Supplementary material for "Powers of layers for image-to-image translation"

Anonymous Author(s)

In this supplementary material we report additional analyses, results and examples that complement our paper. In Appendix A we consider the training phase, which supports the importance of our progressive training strategy compared to one with a fixed number of iterations. Appendix B considers the inference-time choices, in particular possible strategies to select the number of iterations. Appendix C provides additional comparison to CycleGAN and makes a comparison with the Fader network. Finally we present additional visual results for high resolution images and illustrate the progressive evaluation of results along iterations in the appendix D.

A Analysis of our progressive training strategy

Figure 9 shows the different training strategies we explore. The degree of freedom that we can adjust is the number of compositions. It can be set independently per training mini-batch.

Progressive training versus fixed training. Figure 10 illustrates the modulation of the horse→zebra transformation. This is effective only with our progressive learning, which forces the network to produce acceptable intermediate states.

If the network is trained with a fixed number of compositions, the intermediate states do not correspond to modulations of the transformation. The output is satisfactory only when the $n_{\text{te}} = n_{\text{tr}}$. In all other cases we observe artifacts in images, which therefore do not qualify as natural images. The generated images do not look right, and the source and target discriminators would not accept them as real images.

In contrast, with our progressive training, each number of iterations produces a satisfactory output. Iterating the residual block gradually transforms the horse into a zebra.

Warm-up phase. Figure 11 compares the performance obtained during the first three epochs of learning with and without progressive training, with different maximum number of compositions n_{tr}. We compare this way of stabilizing the training with another classical approach: reducing the ranges of initialization of the residual blocks. Figure 11 shows that changing the initialization improves the performance during the first epoch, but that a warm-up phase with progressive training

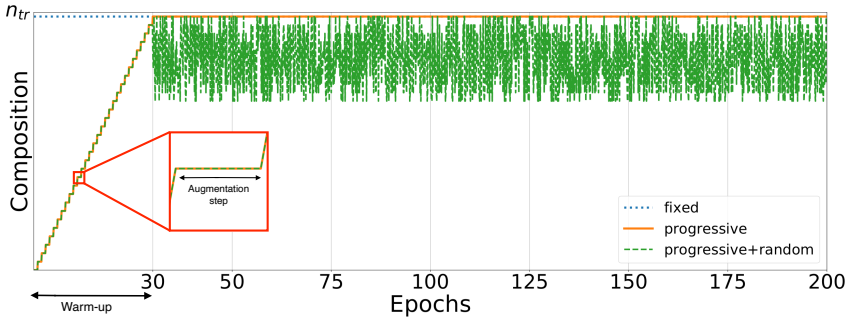

Figure 9: Number of compositions at training time for different training strategies. The number of compositions is adjusted per batch.

Table 2: PSNR on Urban-100. Comparison between our choice (PoL) and independent blocks.

Number of blocks	Gaussian noise (std 30) POL	Gaussian noise (std 30) independent	Gaussian blur (sigma 4) POL	Gaussian blur (sigma 4) independent
1	23.26 ±0.15	23.26 ±0.15	18.61 ±0.13	18.61 ±0.13
2	23.28 ±0.07	23.21 ±0.05	18.48 ±0.25	18.64 ±0.21
4	24.41 ±0.27	23.16 ±0.09	19.19 ±0.04	19.17 ±0.51
8	23.91 ±0.23	22.27 ±0.02	18.95 ±0.14	19.33 ±0.29
12	23.91 ±0.10	22.48 ±0.32	19.70 ±0.21	18.76 ±0.09
16	23.88 ±0.14	22.54 ±0.48	19.00 ±0.30	18.11 ±0.41

is more effective to improve the optimization stability. Figure 9 shows the evolution of the number of compositions for different training strategies.

Block composition or independent successive blocks? Table 2 provides results that complement Table 1 in the main paper. It compares the performance with (1) the composition of the same block or (2) using independent residual blocks. In particular, we report standard deviations that assess the statistical significance of our improvement.

