SinGAN: Learning a Generative Model from a Single Natural Image Supplementary Material

Tamar Rott Shaham
Technion

Tali Dekel
Google Research

Tomer Michaeli
Technion

Contents

1. Optimization

At each scale, the weights of the generator and discriminator are initialized to those from the previous trained scale (except for the coarsest scale or when changing the number of kernels, in which cases we use random initialization). We train each scale for 2000 iterations. In each iteration we alternate between 3 gradient steps for the generator and 3 gradient steps for the discriminator. We use the Adam optimizer [2] with a learning rate of 0.0005 (decreased by a factor of 0.1 after 1600 iterations) and momentum parameters $\beta_1 = 0.5, \beta_2 = 0.999$. The weight of the reconstruction loss is $\alpha = 10$, and the weight of the gradient penalty in the WGAN loss is 0.1. All LeakyReLU activations have a slope of 0.2 for negative values. The generator's last conv-block activation at each scale is Tanh instead of ReLU. The descriminator's last conv-block at each scale includes neither normalization nor activation. Training takes about 30 minutes on a 1080TI GPU for an image of size 256×256 pixels, and the generation at test time is well under one second per image.

2. Boundary conditions and the effect of padding

The type of padding used within SinGAN's generators highly influences the diversity among generated samples at the corners (the net learns to associate the image structures at the corners with the artificial edges caused by the padding). Figure 1 illustrates this effect by depicting the standard deviation among generated samples for each location in the image. As can be seen, zero padding in each layer leads to very small variability at the corners. This effect is reduced when, instead of padding within the layers, we zero-pad the inputs to the generator (both image and noise) by half the net's receptive field from each side. This is our default setting for generating samples. Finally, if the input image is zero-padded this way while the input noise is padded by noise (*i.e.* taken to be larger), then the effect is further reduced. We use this configuration for generating animations.

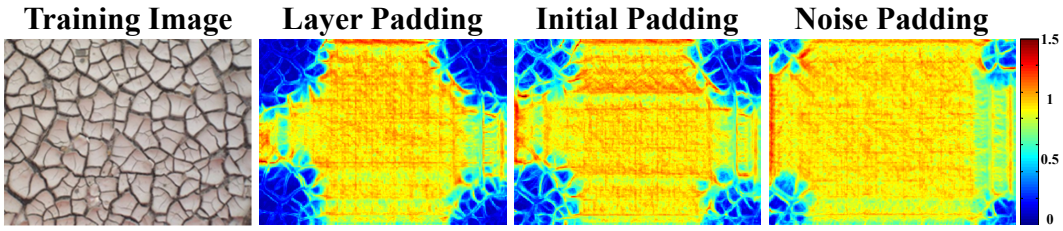


Figure 1: **Effect of padding.** The padding configuration highly effects the diversity between samples at the corners. Zero padding in each convolutional layer (layer padding) leads to only minor variability at the corners. Padding only the input to the net (initial padding), leads to somewhat increased variability at the corners. Finally, padding by noise, leads to high variability.

3. Failure cases

As stated in the main text, SinGAN may produce unrealistic results when the global structure of the image is of high importance. However, we can avoid this by starting the generation process at a finer scale. Figure 2 demonstrates this for face images. In this case, starting the generation at the coarsest scale results in extremely distorted images. However starting the generation at a finer scale, we can preserve the face's global structure while altering only finer details, like the shapes of eyes, nose and lips, texture of the skin and eyebrows, etc.

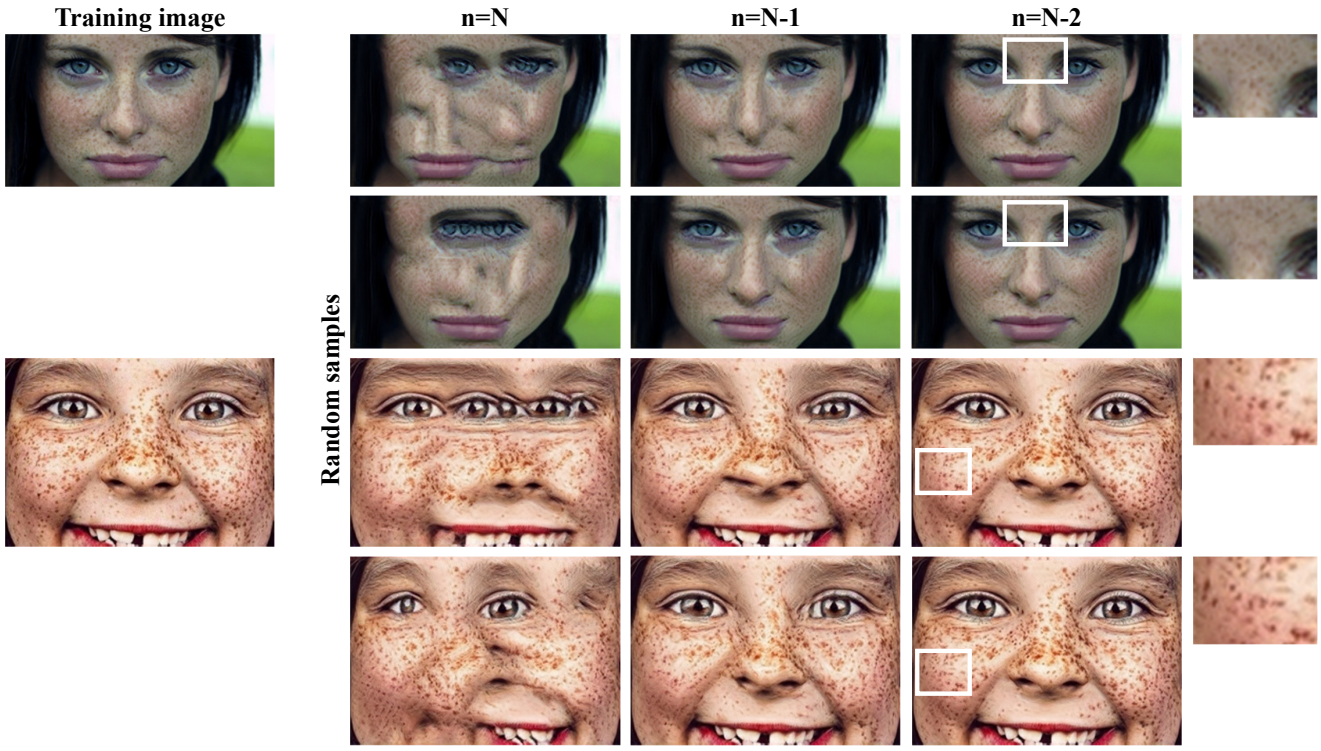


Figure 2: **SinGAN for face images.** Thanks to the multi-scale architecture, generation is possible also in extreme cases like face images. In this case, when starting the generation at the coarsest scale (n=N), SinGAN produceŒŞ unrealistically distorted face images. However, starting the generation at a finer scale ($n = N - 1$, $n = N - 2$), the global structure of the face is preserved, and only finer details are generated.

4. Effect of the injection scale for image manipulation

In all image manipulation tasks illustrated in the paper, we inject (a possibly downsampled version of) an image into the generation pyramid at some scale $n < N$, and feed forward it through the generators up to the finest scale. This paradigm is illustrated in Fig. 3, where a SinGAN trained on a single Zebra image, adds stripes to a horse injected at a coarse scale. Different injection scales lead to different effects. We demonstrate this in Fig. 4 in the context of harmonization, paint-to-image, and image editing. As can be seen, the coarser the injection scale, the larger the structures that SinGAN can modify so as to match the statistics of patches in the generated image to the training image. Injection at fine scales, only modifies small scale structures and textures, while leaving the global structure intact.

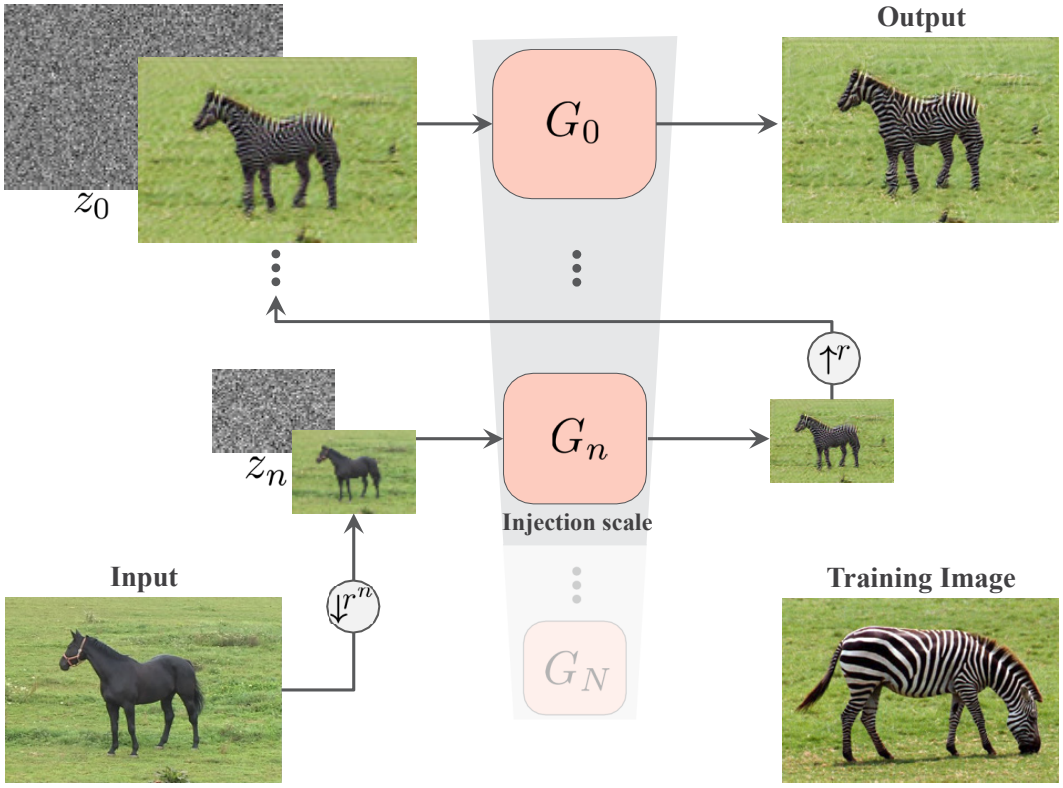


Figure 3: **Image manipulation using SinGAN.** Once a SinGAN model is trained, we can inject an image to one of its coarser scales so as to match its patch distribution to that of the training image. In this example, we feed a downsampled version of a horse image (bottom left) into the 7th scale of a 10-scale SinGAN model that was trained on a Zebra image (bottom right). As the this input propagates through the generation pyramid, stripes are gradually added to the horse, which grant it the appearance of a Zebra.

