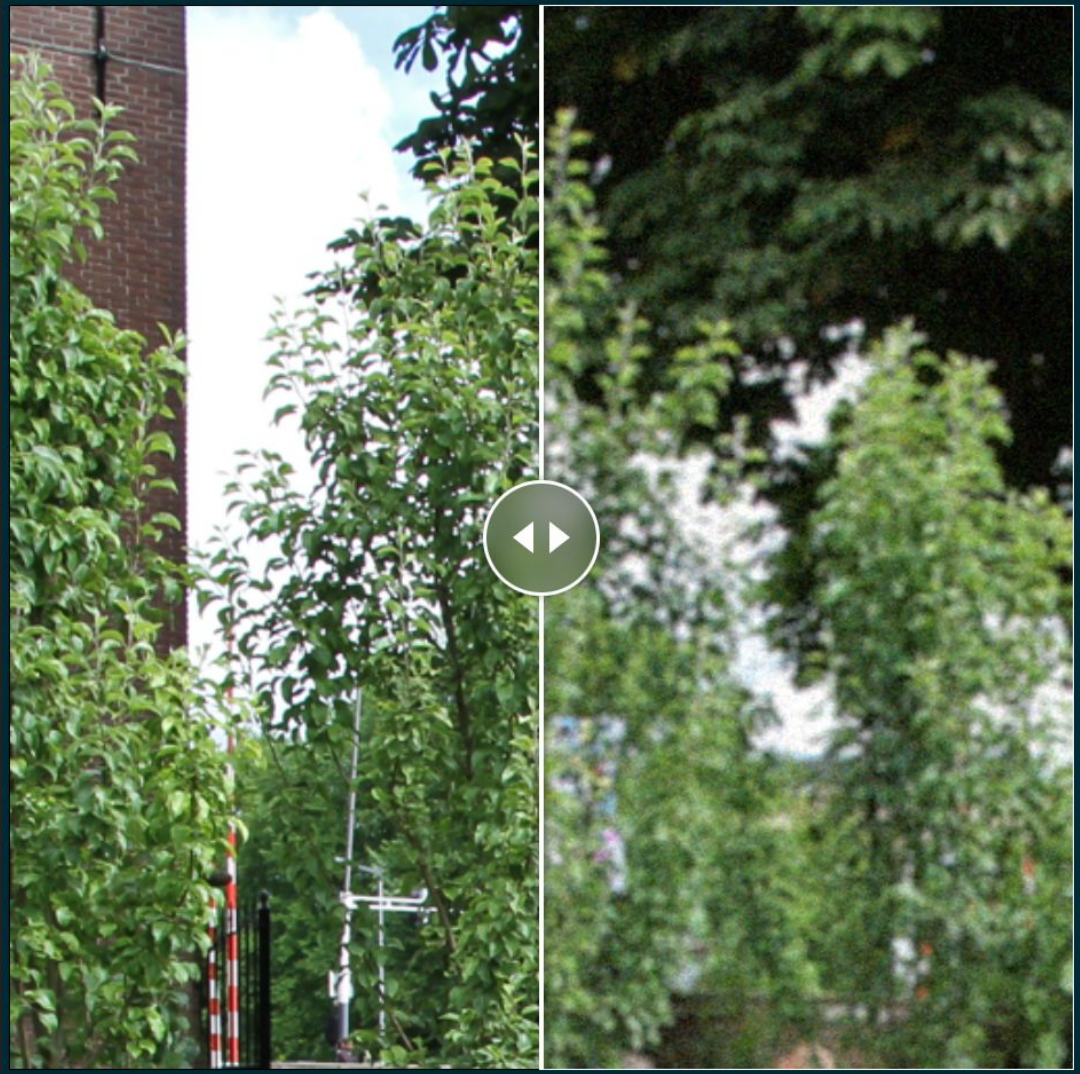
```python
import os.path
import torch
from utils import utils_image as util
from random import uniform
from models.network_ludvae import LUDVAE

def main():

    # ----------------------------------------
    # CHANGE THESE FOLLOWING PARAMETERS
    # ----------------------------------------

    # your input folder path
    H_path = '/home/phhofm/Downloads/RealWebPhoto_v4/nomos8k'
    # your output folder path
    G_path = '/home/phhofm/Downloads/RealWebPhoto_v4/nomos8k_ludvae200'
    # the folder path where you placed the ludvae200 model file
    model_pool = '/home/phhofm/Downloads/RealWebPhoto_v4/degrade/ludvae200'

    # The path where the applied_degradations.txt text file will be generated (or appended to if exists), use if textfile = True
    textfile_path = '/home/phhofm/Downloads/RealWebPhoto_v4/degrade'

    # Name of the output text file. If already existing, it will append new entries, never overwrite the existing one
    textfile_name = 'ludvae200.txt'

    # ----------------------------------------
    # Preparation
    # ----------------------------------------

    model_name = 'ludvae200'

    model_path = os.path.join(model_pool, model_name+'.pth')
    device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
    torch.cuda.empty_cache()

    util.mkdir(G_path)

    # Append given text as a new line at the end of file (if file not exists it creates and inserts line, otherwise it just appends newline)
    def print_text_to_textfile(file_name, text_to_append):
        # Open the file in append & read mode ('a+')
        with open(file_name, "a+") as file_object:
            # Move read cursor to the start of file.
            file_object.seek(0)
            # If file is not empty then append '\n'
            data = file_object.read(100)
            if len(data) > 0:
                file_object.write("\n")
            # Append text at the end of file
            file_object.write(text_to_append)

    # ----------------------------------------
    # load model
    # ----------------------------------------

    model = LUDVAE()
    states = torch.load(model_path)

    model.load_state_dict(states, strict=True)
    model.eval()

    for k, v in model.named_parameters():
        v.requires_grad = False

    model = model.to(device)

    H_paths = util.get_image_paths(H_path)

    # These strength settings are adopted towards my 4xRealWebPhoto_v4 dataset.
    # Adjust these to your needs
    # min: noise 1 with temperature 0.1 - my recommendation for lower limit, creates visually discernible noise, if you want anything in between to even no-noise outputs decrease this)
    # max: noise 10 with temperature 0.4 - pretty strong, if you want less strong decrease temperature upper limit from 0.4 to 0.2 for example

    for idx, img in enumerate(H_paths):
```

