Lab2 - Deep Image Prior

Wen-Jie Tseng (0556146)
Deep Learning and Practice - 2018 Spring Semester

1. INTRODUCTION

Deep Image Prior presented that the structure of a generator network is sufficient to capture a great deal of low-level images statistics prior to any learning. In this lab we demonstrated the concept of deep images prior through showing parametrization process. A randomly-initialized neural network can be used as a handcrafted prior. Then we applied deep image prior in the standard inverse problems, for example, denoising, super-resolution, and inpainting.

2. EXPERIMENT SETUP

2.1 The Detail of Model and Hyper-Parameters

I used the skip model in skip.py from sample code.

- Requirement 1: input depth = 3 (drawn from $U(0, 1/10)$), number of channels down = up = [8, 16, 32, 64, 128], number of channels skip = [0, 0, 0, 4, 4], filter size down = up = [3, 3, 3, 3, 3], filter size down skip = [NA, NA, NA, 1, 1], Gaussian noise for each iteration = 1/30, iteration = 2400, learning rate = 0.01, upsampling method = bilinear.
- Requirement 2: input depth = 32 (drawn from $U(0, 1/10)$), number of channels down = up = [128, 128, 128, 128, 128], number of channels skip = [4, 4, 4, 4, 4], filter size down = up = [3, 3, 3, 3, 3], filter size down skip = [1, 1, 1, 1, 1], Gaussian noise for each iteration = 1/30, iteration = 1800, learning rate = 0.01, upsampling method = bilinear.
- Requirement 3: input depth = 32 (drawn from $U(0, 1/10)$), number of channels down = up = [128, 128, 128, 128, 128], number of channels skip = [4, 4, 4, 4, 4], filter size down = up = [3, 3, 3, 3, 3], filter size down skip = [1, 1, 1, 1, 1], Gaussian noise for each iteration = 1/30, iteration = 2000, downsampling factor = 4, learning rate = 0.01, upsampling method = bilinear.

3. RESULT

3.1 Requirement 1: Parametrization

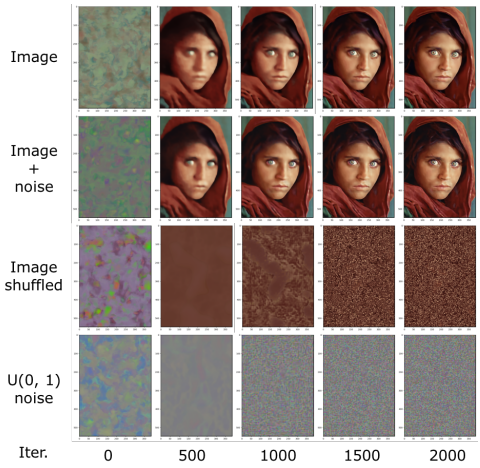

Figure 1. The images inverted in 4 different setup.

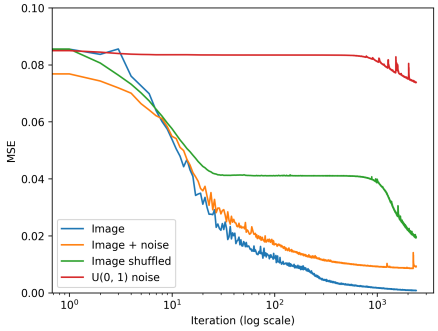

Figure 2. A parametrization with high noise impedance.

First we have to replicate the result of parametrization figure with our own image. I chose Afghan girl as our image. Next we generated 4 input images to see how CNN can learn from them. The images inverted were shown in Figure 1, we can directly make CNN learn from these image prior. The curves in Figure 2 show that the MSEs all decreased and converged at difference iterations. For image plus noise, I used "get_noisy_img" function in sample code, add a $U(0, 1/10)$ noise. For image shuffled, I divided the original image into 2-pixel blocks and randomly switched the blocks by "np.random.shuffled()". The image shuffled and U(0, 1) noise learned slower than the other two, but eventually decreasing around iteration 1000.

3.2 Requirement 2: Denoising

Deep image prior can perform image denoising. The given image was assigned as the variables of our model. Then train this model with a 32 x W x H dimension noise from $U(0, 1/10)$. Our model can learn the original image without noise. I trained the denoising process in three times. The PSNR before training was 20.36406. The PSNR after training got 29.90985, 30.56138, and 29.83106 respectively. Reached PSNR 30.10076 in average. The images inverted were shown in Figure 3

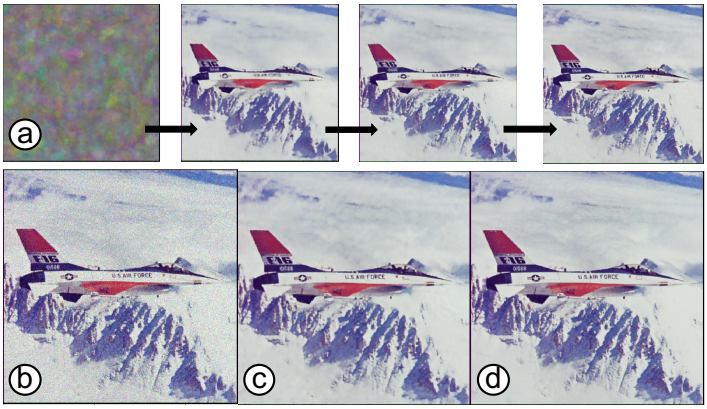

Figure 3. Image inverted in denoising: a) the image inverted from 0, 500, 1000, 1500 iterations, b) the original noisy image, c) our final image in 1800 iteration, and d) ground truth.