Choice of the maximum number of compositions. Table 3 compares the PSNR obtained for denoising and deblurring and NIQE for JPEG deblocking task on Urban-100 Huang et al. (2015). We report more results and standard deviations compared to the main paper. Note that these tasks work well with a relatively low maximum of iterations, in contrast to style transfer image-to-image translations, which require more complex functions.

Composition step. Table 4 compares different choices for the number of steps of augmentation associated with the number of compositions n_{tr}. As we can see, taking too large steps tends to affect performance, it is better to ramp up the number of compositions quickly during the warm-up phase.

Comparison between randomised and fixed number of compositions n_{tr}. Figure 12 compares the trajectories of PSNRs as a function of the number of Powers-of-layers composition. We get a better average performance if we randomly draw the maximum number of composition. The different positions of the maxima in the adaptive case also suggests that it is necessary to adjust the amount of transformation to each image, as discussed below.

B Analysis of choices at inference time

Stopping criterion: fixed, discriminator, versus an oracle. At inference time the number of composition n_{te} applied to each image can be set using several strategies. We can choose to apply a

Figure 10: Comparison between our progressive training approach and a non progressive approach. We represent the images obtained by varying the number of iterations at inference time n_{te} in the network that transforms from domain A (horse) into domain B (zebra). The first image (n_{te}=0) is the original image. Since our method was learned with n_{tr}=30 compositions the last two images are extrapolations. Our progressive training is key to ensure that all outputs look like natural images and therefore that we can modulate transformation strength at inference time.

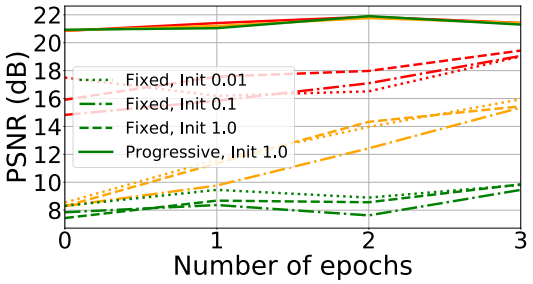

Figure 11: Difference in PSNR between fixed learning and progressive learning during the first training epochs, evaluated on a denoising task.
Green: n_{tr}=30
orange: n_{tr}=16
red: n_{tr}=4

Table 3: Comparison between different maximum number of compositions. We report the most adapted metric on Urban-100: PSNR for the Gaussian noise and blur, NIQE for JPEG deblocking.

n_{tr}	PSNR (higher=better) Gaussian noise (std 30)	PSNR (higher=better) Gaussian blur (σ=4)	NIQE (lower=better) JPEG (quality=25)
1	23.26 ±0.15	18.61 ±0.13	10.17 ±0.55
2	23.28 ±0.07	18.48 ±0.25	10.78 ±0.39
3	23.89 ±0.07	19.13 ±0.08	10.57 ±0.21
4	**24.41** ±0.27	19.19 ±0.04	10.43 ±0.25
5	23.79 ±0.59	19.06 ±0.21	10.65 ±0.52
6	23.80 ±0.31	19.09 ±0.08	10.42 ±0.19
7	23.64 ±0.27	19.12 ±0.13	10.93 ±0.61
8	23.91 ±0.23	18.95 ±0.14	10.34 ±0.61
12	23.91 ±0.10	**19.70** ±0.21	9.74 ±0.41
16	23.88 ±0.14	19.00 ±0.30	8.51 ±0.19
17	23.97 ±0.17	18.83 ±0.21	8.36 ±0.32
18	24.17 ±0.18	18.97 ±0.13	7.49 ±0.63
24	23.83 ±0.20	18.56 ±0.16	**7.22** ±0.45
27	23.62 ±0.02	18.48 ±0.36	7.25 ±0.36
30	23.45 ±0.10	19.09 ±0.31	8.15 ±0.64

