Exploiting Deep Generative Prior for Versatile Image Restoration and Manipulation

Supplementary Material

Xingang Pan[1], Xiaohang Zhan[1], Bo Dai[1],
Dahua Lin[1], Chen Change Loy[2], and Ping Luo[3]

[1] The Chinese University of Hong Kong
[2] Nanyang Technological University [3] The University of Hong Kong

In this supplementary material, we provide more qualitative results and the implementation details in our experiments. We also recommend readers to refer to the attached videos for animated restoration and manipulation effects.

1 Qualitative Examples

We extend the figures of the main paper with more examples, as shown from Fig.1 to Fig.11.

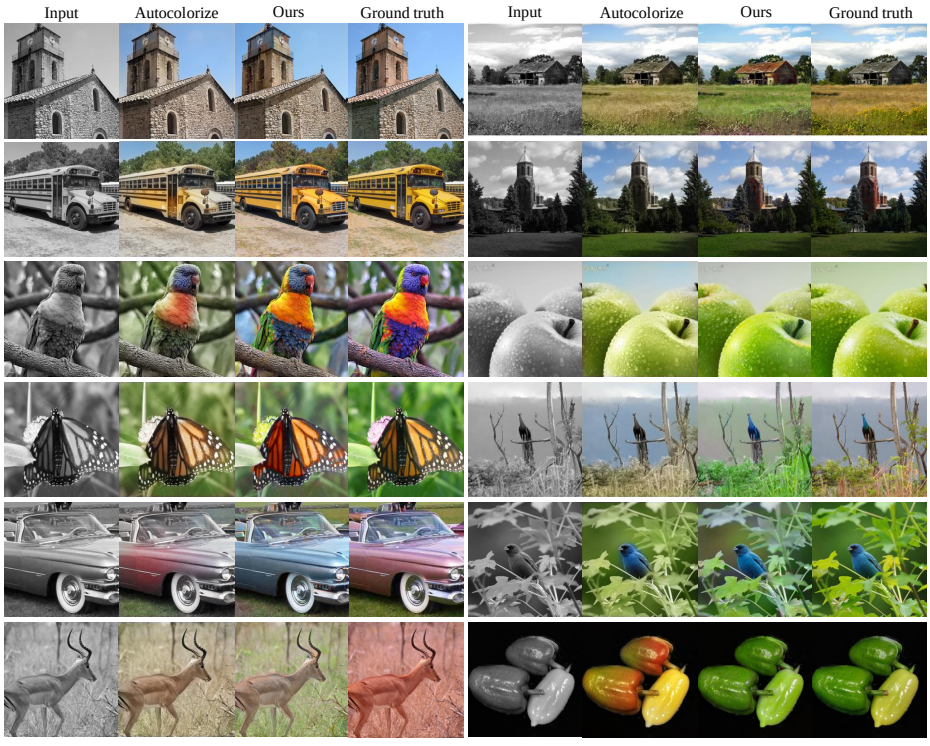

Fig. 1. Colorization. This is an extension of Fig.5 in the main paper.

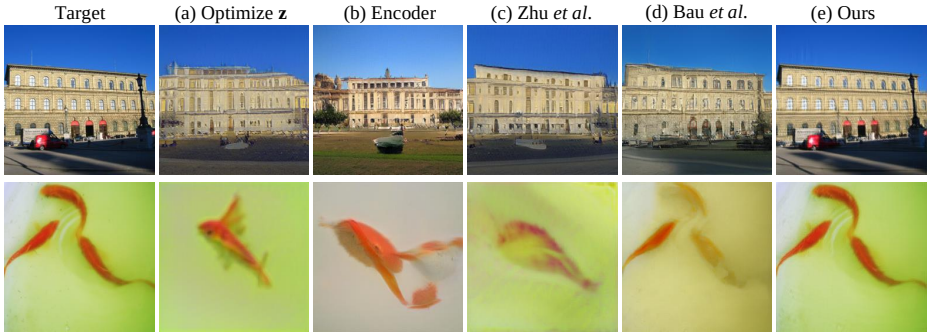

Fig. 2. Image reconstruction. We compare our method with other GAN-inversion methods including (a) optimizing latent vector [4, 1], (b) learning an encoder [9], (c) a combination of (a)(b) [9], and (d) adding small perturbations to early stages based on (c) [2].