3.3 Requirement 3: Super-resolution

In super-resolution, the MSE was calculated by the low-resolution variables and the downsampled variables from high-resolution. The image has to be cropped to fitted the dimension. Next "total_loss.backward()" update the values in high-resolution model. Repeated this process and I finally obtained PSNR 23.8396 (final output and ground truth in high-resolution). The images inverted and final result compare are shown in Figure 4 and 5.

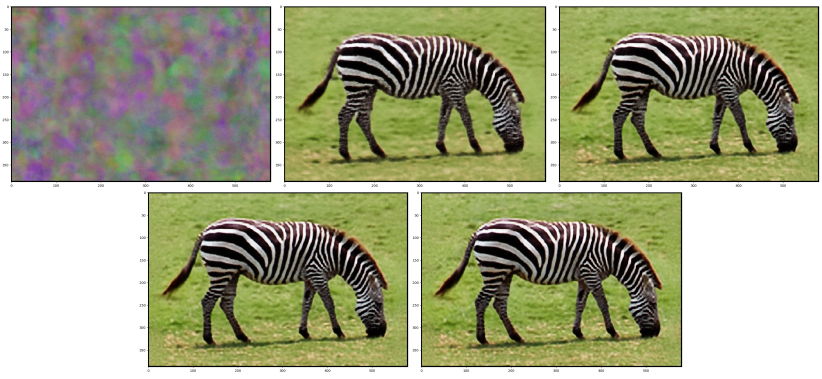

Figure 4. The images inverted in 0, 400, 800, 1200, 1600, and 2000 iterations.

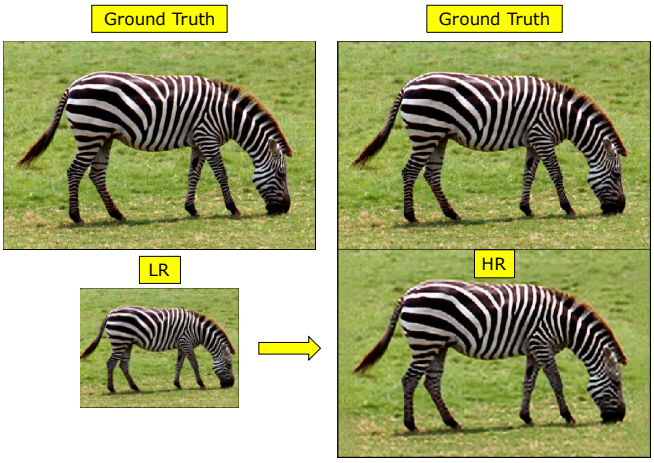

Figure 5. Compare the low-resolution, high-resolution, and ground-truth.

3.4 Bonus: Inpainting

In image inpainting I followed the sample code, and the result was shown in 6.

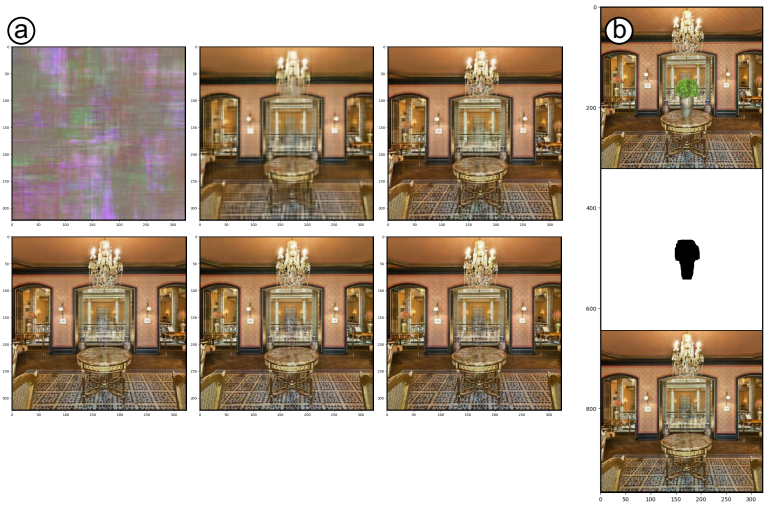

Figure 6. Inpainting: a) The images inverted in 0, 500, 1000, 1500, 2000, 2500, and 3000 iterations. b) the original image, mask and the final output.

4. DISCUSSION

In requirement 1, since we only trained with 2400 iteration, it was not easy to see the decreasing and converging for a larger image. The first figure I tried was a falcon image in 1200 x 800 pixels. The MSEs of image plus noise, image shuffled, and U(0, 1) noise were not decrease in 2400 iterations. After changing into the 400 x 550 image, things went well. Another problem I met was I mistakenly shuffled image only in RGB channels. In that case, the MSE curve of image shuffled went just like the original image, the only difference was the color. If one only shuffled each element in numpy array will turn the image into a random noise, so I turn to implement a function to switch image in 2 pixel blocks.

For image denoising, the denoising process may need a criterion for comparing how well the image was denoised. If an iteration reaches the criterion, the denoising process can stop. Otherwise, the MSE of training may vary each time.

At last, the super-resolution took me some time to understand the sample code. I think doing downsample to the model in high-resolution, compare MSE in low-resolution, and backpropagate with that MSE value in high-resolution model, are the formular in the description about super-resolution in deep image prior's paper. I am wondering this learning phenomenon of convolution neural network shows only in image? Maybe one can try this prior effect in sound or NLP. The 4 different loss function are the crucial part to perform each technique in denoising, super-resolution, and inpainting.

REFERENCES

1. Deep image prior project page.
2. Wikipedia: Afghan girl