Lens with Realistic Noise

Example right:

Lens - radius: 2, components: 4, exposure_gamma: 2

Ludvae200 - noise: 2.102180071606248, temperature: 0.13628391090182937



Downsample

With Kim’s Dataset Destroyer

With down_up, linear, cubic_mitchell, lanczos, gaussian and box downsampling algorithms

```
# Scale settings
[scale]
# List of available scale algorithms (e.g., down_up,linear,cubic_catrom,cubic_mitchell,cubic_bspline,lanczos,gauss,hamming,hann,box,hermite,lagrange)
algorithms = down_up,linear,cubic_mitchell,lanczos,gauss,box
# List of available scale algorithms when applying down_up
down_up_algorithms = linear,cubic_mitchell,lanczos,gauss,box
# Whether to choose a random scale algorithm each time (True or False)
randomize = True
# Factor to scale your images to (e.g., 0.25, 0.50, 0.75) (0.25 = 25%, 0.50 = 50%)
size_factor = 0.25
# Range of values for down_up (e.g., 0.5,2.0) (0.5 = 50%, 2.0 = 200%)
range = 0.15,1.5
```

```
0476.png - scale: lanczos size factor=0.25
7667.png - scale: lanczos size factor=0.25
2750.png - scale: down_up scale1factor=0.82 scale1algorithm=cubic_mitchell scale2factor=0.31 scale2algorithm=box
8116.png - scale: box size factor=0.25
2161.png - scale: linear size factor=0.25
0553.png - scale: lanczos size factor=0.25
4918.png - scale: linear size factor=0.25
5043.png - scale: box size factor=0.25
4512.png - scale: down_up scale1factor=1.16 scale1algorithm=gauss scale2factor=0.22 scale2algorithm=linear
6249.png - scale: linear size factor=0.25
3088.png - scale: lanczos size factor=0.25
3454.png - scale: cubic_mitchell size factor=0.25
3527.png - scale: cubic_mitchell size factor=0.25
8352.png - scale: cubic_mitchell size factor=0.25
1999.png - scale: down_up scale1factor=1.32 scale1algorithm=box scale2factor=0.19 scale2algorithm=linear
0440.png - scale: box size factor=0.25
2773.png - scale: cubic_mitchell size factor=0.25
6884.png - scale: linear size factor=0.25
2582.png - scale: linear size factor=0.25
2178.png - scale: linear size factor=0.25
2717.png - scale: cubic_mitchell size factor=0.25
6463.png - scale: cubic_mitchell size factor=0.25
3176.png - scale: cubic_mitchell size factor=0.25
4434.png - scale: gauss size factor=0.25
7154.png - scale: cubic_mitchell size factor=0.25
5950.png - scale: lanczos size factor=0.25
1162.png - scale: down_up scale1factor=0.28 scale1algorithm=cubic_mitchell scale2factor=0.89 scale2algorithm=linear
4672.png - scale: gauss size factor=0.25
4496.png - scale: down_up scale1factor=1.44 scale1algorithm=gauss scale2factor=0.17 scale2algorithm=cubic_mitchell
6488.png - scale: linear size factor=0.25
3640.png - scale: gauss size factor=0.25
4475.png - scale: lanczos size factor=0.25
6006.png - scale: down_up scale1factor=0.94 scale1algorithm=cubic_mitchell scale2factor=0.27 scale2algorithm=gauss