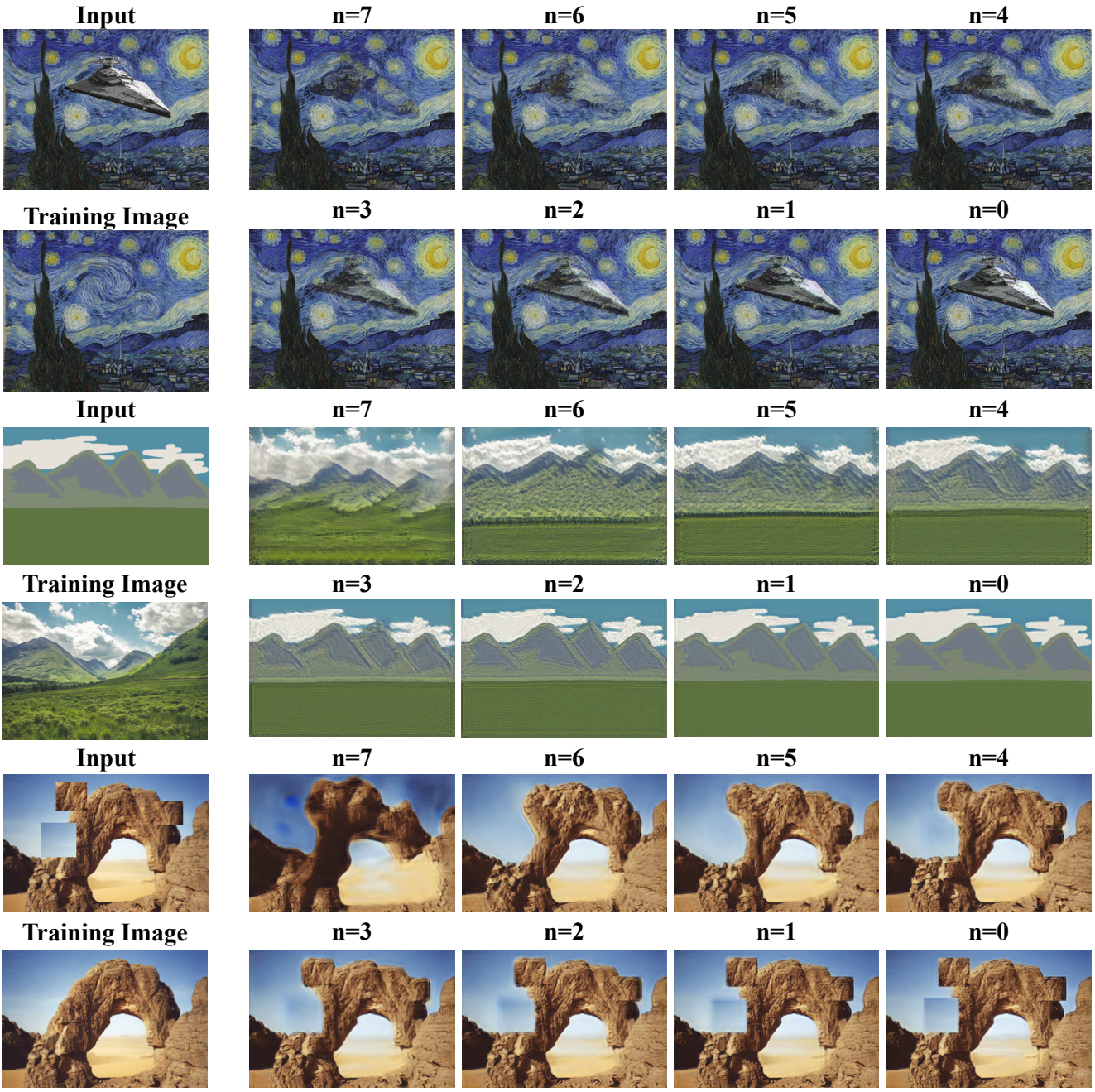


Figure 4: **Effect of the injection scale for image manipulation.** In applications such as harmonization (top), paint-to-image (middle), and editing (bottom), the coarser the scale we inject the input image, the larger the structures that get modified. When the input is injected at fine scales, only the fine textures get modified while the global structure remains fixed.