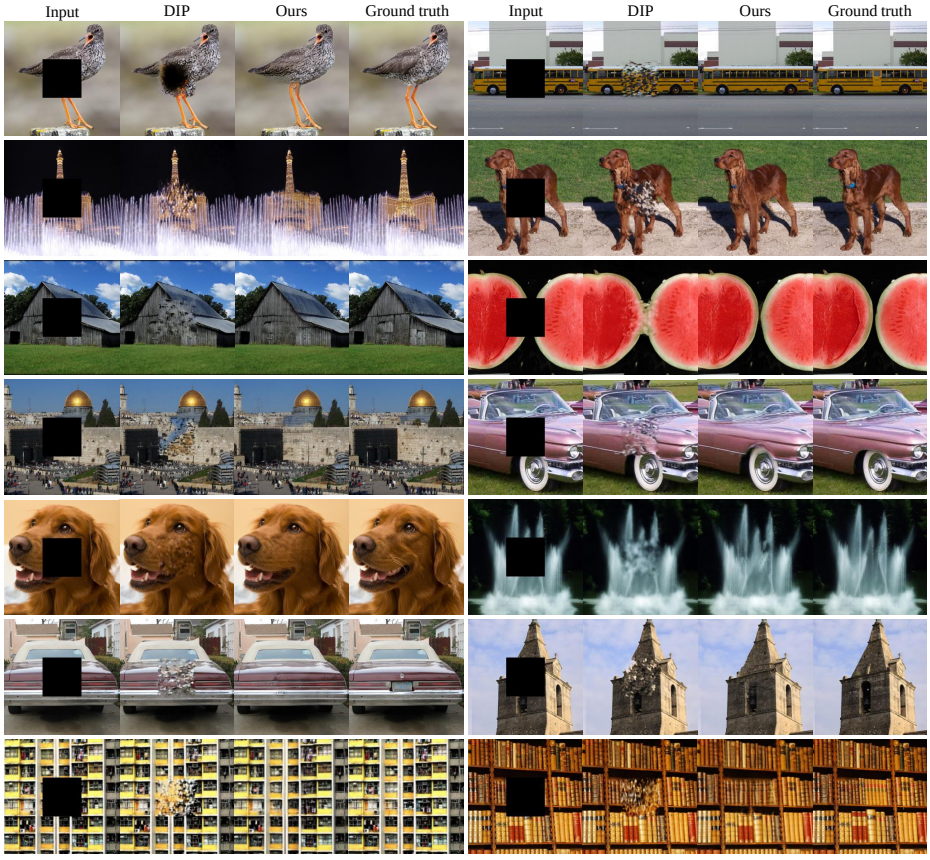

Fig. 3. Inpainting. This is an extension of Fig.6 in the main paper. The proposed DGP tends to recover the missing part in harmony with the context. Images of the last row are scratched from the Internet.

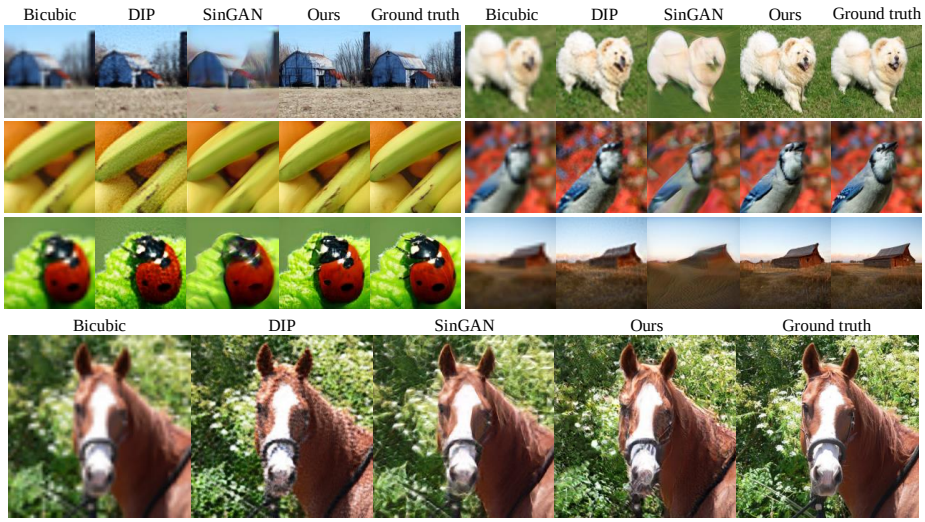

Fig. 4. Super-resolution ($\times 4$) on 32×32 (above) and 64×64 (below) size images. This is an extension of Fig.7 in the main paper.