1468.png - scale: gauss size factor=0.25
0974.png - scale: gauss size factor=0.25
0046.png - scale: box size factor=0.25
5835.png - scale: gauss size factor=0.25
3893.png - scale: lanczos size factor=0.25
7911.png - scale: lanczos size factor=0.25
7989.png - scale: linear size factor=0.25
4631.png - scale: box size factor=0.25
0769.png - scale: cubic_mitchell size factor=0.25
0192.png - scale: box size factor=0.25
1497.png - scale: down_up scale1factor=1.10 scale1algorithm=linear scale2factor=0.23 scale2algorithm=lanczos
6309.png - scale: box size factor=0.25
8054.png - scale: cubic_mitchell size factor=0.25
6012.png - scale: linear size factor=0.25
7994.png - scale: linear size factor=0.25
0837.png - scale: linear size factor=0.25
3488.png - scale: box size factor=0.25
1017.png - scale: linear size factor=0.25
6101.png - scale: lanczos size factor=0.25
6587.png - scale: down_up scale1factor=0.27 scale1algorithm=linear scale2factor=0.91 scale2algorithm=gauss
6450.png - scale: down_up scale1factor=0.45 scale1algorithm=gauss scale2factor=0.55 scale2algorithm=linear
2199.png - scale: box size factor=0.25
6717.png - scale: lanczos size factor=0.25
3480.png - scale: gauss size factor=0.25
3310.png - scale: linear size factor=0.25
5396.png - scale: lanczos size factor=0.25
3868.png - scale: gauss size factor=0.25
2600.png - scale: lanczos size factor=0.25
1244.png - scale: box size factor=0.25
5293.png - scale: box size factor=0.25
1339.png - scale: linear size factor=0.25
5359.png - scale: box size factor=0.25
0751.png - scale: box size factor=0.25
3963.png - scale: lanczos size factor=0.25
0921.png - scale: cubic_mitchell size factor=0.25
3943.png - scale: gauss size factor=0.25
8053.png - scale: linear size factor=0.25
2253.png - scale: gauss size factor=0.25
6373.png - scale: gauss size factor=0.25
```

Compression

Added jpg and webp (re)compression with Kim's Dataset Destroyer

```
# Compression settings
[compression]
# List of available compression algorithms (e.g., mpeg,mpeg2,h264,hevc,jpeg,webp,vp9)
# Using more intensive codecs (such as vp9) in combination with other degradations may result in ffmpeg errors
algorithms = jpeg,webp
# Whether to choose a random algorithm from the list
randomize = True
# JPEG Quality Levels
jpeg_quality_range = 40, 95
# WebP Quality levels
webp_quality_range = 45, 95
# H.264 video quality levels in CRF format
h264_crf_level_range = 23,32
# HEVC video quality levels in CRF format
hevc_crf_level_range = 25,34
# VP9 video quality levels in CRF format
vp9_crf_level_range = 25,35
# Quality control for MPEG codec. Range 1-31
mpeg_qscale_range = 2,15
# Quality  control for MPEG2 codec. Range 1-31
mpeg2_qscale_range = 2,15
```