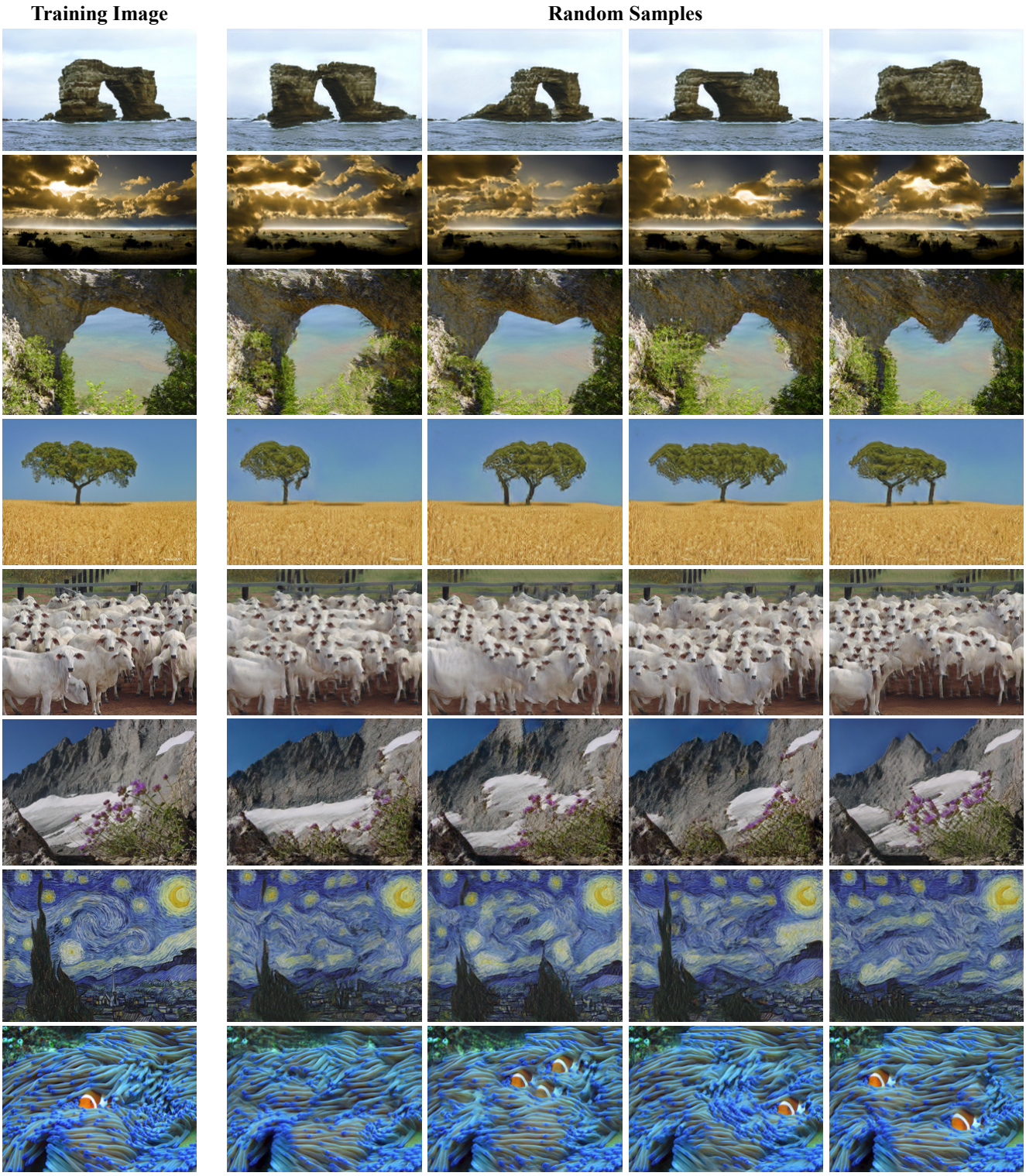
5. Animation from a single image

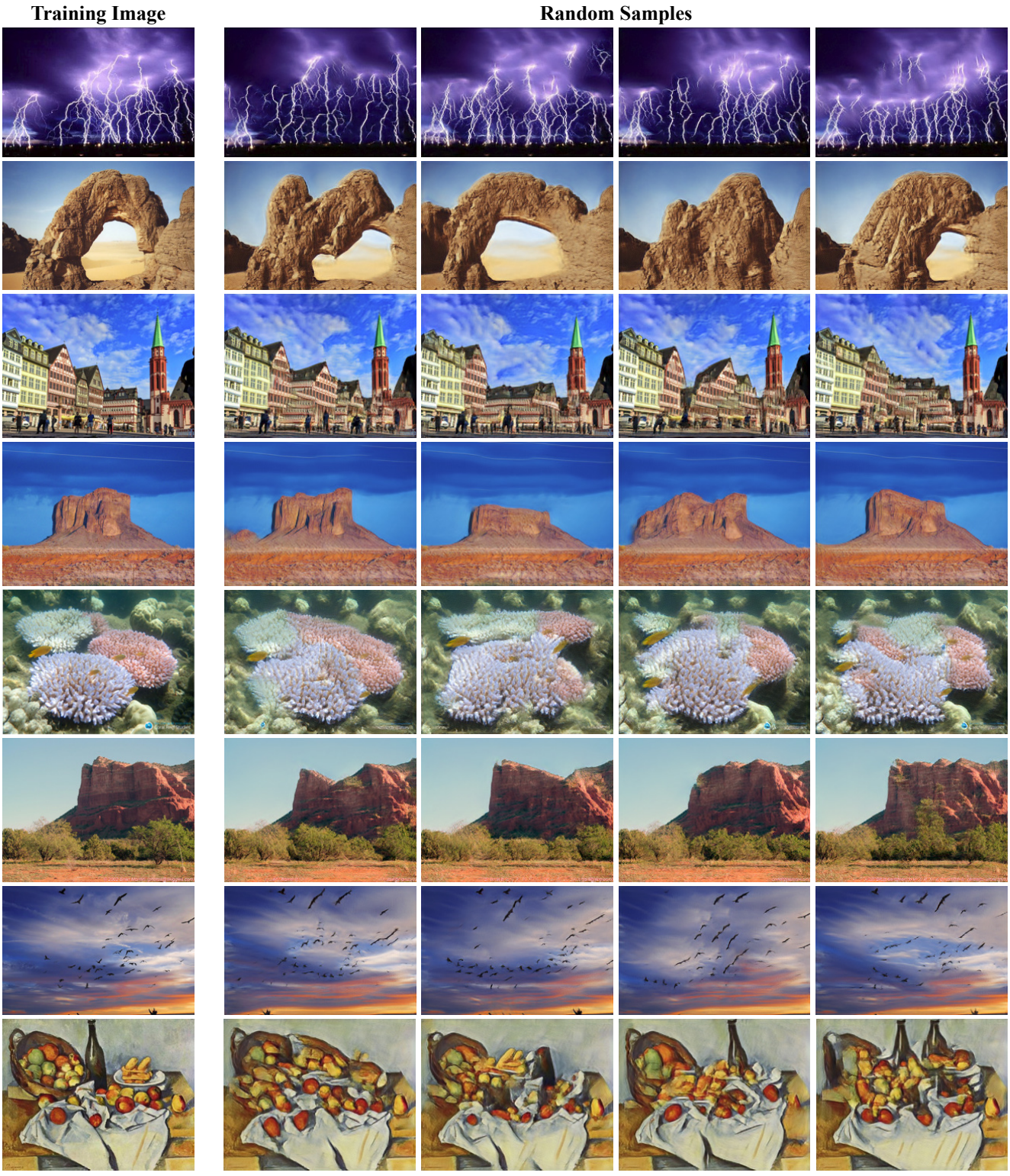
To animate a single image x, we introduce smooth random changes to SinGAN's noise maps, while restricting them to remain close to the noise maps that generate x. Those perturbations produce images with similar layout, and thus create a sequence of images with gradually changed composition. Specifically, denoting by $z_n(t)$ the nth scale noise map at time t, we construct our random walk as

$$\begin{aligned} z_n(t+1) &= \alpha z_n^{rec} + (1-\alpha)\left(z_n(t) + z_n^{\text{diff}}(t+1)\right), \\ z_n^{\text{diff}}(t+1) &= \beta\left(z_n(t) - z_n(t-1)\right) + (1-\beta)u_n(t) \end{aligned}$$

where $z_n(0) = z_n^{\text{rec}}$ and $u_n(t)$ is a an iid sequence of noise maps having the same dimensions as $z_n(t)$. Here, α determines how close the frames of the sequence remain to x, whereas β controls the smoothness and rate of change in the generated clip. Note that similarly to the other image manipulation tasks, here as well we can start the animation from different scales so as to obtain different effects. Please see the supplementary video for animation results.

6. Random Samples




Training Image
Random Samples

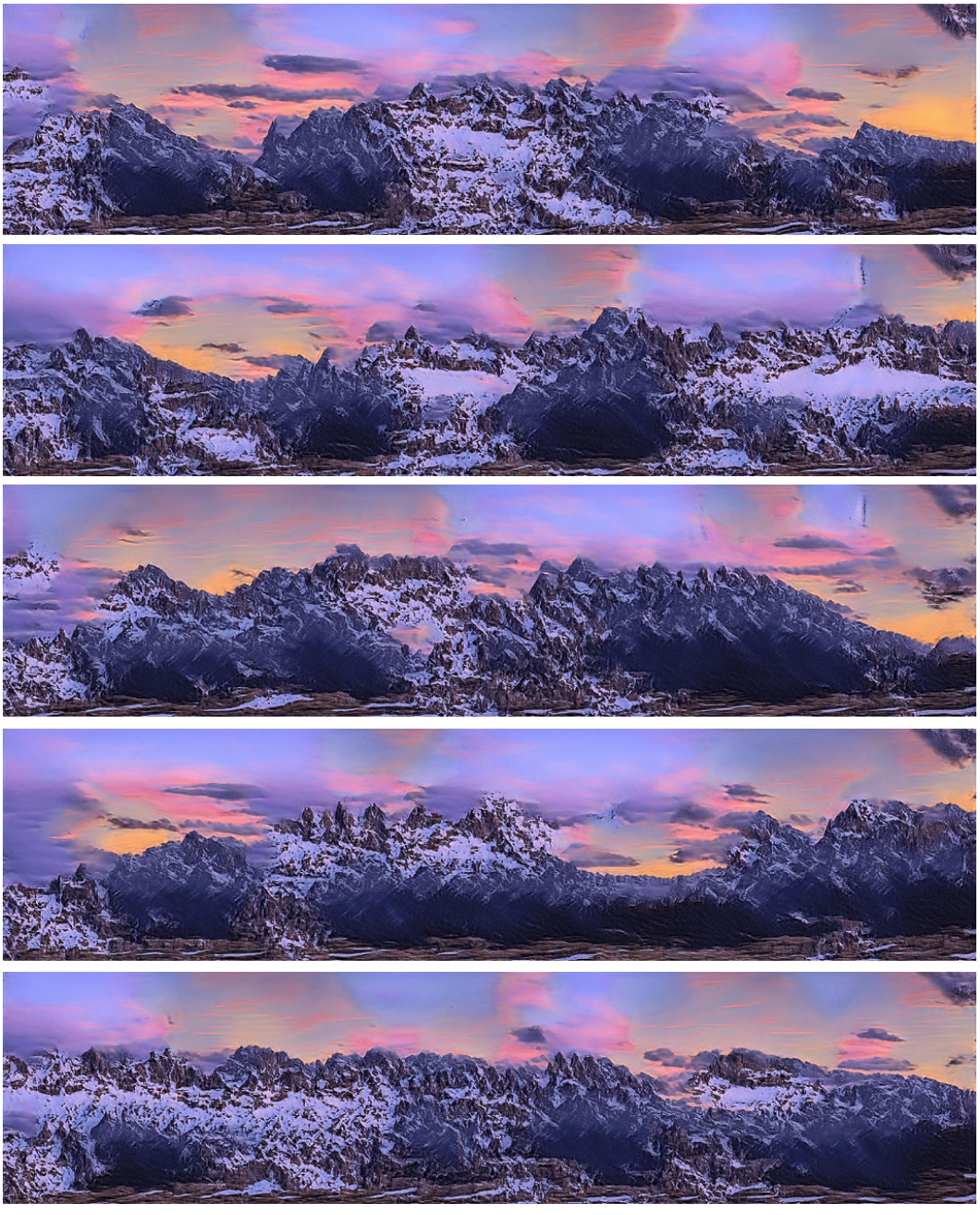
7. High Resolution Generation

SinGAN's architecture is resolution agnostic and thus can handle images of arbitrarily high resolution. The larger an image, the more scales we have, and each scale is only responsible for the finest details not accounted for by the previous scale. Therefore, putting aside computation considerations, there is no resolution limit. Bellow are a few examples for random samples generated by SinGAN models trained on 0.25Mpix images.

Generation of higher resolution samples does not necessarily require directly training SinGAN on image of the same resolution. Instead, we can first train SinGAN on a lower resolution image, and then use our SR scheme to achieve the desired final resolution (see Section 4 in the main paper). With this approach, we reduce both training time and memory requirements and still achieve high quality generation results. Figure 5 shows 4Mpix random samples generated by SinGAN when trained on a 0.25Mpix image. As can be seen, the new samples enhance the training image and contain new realistic global structures and fine details.