Table 4: PSNR on Urban-100 Huang et al. (2015) – Gaussian noise (std=30). Comparison between augmentation steps during the warm-up phase. The augmentation step is to the number of epochs performed with the same number of compositions (1 epoch corresponds to 800 backward passes).

n_{tr}	augmentation step 1	2	4	8	16
4	24.41 ±0.27	24.44 ±0.11	23.62 ±0.16	23.93 ±0.10	23.22 ±0.07
16	23.88 ±0.14	23.97 ±0.22	23.57 ±0.06	23.52 ±0.1	23.19 ±0.02
30	23.45 ±0.10	23.77 ±0.14	23.95 ±0.17	23.14 ±0.04	23.17 ±0.04

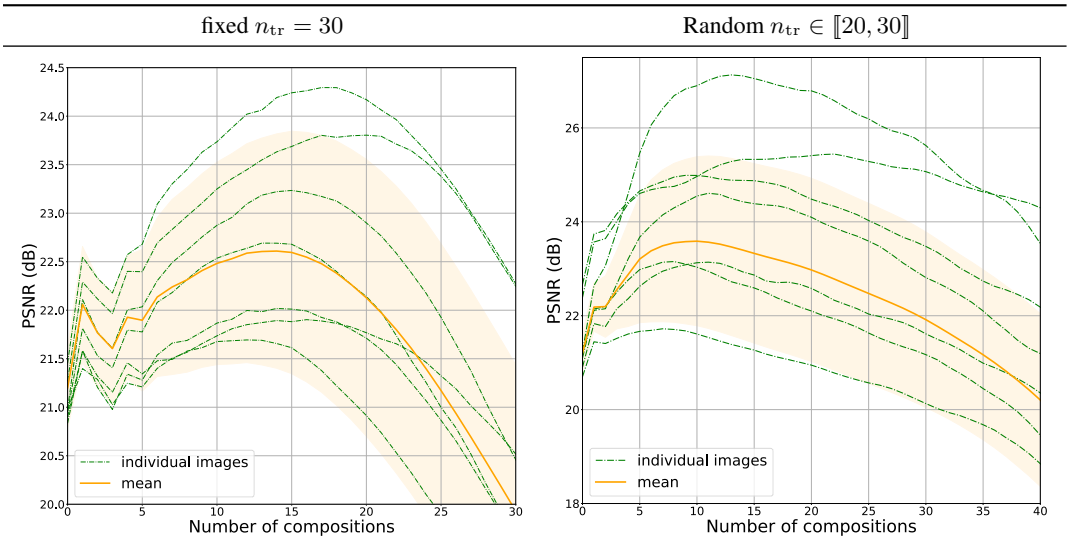

Figure 12: Evolution of the average PSNR for different individual images according to the number of compositions n_{te}. The maximum number of compositions used for training is $n_{\text{tr}} = 30$ and the Gaussian Noise standard deviation is 30.

Table 5: Effect of setting the number of compositions n_{tr} randomly on the PSNR on Urban-100 with two types of noise. We compare (Constant) a fixed $n_{\text{te}} = 30$ and (Adaptive) value n_{te} maximizing the target discriminator error for each image. Oracle: n_{te} minimizing PSNR for each image.

	Gaussian Noise (std=30)			Gaussian Blur (σ=4)		
Random range	Constant	Adaptive	Oracle	Constant	Adaptive	Oracle
$[\![0, 30]\!]$	21.41 ±3.33	22.11 ±0.56	23.86 ±0.27	16.95 ±1.29	16.16 ±1.33	19.43 ±0.09
$[\![7, 30]\!]$	22.56 ±1.94	22.84 ±0.67	23.73 ±0.98	16.39 ±2.00	15.42 ±2.26	19.58 ±0.14
$[\![10, 30]\!]$	21.99 ±1.20	23.14 ±0.21	23.42 ±0.32	17.48 ±1.56	18.10 ±0.70	**19.65** ±0.10
$[\![15, 30]\!]$	21.41 ±1.71	23.47 ±0.69	23.77 ±0.61	18.14 ±0.54	18.81 ±0.61	19.42 ±0.07
$[\![20, 30]\!]$	21.53 ±1.90	**23.68** ±0.17	**23.99** ±0.06	18.58 ±0.10	**19.22** ±0.37	19.56 ±0.09
$[\![22, 30]\!]$	21.16 ±0.99	23.08 ±0.37	23.42 ±0.28	17.55 ±1.32	19.06 ±0.39	19.52 ±0.05
$[\![30, 30]\!]$	**23.45** ±0.10	23.46 ±0.40	23.81 ±0.48	**19.09** ±0.31	18.87 ±0.41	19.49 ±0.08