Fig. 5. The reconstruction process of DGP in various image restoration tasks.

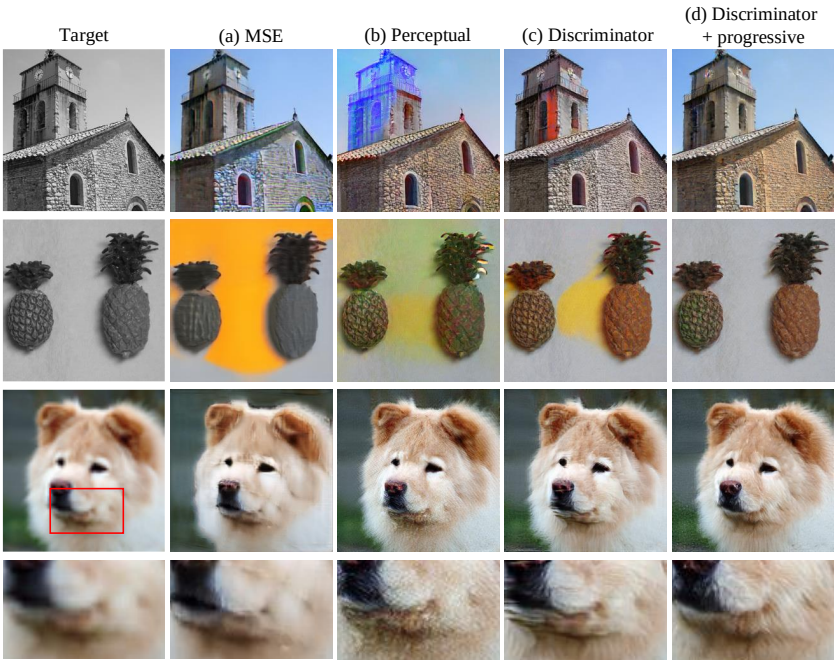

Fig. 6. Comparison of different loss types and optimization techniques in colorization and super-resolution, including (a) MSE, (b) perceptual loss with VGG network [6], (c) discriminator feature matching loss, and (d) combined with progressive reconstruction.

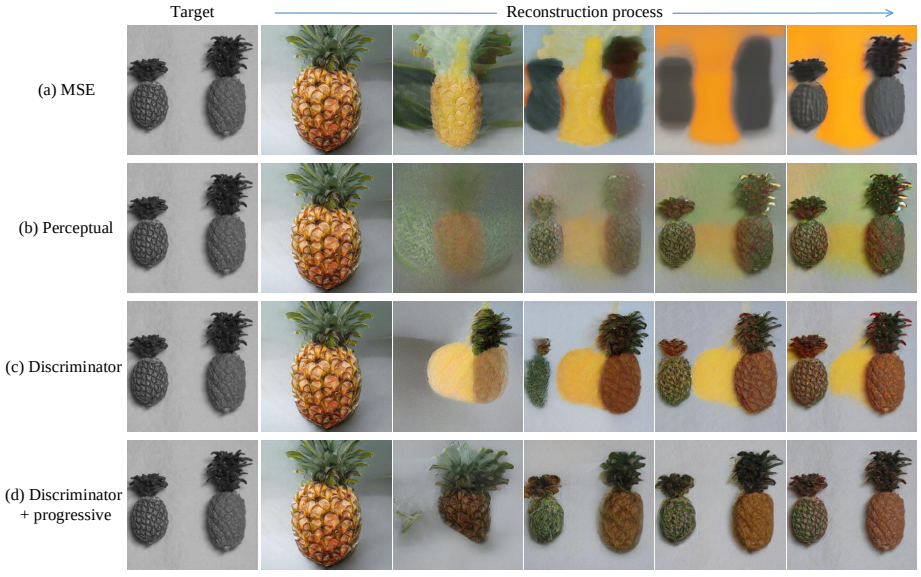

Fig. 7. Comparison of different loss types and optimization techniques when fine-tuning the generator to restore the image.