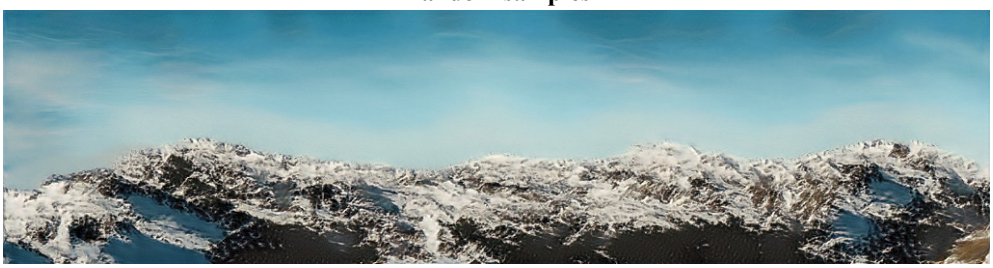
Training Image

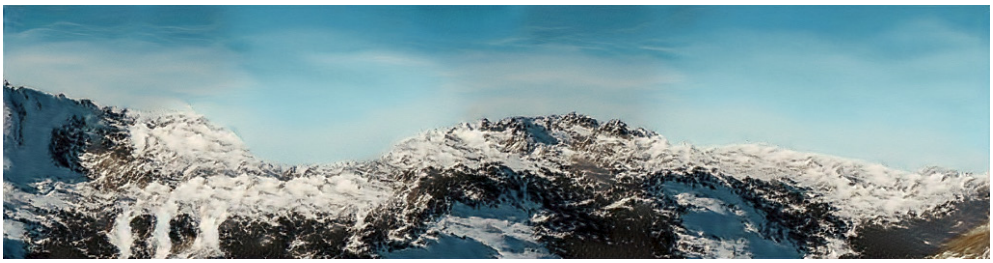
Random samples

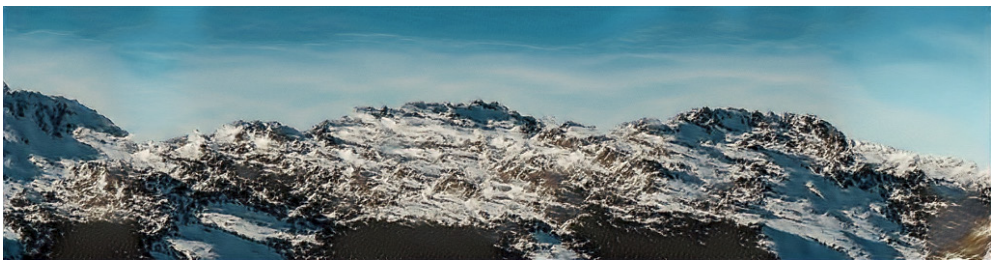


Training Image

Random samples



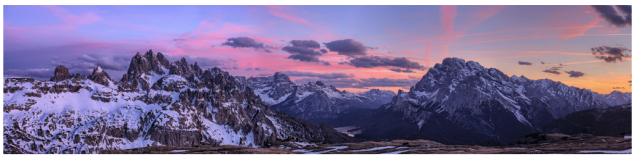




Training Image

Random samples

Training image



Random samples + SR

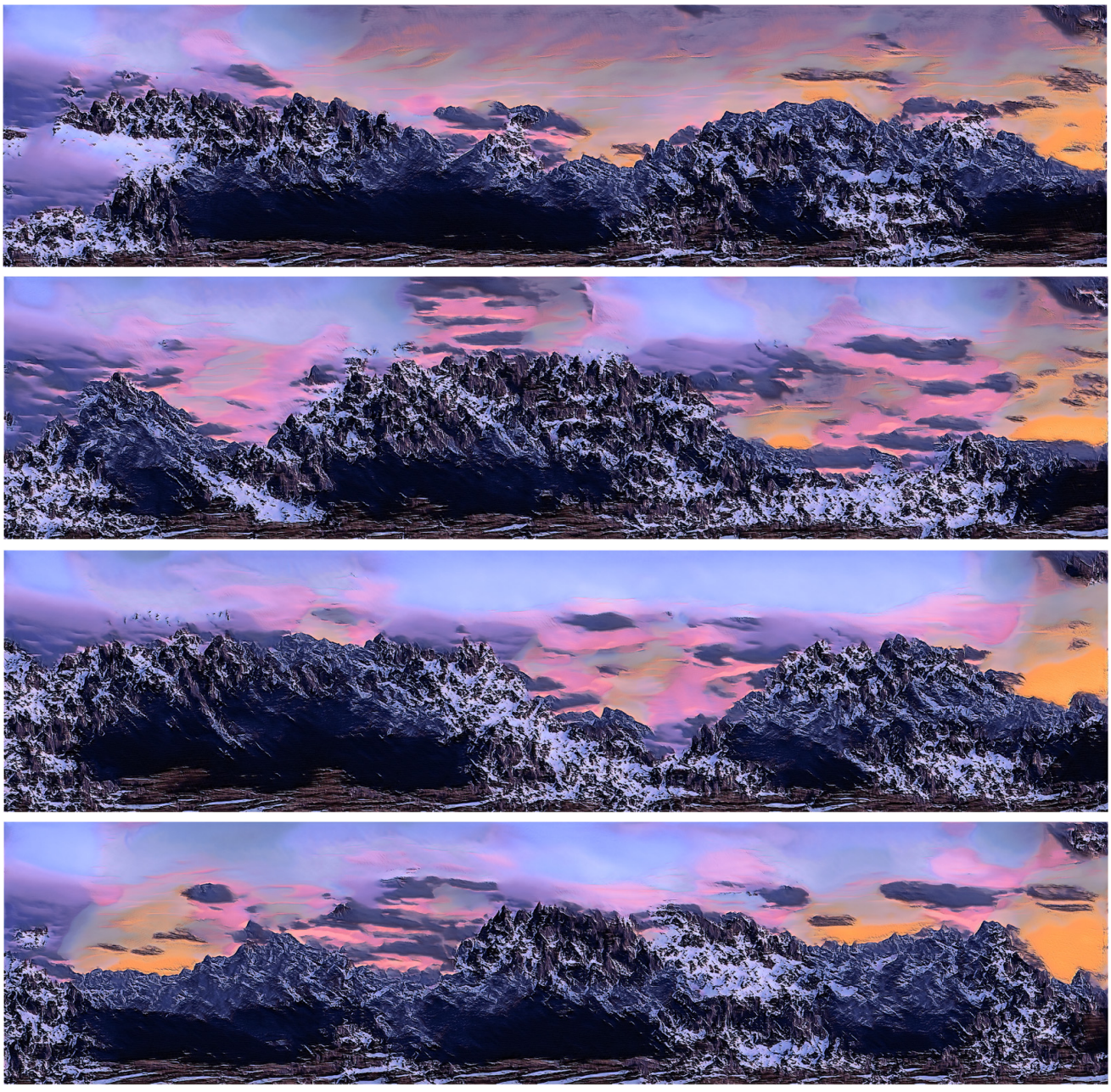
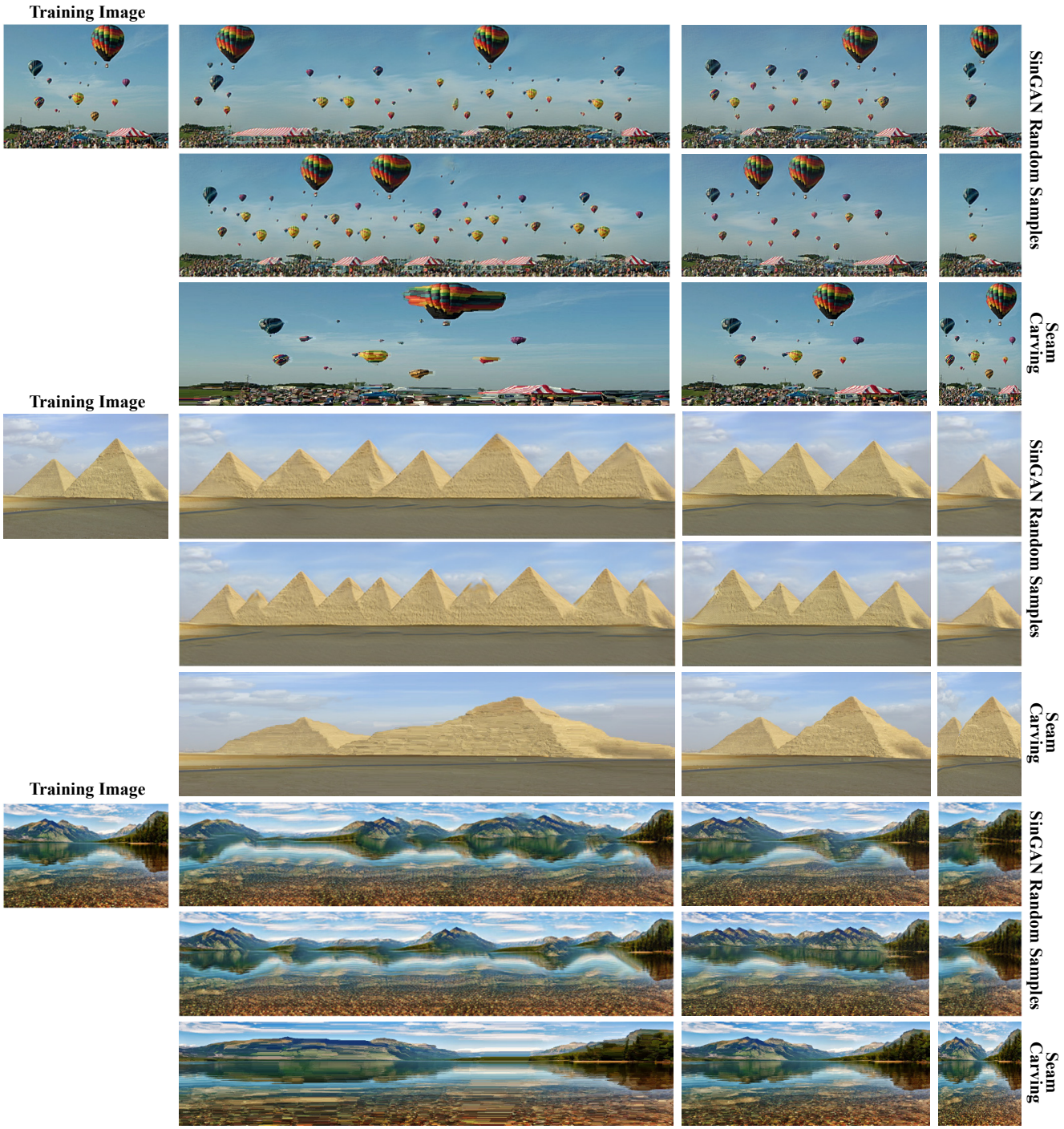
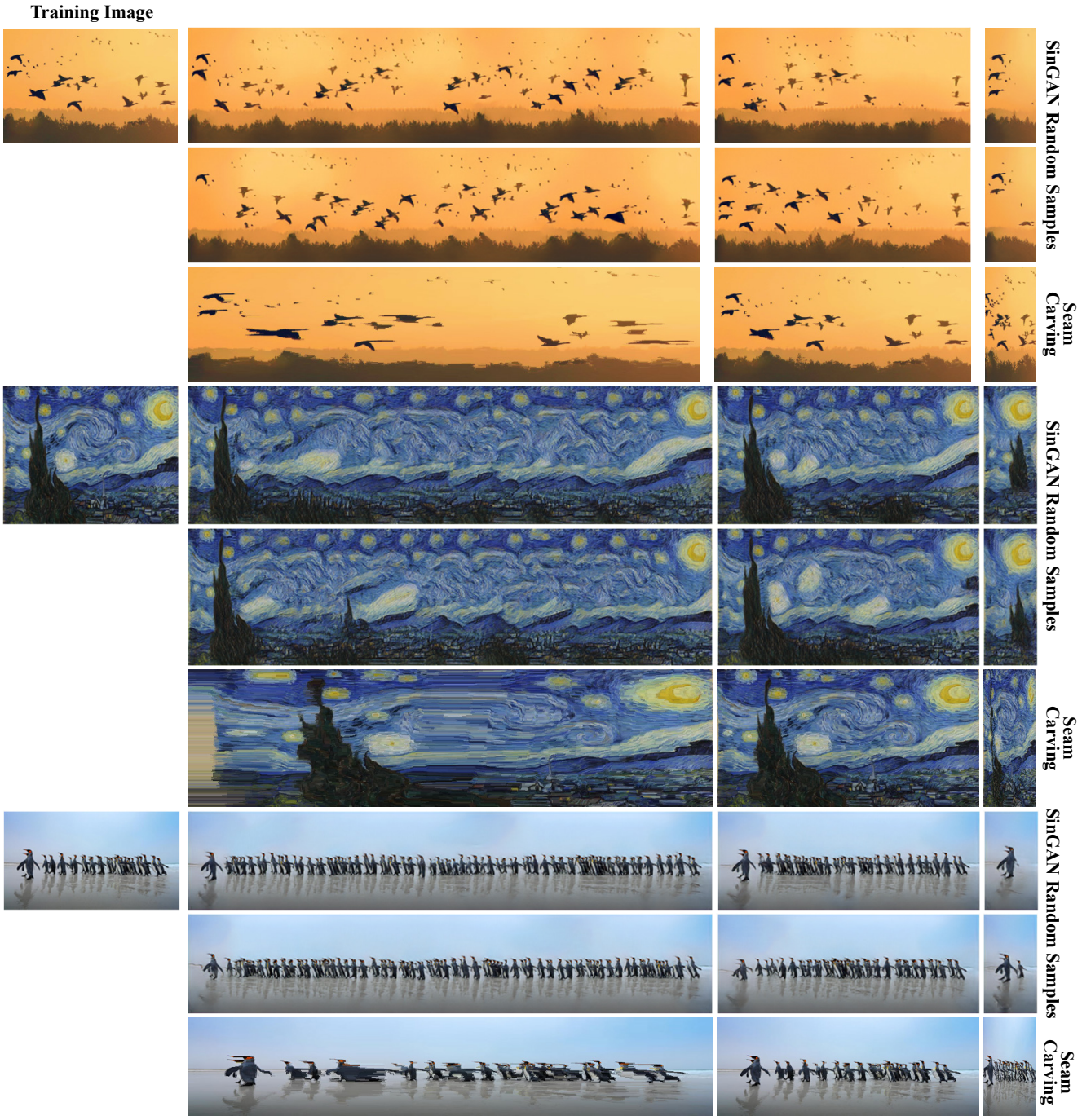