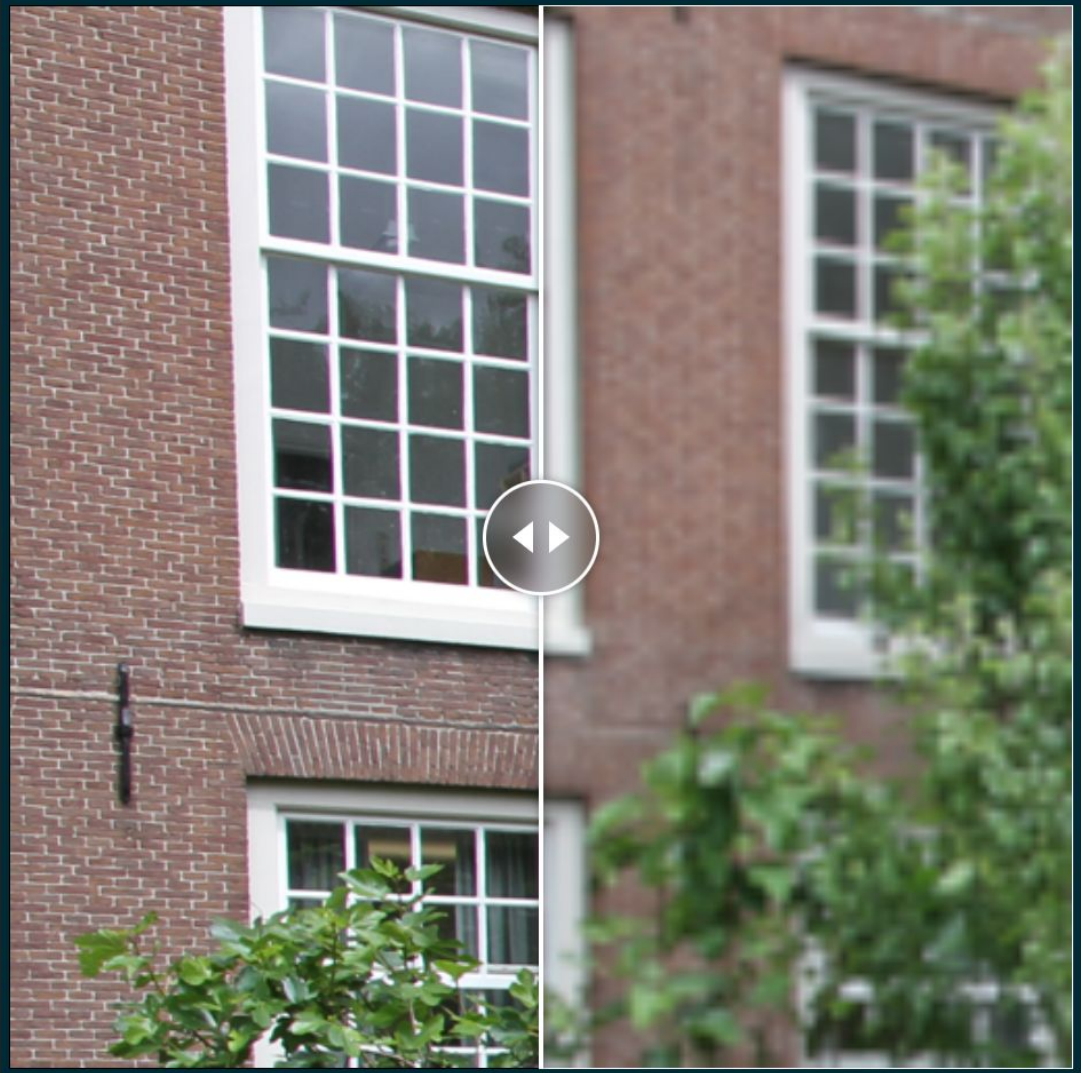
1. Variant

Downsample only

Simulates good quality downloaded photo

Example:

scale: linear size factor=0.25

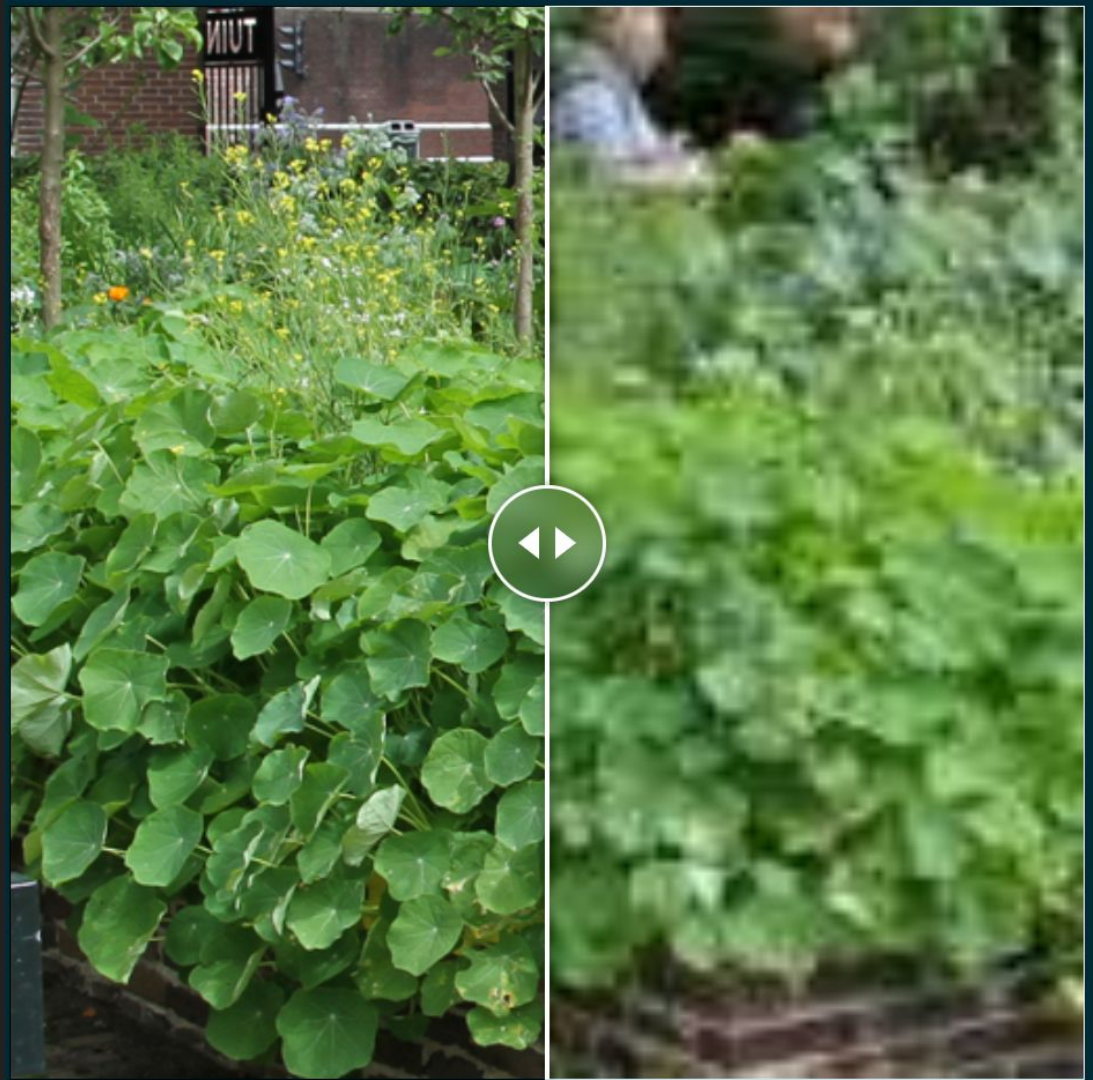


2. Variant

Downsample and compression simulates a photo that had been uploaded once (downsamples and compressed by service provider)