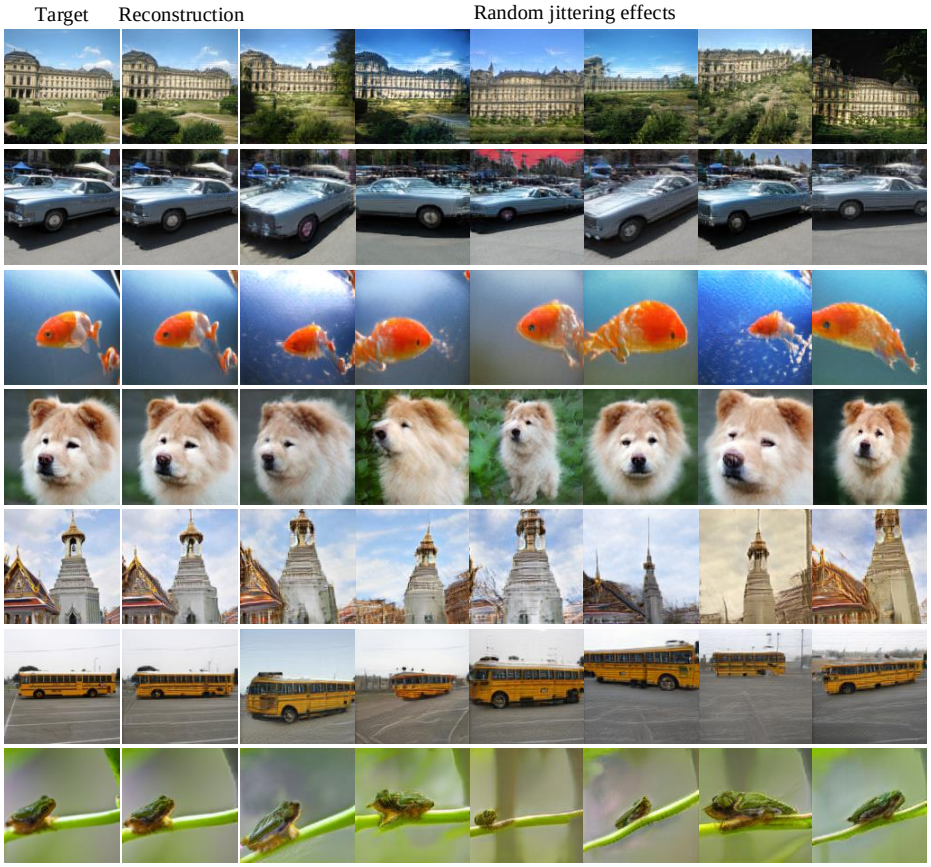

Fig. 8. Random jittering. This is an extension of Fig.11 in the main paper.