Figure 5: **Random 4Mpix samples generated by SinGAN.** These are achieved by training SinGAN on a 0.25Mpix image, generate random samples and then applying our SR scheme. Our model generates high quality, realistic samples by only observing image content at 1/16 of the output resolution during training.

8. Samples at arbitrary dimensions

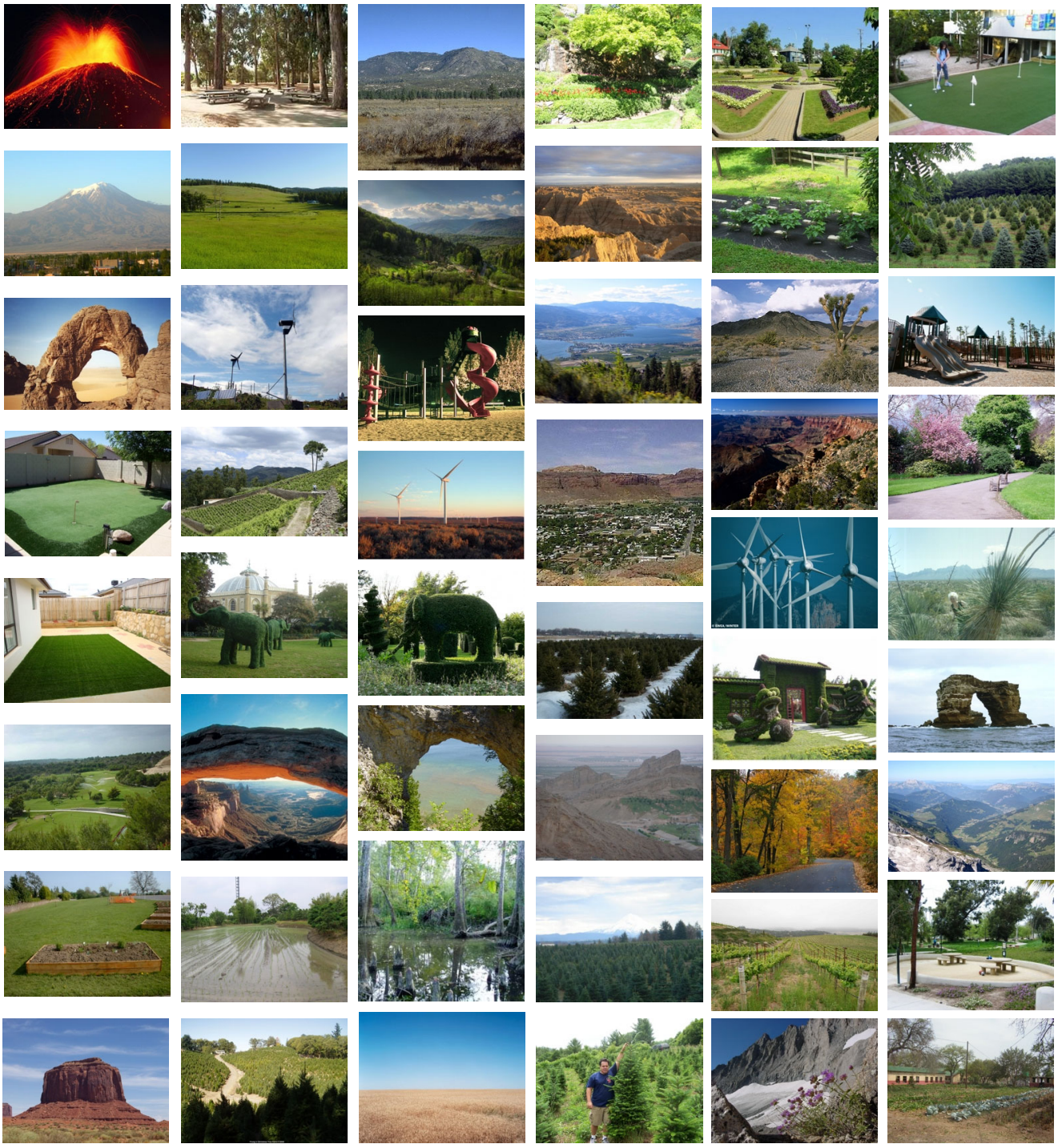
Using SinGAN, we can generate samples of arbitrary size and aspect ratio. While this is superficially similar to retargeting, the two tasks are distinct. We illustrate this below through comparison to seam carving [1]. In our case, the samples are *random* and are not optimized to maintain salient objects or strong edges. This often leads to plausible configurations that do not appear in the input image. In image retargeting, on the other hand, structures are typically restricted to those appearing in the image, and may deform unrealistically at extreme aspect ratios.