Example:

scale: box size factor=0.25, compression: jpeg quality=54

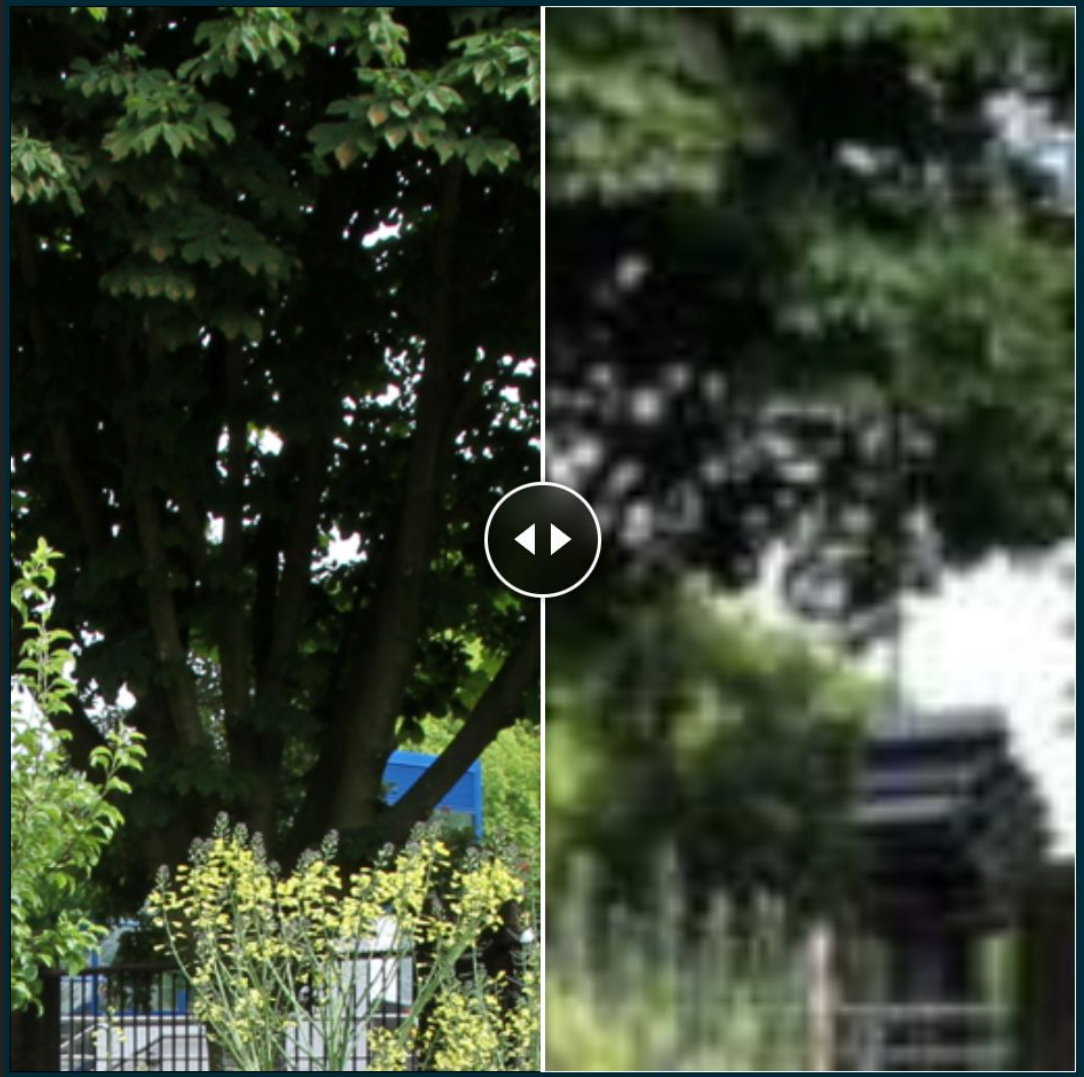


3. Variant

Scale, compression, scale, compression simulates a photo that had been uploaded to the web (processed by service provider) then downloaded and re-uploaded to the web

Example right:

scale: box size factor=0.5, compression: jpeg quality=54, scale: down_up scale1factor=1.06 scale1algorithm=box scale2factor=0.47 scale2algorithm=gauss, compression: jpeg quality=66



Variants

I skip a bit here, but this process had been repeated for all the other intermediate output folders

We finally end up with 12 output folder / variants.