constant number of composition or use the discriminator to choose the n_{te} for which it gets the best response.

Table 5 compares the performance of two strategies at test time for different random ranges at training time, and compare it to the upper bound achieved by an oracle (i.e,. the performance attained when the optimal number of iteration is known for each image).

With n_{tr} =30, we have chosen different ranges of the form $[\![d \times 30, 30]\!]$ for $d \in \{100\%, 75\%, 66\%, 50\%, 33\%, 25\%, 0\%\}$. The optimal range for debluring and denoising is with $d = 66\%$

C Additional comparisons with CycleGAN and the Fader Network

Table 6 compares the results obtained with PoL and CycleGAN for different noise levels. Table 7 compares the results obtained with PoL and CycleGAN for different amounts of data. These numbers are the same as Figure 3, with standard deviations. In most cases, whether with different amounts of data or different noise, our method is better than CycleGAN. This is mainly due to its smaller number of parameters and the flexibility brought by the adaptive criterion.

Table 6: Comparison between our approach (PoL) and CycleGAN. We three tasks, all computed with the Urban-100 dataset: PSNR (higher is better) with different amount of Gaussian noise and Gaussian blur, and NIQE (lower is better) measured for different JPEG compression quality.

Noise (std)	Noisy images	Denoising CycleGAN	Denoising PoL	Sigma Blur	Blur images	Debluring CycleGAN	Debluring PoL	JPEG quality	JPEG images	Deblocking CycleGAN	Deblocking PoL
15	24.9	22.37 ±0.19	**27.37** ±0.26	2	21.58	20.37 ±0.26	**22.14** ±0.21	15	9.01	7.89 ±1.14	7.90 ±0.32
30	19.2	21.93 ±0.04	**23.68** ±0.17	4	19.20	18.55 ±0.37	**19.22** ±0.37	25	8.94	7.45 ±0.63	**7.10** ±0.58
50	15.2	21.57 ±0.04	**22.52** ±0.37	8	17.48	16.09 ±0.14	**17.53** ±0.34	30	8.90	7.46 ±0.32	**6.76** ±0.83
70	12.7	**21.02** ±0.17	20.94 ±0.24	16	16.13	16.11 ±0.31	**16.16** ±0.14	50	8.99	6.96 ±0.55	**6.56** ±0.48
100	10.4	**20.00** ±0.12	19.42 ±0.22	24	15.50	12.88 ±0.24	**15.48** ±0.07	70	8.94	7.11 ±0.21	**6.81** ±0.39

Table 7: Comparison between CycleGAN and Powers-of-layers on Urban-100 (Huang et al., 2015) with different amount of training data. We use PSNR to compare methods for Gaussian noise and Gaussian blur.