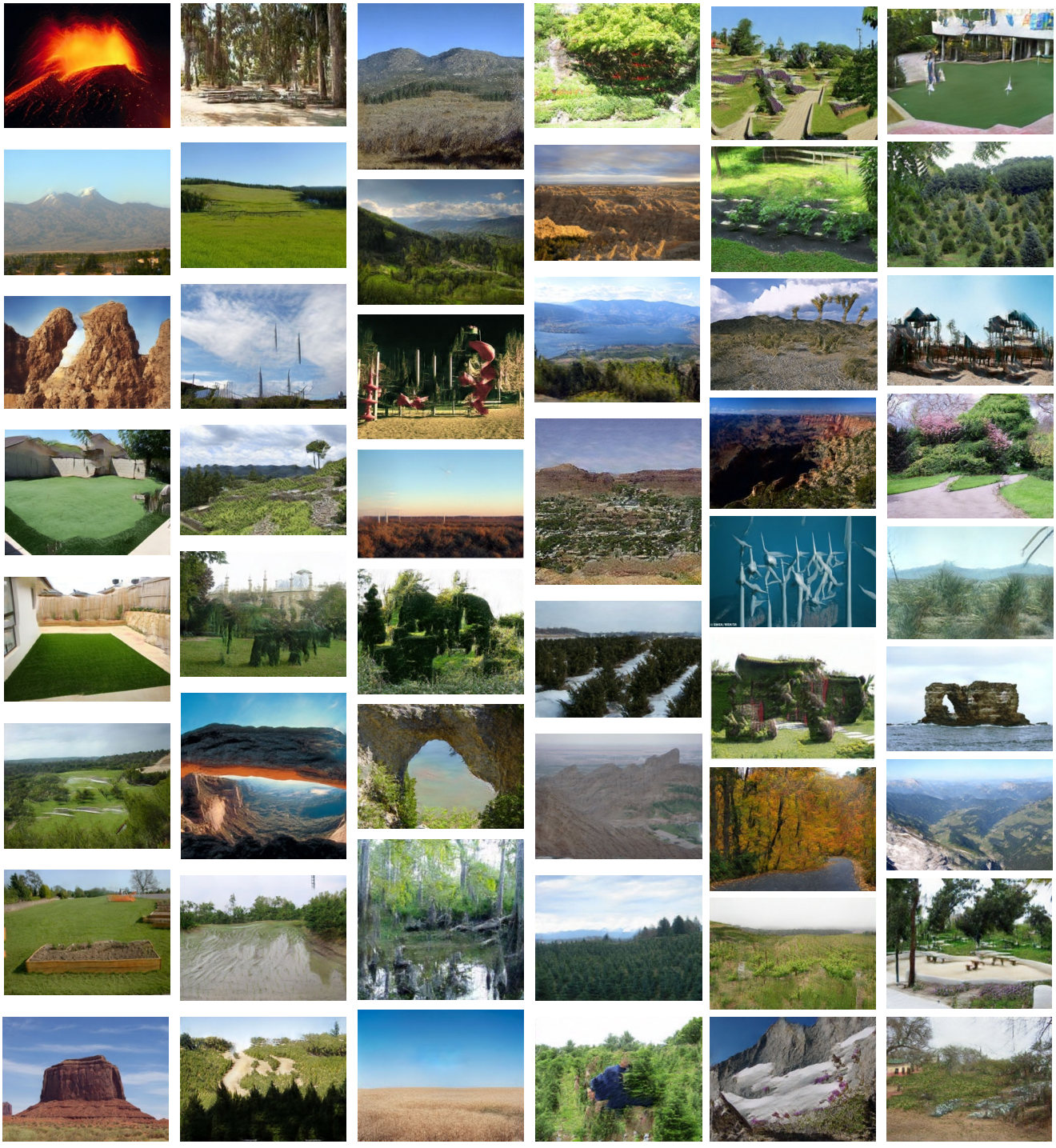

Training Image
SinGAN Random Samples
Seam Carving
SinGAN Random Samples
Seam Carving
SinGAN Random Samples
Seam Carving

9. AMT Survey Images

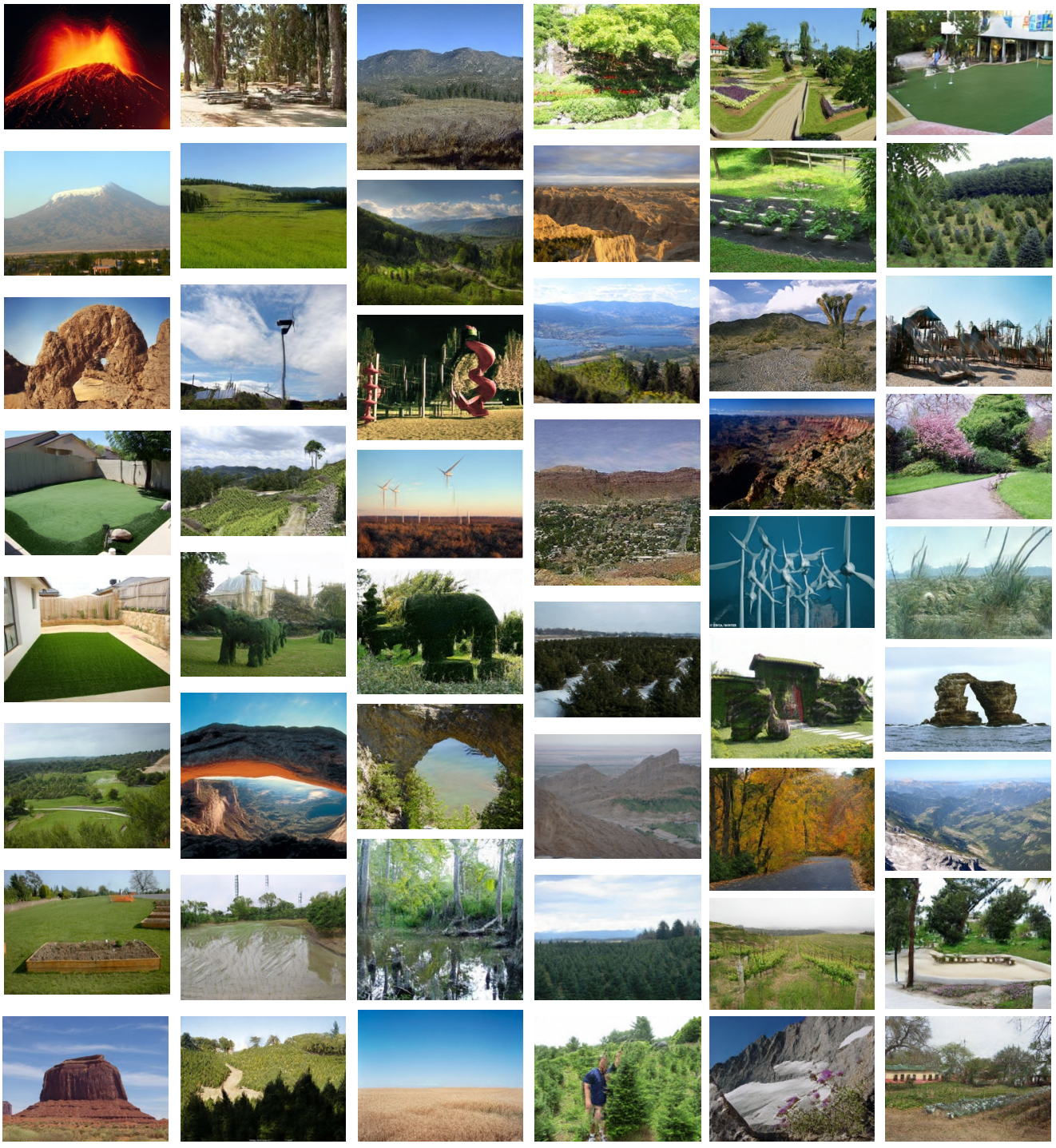
Real Images (Randomly chosen from the "places" database)