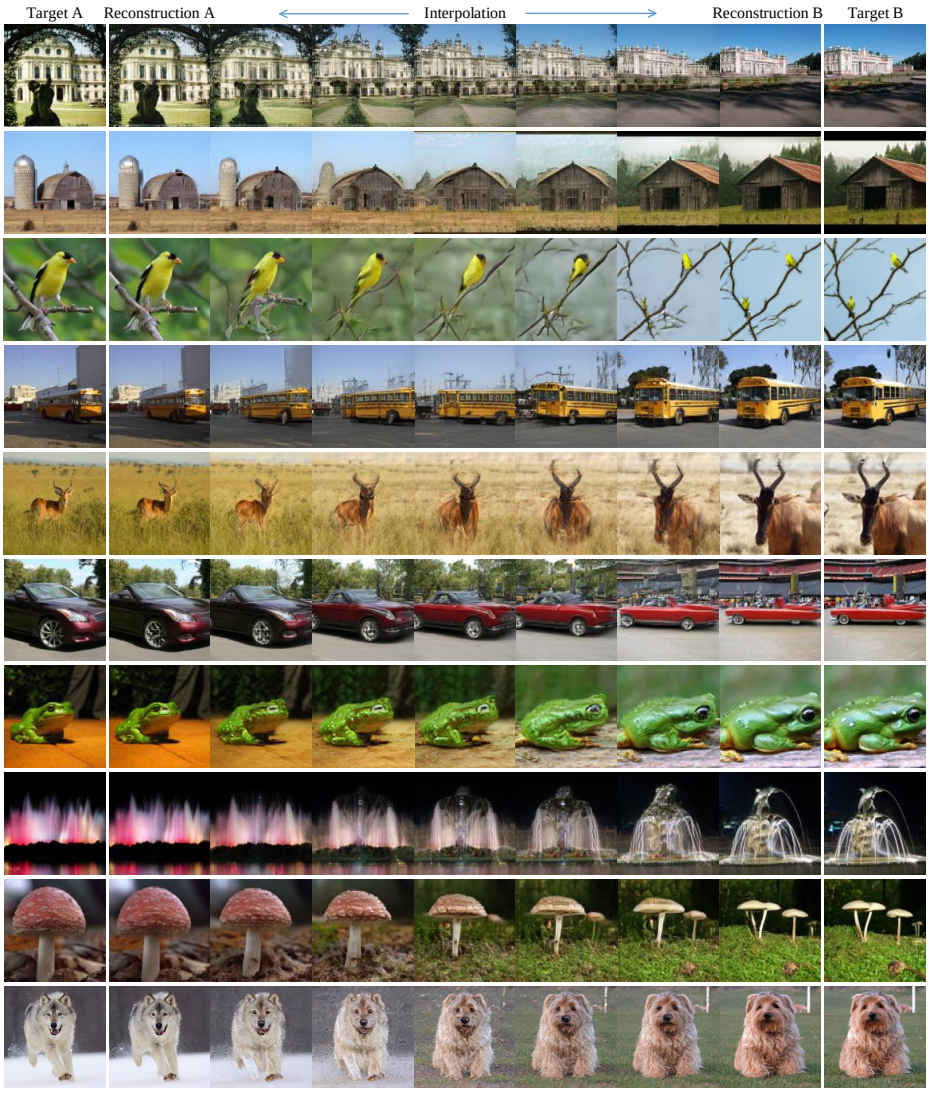

Fig. 9. Imaeg morphing. This is an extension of Fig.12 in the main paper.