Non degraded: Down, down_comp and down_comp_down_comp

Lens Blur: Down, down_comp and down_comp_down_comp

Ludvae200: Down, down_comp and down_comp_down_comp

Lens Blur with Ludvae200: Down, down_comp and down_comp_down_comp

LR Folders

Now I merged all these output folders into 1 LR folder.

I first wanted to make multiple different LR folders as to ensure a good distribution of degradation values, and switch LR folders during training.

But I opted to only create 1 LR folder and training a DAT-2 model it showed that the model would be able to learn enough this way.

In the following slide is how i built the LR folder:

down	down_comp	down_comp_	lens_down	lens_down_c	lens_down_c	ludvae200_d	ludvae200_d	ludvae200_d	lens_ludvae_	lens_ludvae_	lens_ludvae_dow
0001	0002	0003	0004	0005	0006	0007	0008	0009	0010	0011	0012
0013	0014	0015	0016	0017	0018	0019	0020	0021	0022	0023	0024
0025	0026	0027	0028	0029	0030	0031	0032	0033	0034	0035	0036
0037	0038	0039	0040	0041	0042	0043	0044	0045	0046	0047	0048
0049	0050	0051	0052	0053	0054	0055	0056	0057	0058	0059	0060
0061	0062	0063	0064	0065	0066	0067	0068	0069	0070	0071	0072
0073	0074	0075	0076	0077	0078	0079	0080	0081	0082	0083	0084
0085	0086	0087	0088	0089	0090	0091	0092	0093	0094	0095	0096
0097	0098	0099	0100	0101	0102	0103	0104	0105	0106	0107	0108
0109	0110	0111	0112	0113	0114	0115	0116	0117	0118	0119	0120
0121	0122	0123	0124	0125	0126	0127	0128	0129	0130	0131	0132
0133	0134	0135	0136	0137	0138	0139	0140	0141	0142	0143	0144
0145	0146	0147	0148	0149	0150	0151	0152	0153	0154	0155	0156
0157	0158	0159	0160	0161	0162	0163	0164	0165	0166	0167	0168
0169	0170	0171	0172	0173	0174	0175	0176	0177	0178	0179	0180
0181	0182	0183	0184	0185	0186	0187	0188	0189	0190	0191	0192
0193	0194	0195	0196	0197	0198	0199	0200	0201	0202	0203	0204
0205	0206	0207	0208	0209	0210	0211	0212	0213	0214	0215	0216
0217	0218	0219	0220	0221	0222	0223	0224	0225	0226	0227	0228
0229	0230	0231	0232	0233	0234	0235	0236	0237	0238	0239	0240
0241	0242	0243	0244	0245	0246	0247	0248	0249	0250	0251	0252
0253	0254	0255	0256	0257	0258	0259	0260	0261	0262	0263	0264
0265	0266	0267	0268	0269	0270	0271	0272	0273	0274	0275	0276
0277	0278	0279	0280	0281	0282	0283	0284	0285	0286	0287	0288
0289	0290	0291	0292	0293	0294	0295	0296	0297	0298	0299	0300
0301	0302	0303	0304	0305	0306	0307	0308	0309	0310	0311	0312
0313	0314	0315	0316	0317	0318	0319	0320	0321	0322	0323	0324
0325	0326	0327	0328	0329	0330	0331	0332	0333	0334	0335	0336
0337	0338	0339	0340	0341	0342	0343	0344	0345	0346	0347	0348
0349	0350	0351	0352	0353	0354	0355	0356	0357	0358	0359	0360
0361	0362	0363	0364	0365	0366	0367	0368	0369	0370	0371	0372
0373	0374	0375	0376	0377	0378	0379	0380	0381	0382	0383	0384
0385	0386	0387	0388	0389	0390	0391	0392	0393	0394	0395	0396
0397	0398	0399	0400	0401	0402	0403	0404	0405	0406	0407	0408
0409	0410	0411	0412	0413	0414	0415	0416	0417	0418	0419	0420
0421	0422	0423	0424	0425	0426	0427	0428	0429	0430	0431	0432
0433	0434	0435	0436	0437	0438	0439	0440	0441	0442	0443	0444