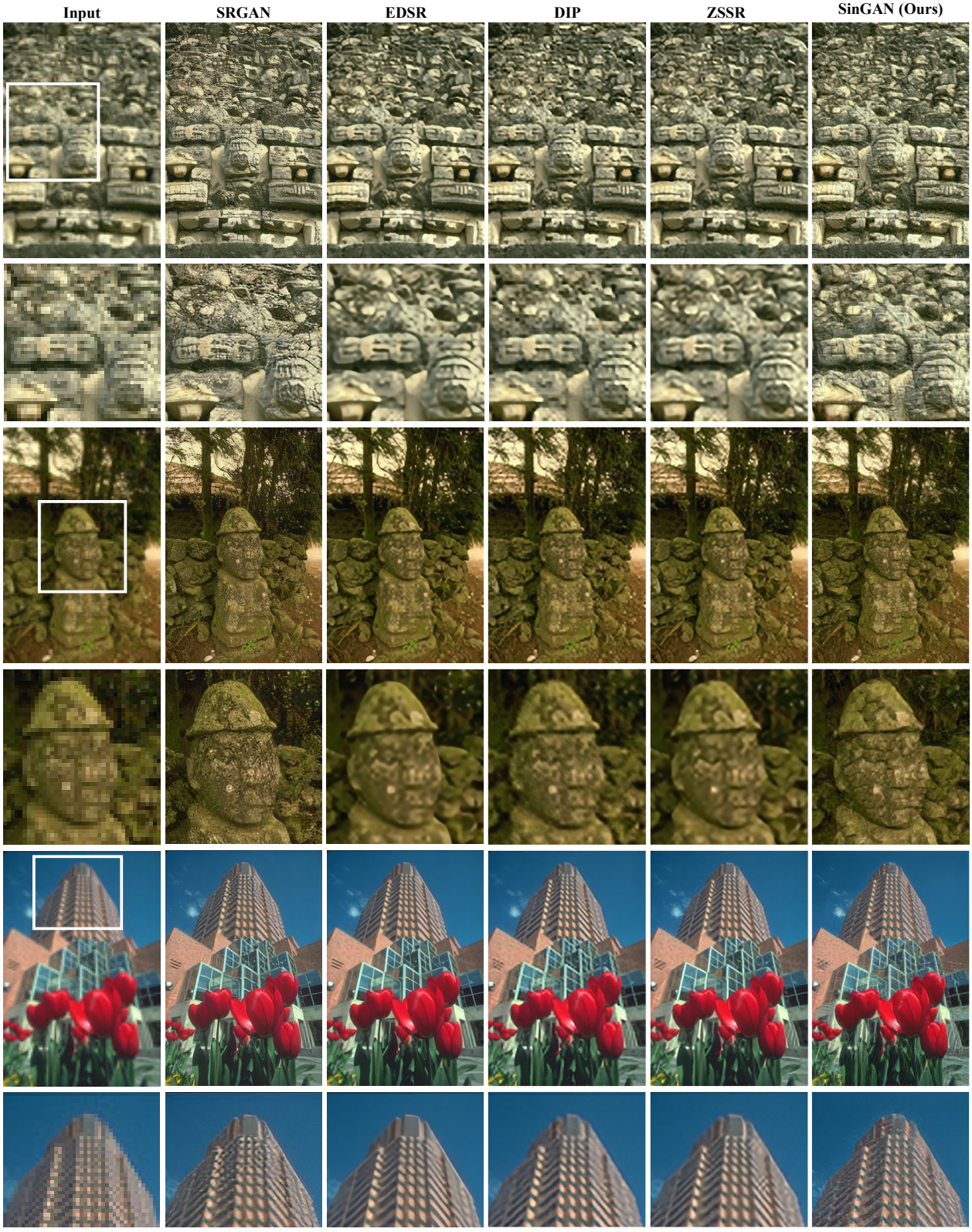
Fake Images, High Diversity Level

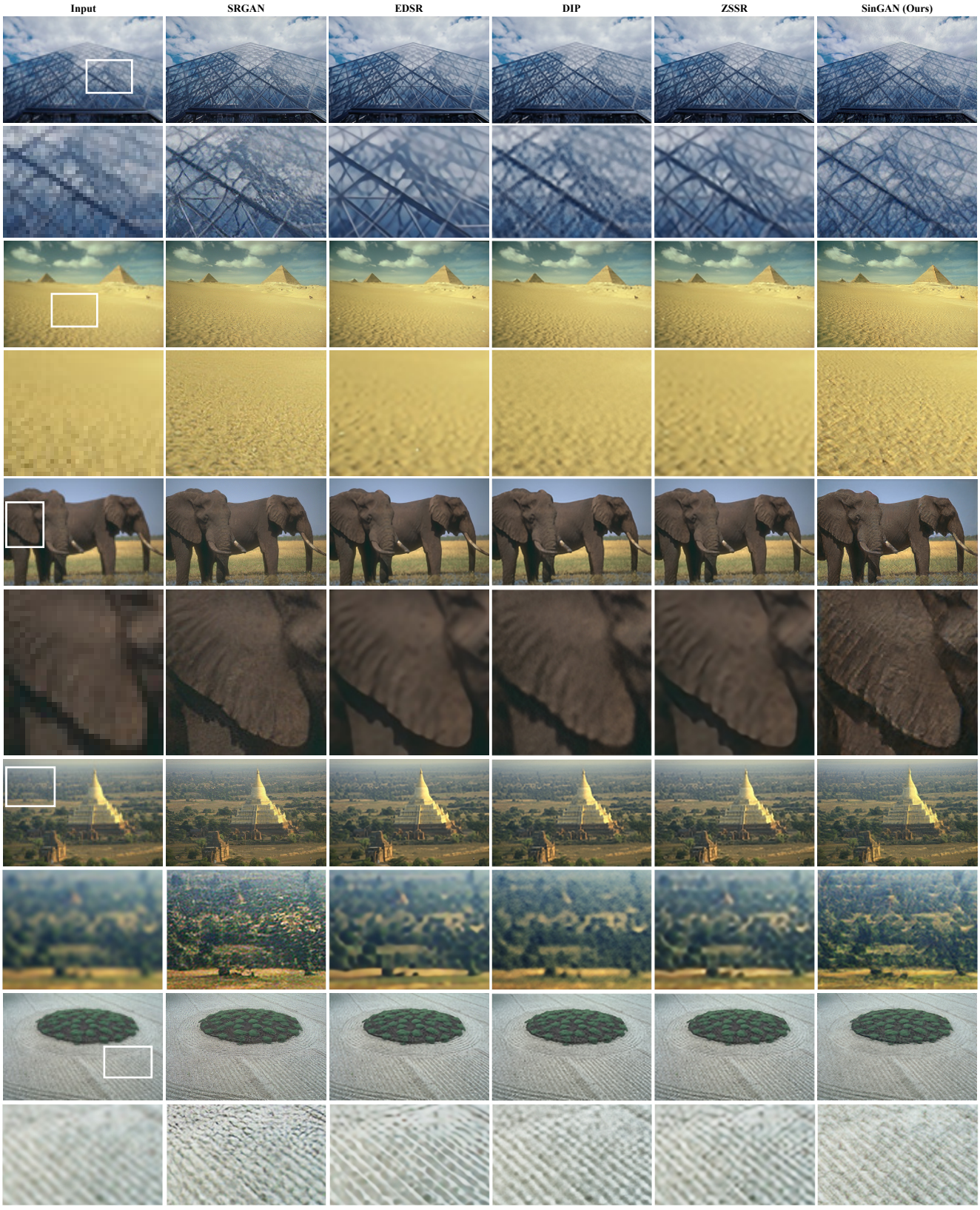


Fake Images, Low Diversity Level

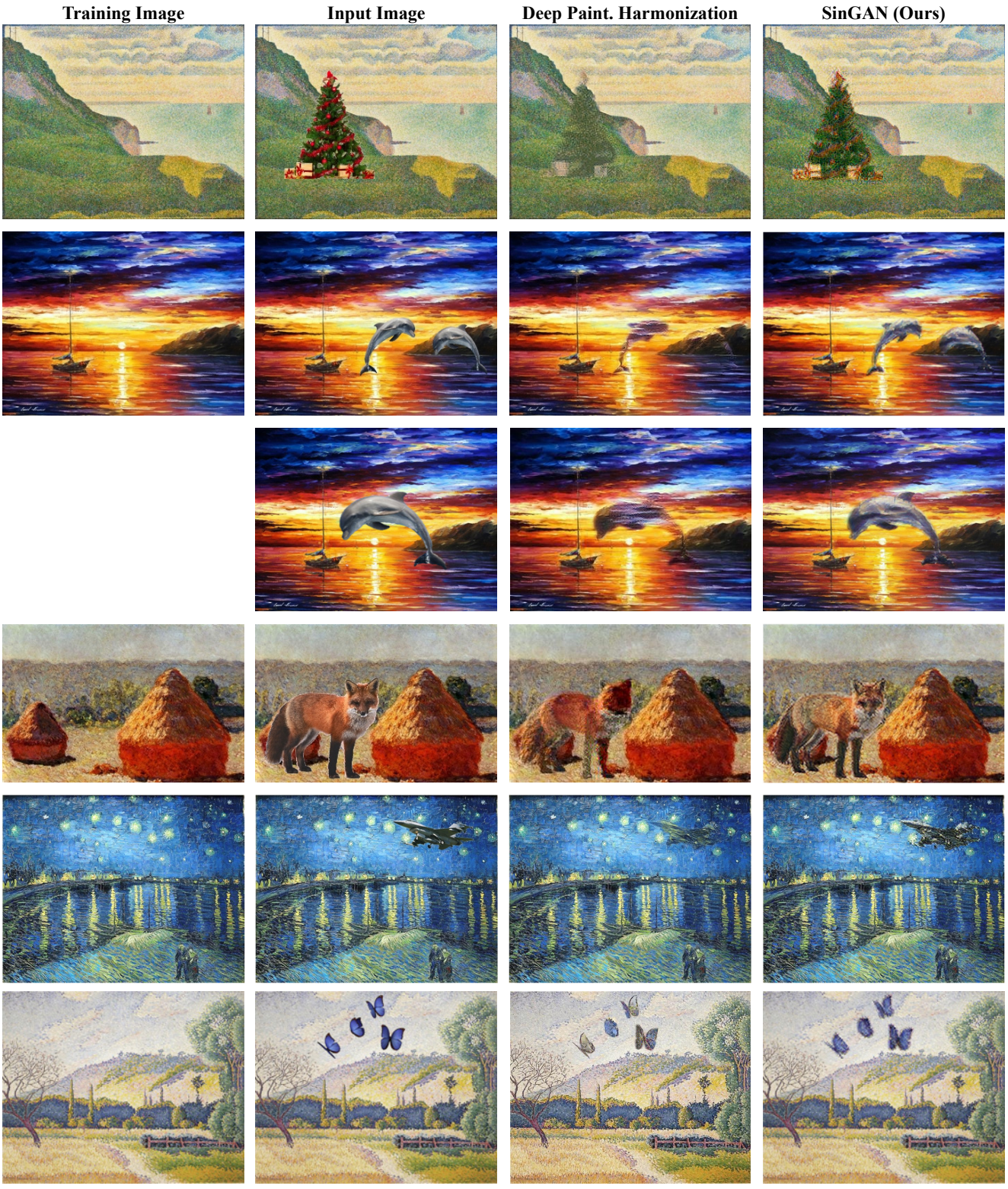


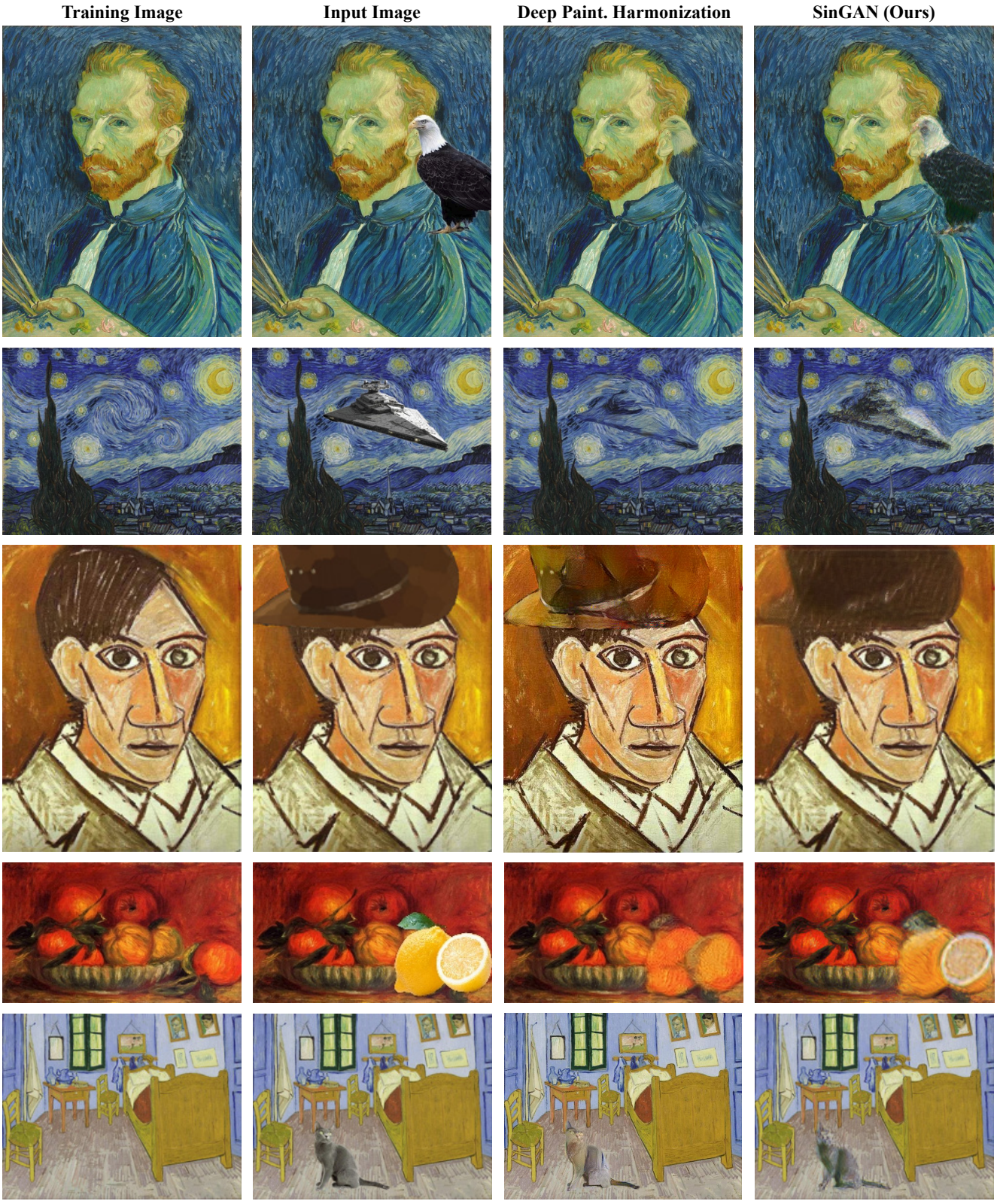
10. Super-resolution




Input
SRGAN
EDSR
DIP
ZSSR
SinGAN (Ours)