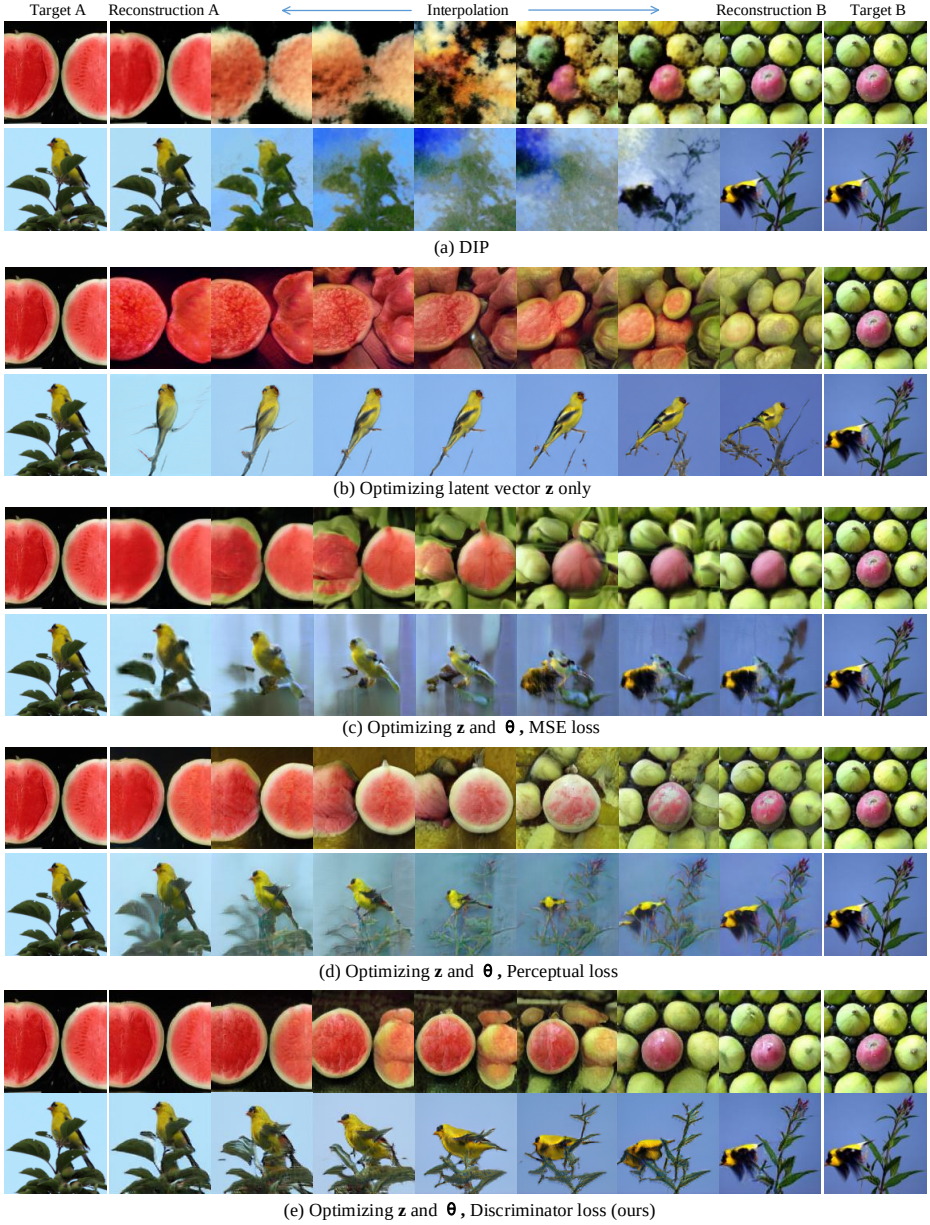

Fig. 10. Comparison of various methods in image morphing, including (a) using DIP, (b) optimizing the latent vector $\mathbf{z}$ of the pre-trained GAN, and (c)(d)(e) optimizing both $\mathbf{z}$ and the generator parameter $\boldsymbol{\theta}$ with (c) MSE loss, (d) perceptual loss with VGG network [6], and (e) discriminator feature matching loss. (b) fails to produce accurate reconstruction while (a)(c)(d) could not obtain realistic interpolation results. In contrast, our results in (e) are much better.

Fig. 11. Category transfer. The red box shows the target, and the blue box shows the reconstruction. Others are category transfer results.

2 Implementation Details

Architectures. We adopt the BigGAN[3] architechtures of 128^2 and 256^2 resolutions in our experiments. For the 128^2 resolution, we use the best setting of [3], which has a channel multiplier of 96 and a batchsize of 2048. As for the 256^2 resolution, the channel multiplier and batchsize are respectively set to 64 and 1920 due to limited GPU resources. We train the GANs on the ImageNet training set, and the 128^2 and 256^2 versions have Inception scores of 103.5 and 94.5 respectively. Our experiments are conducted based on PyTorch [8].

Initialization. In order to ease the optimization goal of Eq.4 in the paper, it is a good practice to start with a latent vector $\mathbf{z}$ that produces an approximate reconstruction. Therefore, we randomly sample 500 images using the GAN, and select the nearest neighbor of the target image under the discriminator feature metric as the starting point. Since encoder based methods tend to fail for degraded input images, they are not used in this work.

Note that in BigGAN, a class condition is needed as input. Therefore, in order to reconstruct an image, its class condition is required. This image classification problem could be solved by training a corresponding deep network classfier and is not the focus of this work, hence we assume the class label is given except for the adversarial defense task. For adversarial defense and images whose classes are not given, both the latent vector $\mathbf{z}$ and the class condition are randomly sampled.

Fine-tuning. With the above pre-trained BigGAN and initailized latent vector $\mathbf{z}$, we fine-tune both the generator and the latent vector to reconstruct a target image. As the batchsize is only 1 during fine-tuning, we use the tracked global statistics (*i.e.*, running mean and running variance) for the batch normalization (BN) [5] layers to prevent inaccurate statistic estimation. The discriminator of BigGAN is composed of a number of residual blocks (6 blocks and 7 blocks for 128^2 and 256^2 resolution versions respectively). The output features of these blocks are used as the discriminator loss, as described in Eq.(6) of the paper. In order to prevent the latent vector from deviating too much from the prior gaussian distribution, we add an additional L2 loss to the latent vector $\mathbf{z}$ with a loss weight of 0.02. We adopt the ADAM optimizer [7] in all our experiments. The detailed training settings for various tasks are listed from Table.1 to Table.6, where the parameters in these tables are explained below:

Blocks num.: the number of generator blocks to be fine-tuned. For example, for blocks num.=1, only the shallowest block is fine-tuned.
D loss weight: the factor multiplied to the discriminator loss.
MSE loss weight: the factor multiplied to the MSE loss.
Iterations: number of training iterations of each stage.
G lr: the learning rate of the generator blocks.
z lr: the learning rate of the latent vector $\mathbf{z}$.