Number of data training images	Denoising (std=30) CycleGAN	Denoising (std=30) PoL	Debluring (sigma=4) CycleGAN	Debluring (sigma=4) PoL
1	14.50 ±0.24	**22.27** ±0.16	14.36 ±0.12	**18.68** ±0.27
5	16.57 ±0.04	**23.13** ±0.18	16.72 ±0.15	**18.76** ±0.23
10	20.88 ±0.36	**23.19** ±0.58	17.03 ±0.07	**18.82** ±0.56
100	21.03 ±0.76	**23.39** ±0.33	18.17 ±0.15	**18.97** ±0.23
400	21.78 ±0.12	**23.60** ±0.32	18.21 ±0.11	**19.01** ±0.31
800	21.93 ±0.04	**23.88** ±0.14	18.55 ±0.37	**19.22** ±0.37

Experiments on transformation adjustment As baseline we use the Fader network Lample et al. (2017) for transformation adjustment. The Fader Network is a neural network composed of an encoder and a decoder, for which it it is possible to modulate a transformation. This is done by removing the factors of variations related to this transformation in the latent space resulting from the encoder, and in turn by choosing the factors to be added to the embedding going into the decoder.

To interpolate between domain $\mathcal{A}$ and domain $\mathcal{B}$, the Fader network has a latent representation where the attributes relative to each domain have been disentangled. The Fader network has been applied to faces, for instance to add glasses on a face, to age a person, etc. We observe experimentally with smaller datasets, where the variability from one image to another is larger than with faces, that the Fader's results are not as good. In contrast, our approach, like CycleGAN, does not have limitations incurred by a latent space disentanglement because it exploits a cyclic loss.

Figure 13 shows the results obtained with the Fader network and with our method on the Horse→Zebra transformation adjustment. The Fader network is unable to significantly transform the source when the network is too shallow (3 layers) and destroys the image when it is deep (6 layers). In contrast, Powers-of-layers convincingly hybridizes a horse and a zebra. In terms of FID for the Horse to Zebra task, the Fader network is significantly worse: it obtains a FID greater than 163.0 in the both case against 53.0 for our method (lower is better).

D Additional results: visualizations of transform modulation and high resolution

Progressive results

Results in high resolution. Figure 14 shows the high-resolution results of the section 5, along with the original images.

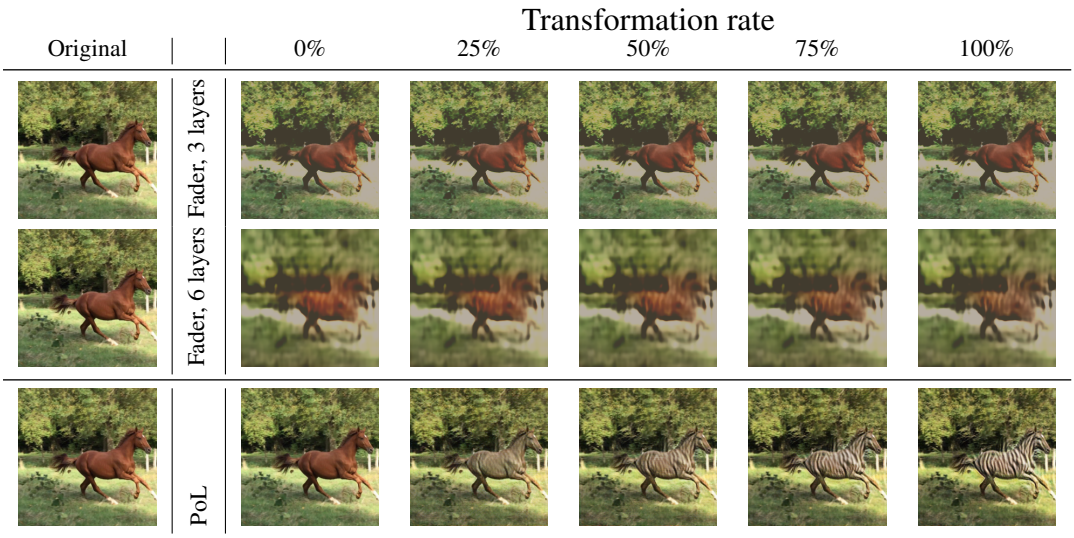

Figure 13: Visual comparison between Fader networks Lample et al. (2017) and our power-of-layers on the task Horse to Zebra.