11. Harmonization

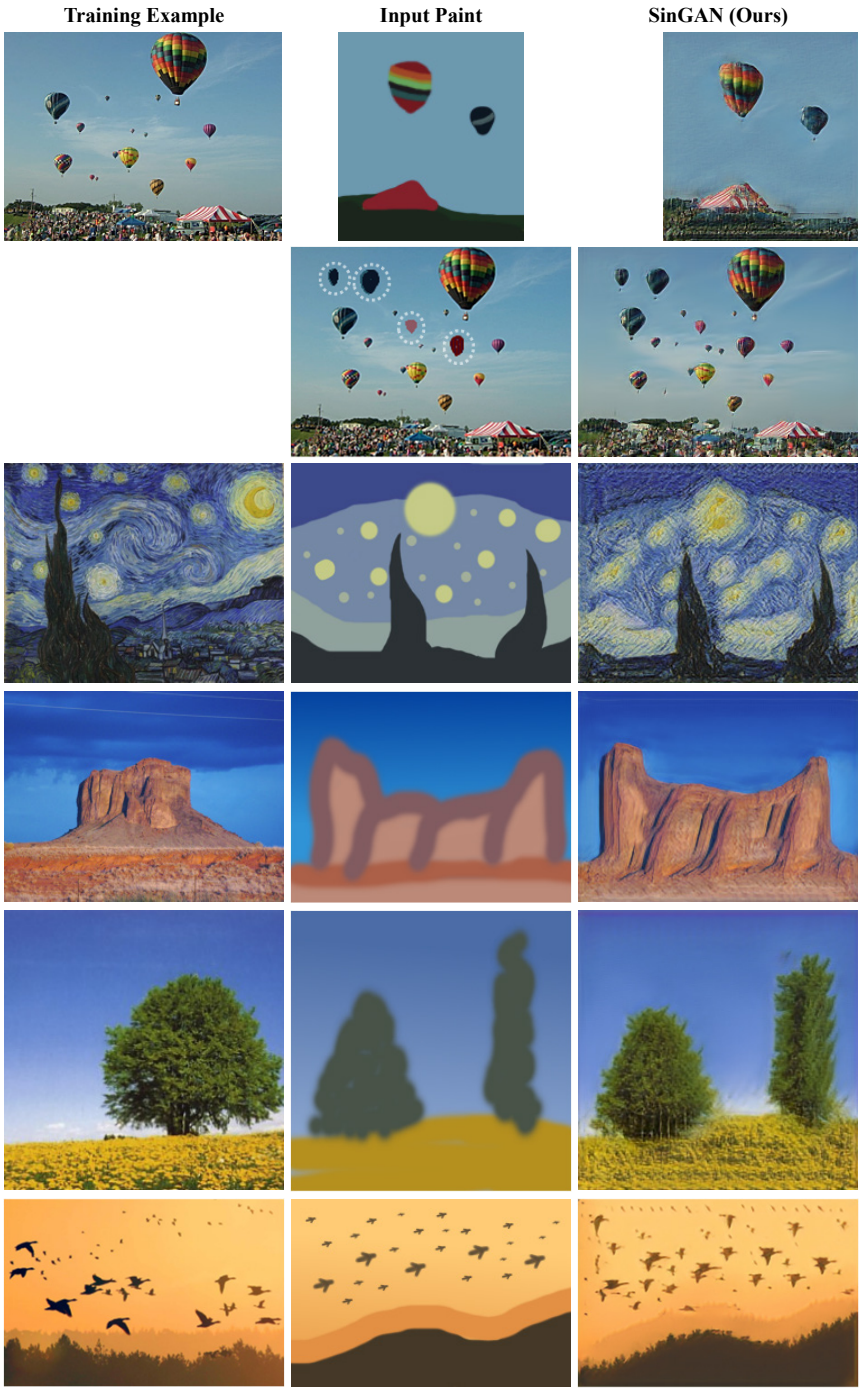



Training Image
Input Image
Deep Paint. Harmonization
SinGAN (Ours)

12. Editing


Training Example
Edited Input
Content Aware Move
SinGAN (Ours)

13. Paint to Image



When the training image is very textured, SinGAN may produce less realistic results in the paint-to-image task. This is because the conditional GANs at every scale $n < N$, are exposed to only textured images during training (those generated by the previous scale), and thus are not guaranteed to produce high quality results when fed with a smooth clipart image, as we do at test time. This effect can be somewhat countered by performing an additional short training phase of only the generator at the injection scale. During this fine tuning phase, instead of inputting the upsampled outputs from the previous scale, we input their quantized versions (which are more similar to cliparts). The generators at all other scales remain fixed. An example result of this approach is presented in Fig. 6. As can be seen, SinGAN's output in this case is more realistic.

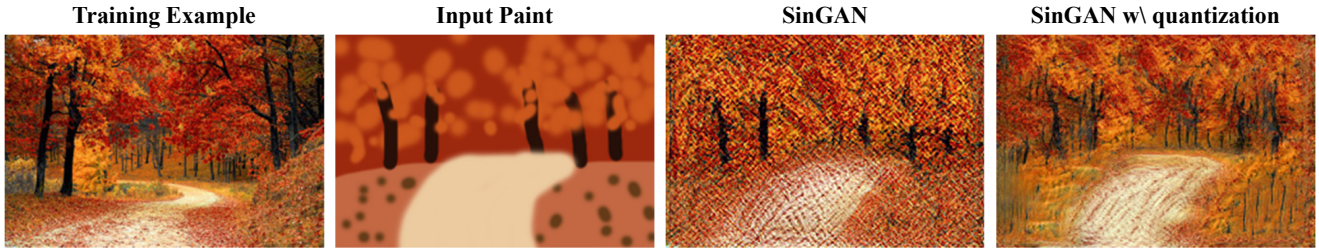


Figure 6: **Color quantization for paint to image.** When retraining the injection scale to accept color quantized input images, SinGAN manages to produce more realistic results in the paint-to-image task.

14. Experimental Settings

The tables below report the settings used for each of the results presented in the main text as well as in this supplementary material. For each application, we specify the scale in which the input image was injected and the total number of scales we used for SinGAN's generation pyramid.

Image	Injection scale	Total number of scales
Tree (also Fig. 2, main text)	$n = 1$	$N = 9$
Two Dolphins (also Fig. 13, main text)	$n = 3$	$N = 9$
Single Dolphin	$n = 3$	$N = 9$
Fox	$n = 2$	$N = 8$
Airplane	$n = 2$	$N = 8$
Butterfly	$n = 2$	$N = 8$
Eagle	$n = 2$	$N = 8$
Spaceship (also Fig. 13, main text)	$n = 3$	$N = 8$
Hat	$n = 4$	$N = 9$
Lemon	$n = 3$	$N = 7$
Cat	$n = 2$	$N = 8$

Table 1: **Harmonization.**

Image	Injection scale	Total number of scales
Rock1	$n = 5$	$N = 7$
Rock2	$n = 5$	$N = 7$
Rock3 (also Fig. 12, main text)	$n = 5$	$N = 7$
Tree	$n = 7$	$N = 9$
Mountain	$n = 4$	$N = 8$
Red cliff	$n = 5$	$N = 9$
Hay	$n = 6$	$N = 9$

Table 2: **Editing.**

Image	Injection scale	Total number of scales
Balloons1	$n = 7$	$N = 9$
Balloons2	$n = 5$	$N = 9$
Starry night	$n = 6$	$N = 8$
Rock	$n = 6$	$N = 8$
Tree	$n = 6$	$N = 8$
Birds	$n = 5$	$N = 7$
View (Fig. 2, main text)	$n = 7$	$N = 8$
Pyramids (Fig. 11, main text)	$n = 6$	$N = 8$
cows (Fig. 11, main text)	$n = 5$	$N = 7$

Table 3: **Paint to image.**

Image	Random walk starting scale	Total number of scales	α	β
Coral	$n = 5$	$N = 7$	0.1	0.9
Corals and fish	$n = 6$	$N = 8$	0.1	0.9
Water	$n = 8$	$N = 8$	0.1	0.8
Bush	$n = 6$	$N = 8$	0.1	0.9
Trees (slow wind)	$n = 6$	$N = 8$	0.1	0.9
Trees (strong wind)	$n = 6$	$N = 8$	0.1	0.8
Lightning	$n = 7$	$N = 7$	0.1	0.9
Fog	$n = 5$	$N = 7$	0.02	0.95
Fire1	$n = 8$	$N = 8$	0.2	0.6
Aurora	$n = 7$	$N = 8$	0.1	0.9

Table 4: **Single image animation.** See supplementary video.

References

[1] Shai Avidan and Ariel Shamir. Seam carving for content-aware image resizing. In *ACM Transactions on graphics (TOG)*, volume 26, page 10. ACM, 2007. 14

[2] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 2