For inpainting and super-resolution, we use a weighted combination of discriminator loss and MSE loss, as the MSE loss is beneficial for the PSNR metric. We

Table 1. The fine-tuning setting of colorization. The explanation of these parameters are in the main text

Stage	1	2	3	4	5
Blocks num.	1	2	3	4	5
D loss weight	1	1	1	1	1
MSE loss weight	0	0	0	0	0
Iterations	200	200	300	400	300
G lr	5e-5	5e-5	5e-5	5e-5	2e-5
z lr	2e-3	1e-3	5e-4	5e-5	2e-5

Table 2. The fine-tuning setting of inpainting. In this task we also fine-tune the class embedding apart from the generator blocks

Stage	1	2	3	4
Blocks num.	5	5	5	5
D loss weight	1	1	0.1	0.1
MSE loss weigh	1	1	100	100
Iterations	400	200	200	200
G lr	2e-4	1e-4	1e-4	1e-5
z lr	1e-3	1e-4	1e-4	1e-5

Table 3. The fine-tuning setting of super-resolution. This setting is biased towards MSE loss

Stage	1	2	3	4	5
Blocks num.	1	2	3	4	5
D loss weight	1	1	1	0.5	0.1
MSE loss weight	1	1	1	50	100
Iterations	200	200	200	200	200
G lr	2e-4	2e-4	1e-4	1e-4	1e-5
z lr	1e-3	1e-3	1e-4	1e-4	1e-5

Table 4. The fine-tuning setting of super-resolution. This setting is biased towards discriminator loss

Stage	1	2	3	4	5
Blocks num.	1	2	3	4	5
D loss weight	1	1	1	1	1
MSE loss weight	1	1	1	1	1
Iterations	200	200	200	200	200
G lr	5e-5	5e-5	2e-5	1e-5	1e-5
z lr	2e-3	1e-3	2e-5	1e-5	1e-5

Table 5. The fine-tuning setting of adversarial defense. The fine-tuning is stopped if the MSE loss reaches 5e-3

	stage 1	stage 2
Blocks num.	6	6
D loss weight	0	0
MSE loss weight	1	1
Iterations	100	900
G lr	2e-7	1e-4
z lr	5e-2	1e-4

Table 6. The fine-tuning setting of manipulation tasks including random jittering, image morphing, and category transfer

	stage 1	stage 2	stage 3
Blocks num.	5	5	5
D loss weight	1	1	1
MSE loss weight	0	0	0
Iterations	125	125	100
G lr	2e-7	2e-5	2e-6
z lr	1e-1	2e-3	2e-6

also seamlessly replace BN with instance normalization (IN) for the setting in Table. 2, Table. 3, and Table. 5, which enables higher learning rate and leads to better PSNR. This is achieved by initialize the scale and shift parameters of IN with the statistics of the output features of BN. Our quantitative results on adversarial defense is based on the 256^2 resolution model, while those for other tasks are based on the 128^2 resolution models.

References

1. Albright, M., McCloskey, S.: Source generator attribution via inversion. In: CVPR Workshops (2019)

2. Bau, D., Zhu, J.Y., Wulff, J., Peebles, W., Strobelt, H., Zhou, B., Torralba, A.: Seeing what a gan cannot generate. In: ICCV. pp. 4502–4511 (2019)
3. Brock, A., Donahue, J., Simonyan, K.: Large scale gan training for high fidelity natural image synthesis. In: ICLR (2019)
4. Creswell, A., Bharath, A.A.: Inverting the generator of a generative adversarial network. In: IEEE transactions on neural networks and learning systems (2018)
5. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: ICML. pp. 448–456 (2015)
6. Johnson, J., Alahi, A., Fei-Fei, L.: Perceptual losses for real-time style transfer and super-resolution. In: ECCV. pp. 694–711. Springer (2016)
7. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
8. Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., Lerer, A.: Automatic differentiation in pytorch (2017)
9. Zhu, J.Y., Krähenbühl, P., Shechtman, E., Efros, A.A.: Generative visual manipulation on the natural image manifold. In: ECCV (2016)