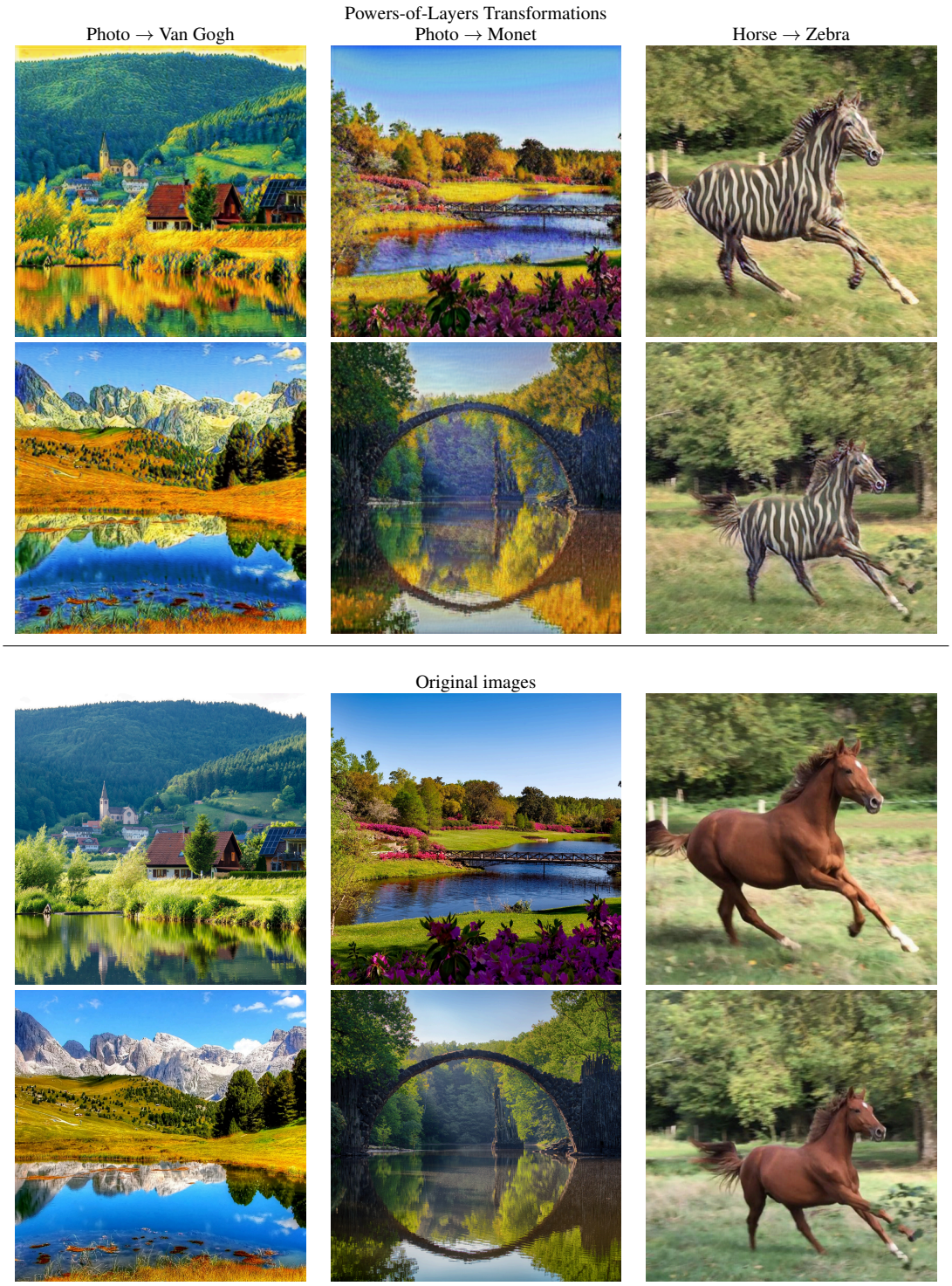

Figure 14: *Top:* Differents visuals results with high resolution image. *Bottom:* Original images