Improved Techniques for Training Single-Image GANs

Tobias Hinz[1], Matthew Fisher[2], Oliver Wang[2], and Stefan Wermter[1]
[1]Knowledge Technology, University of Hamburg, Germany
[2]Adobe Research

Abstract

In this supplementary material we show additional results, results for applications not shown in the main paper, and implementation details. We first show additional results on unconditional image generation and then show results on unconditional image generation of arbitrary sizes. After that, we show how the number of concurrently trained stages, the learning rate scaling, and the novel rescaling approach affect the training outcome. We proceed with showing more examples from the image harmonization task and the depict the images used for our user studies. We conclude with examples of an image editing task, approaches we explored that did not work, failure cases, and more details about our implementation.

Contents

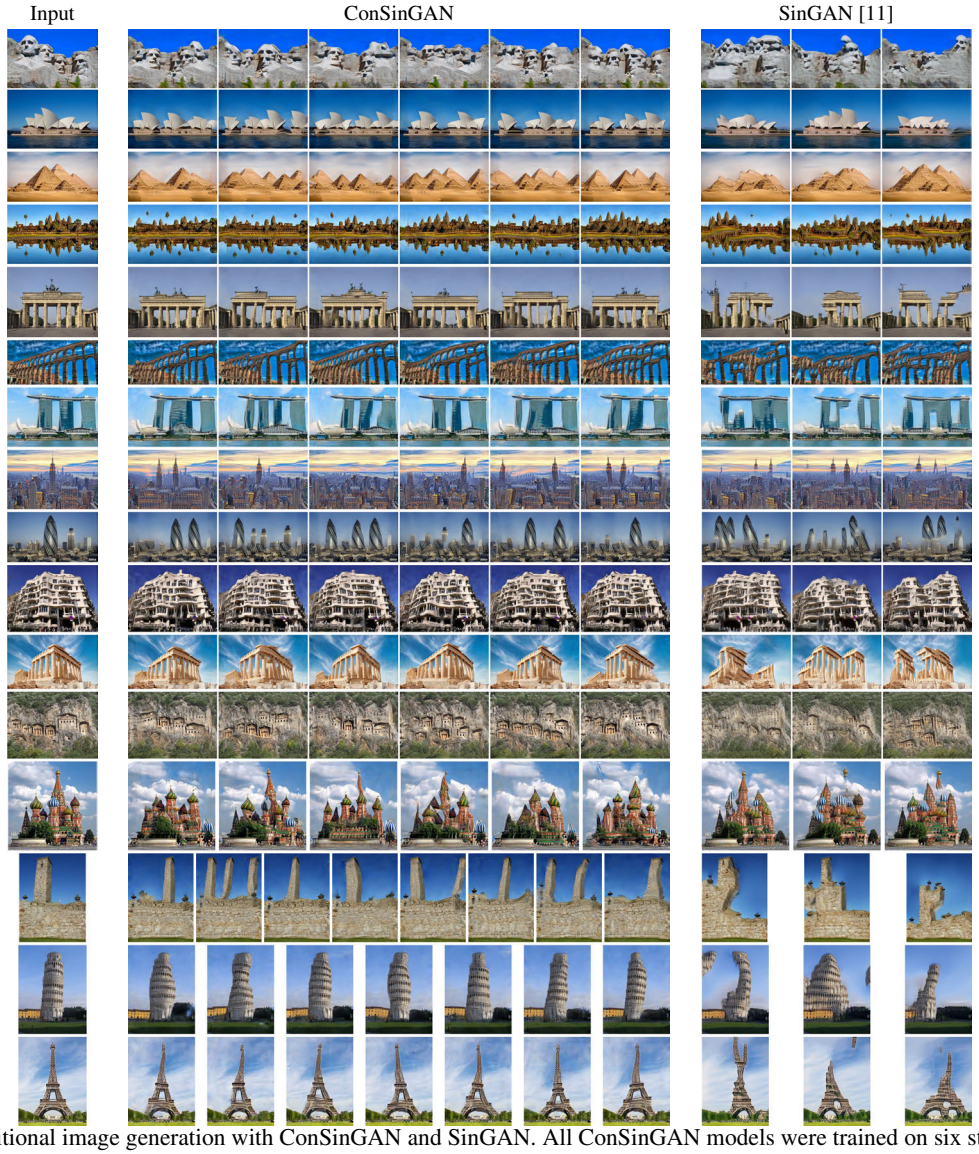


Figure 1. Unconditional image generation with ConSinGAN and SinGAN. All ConSinGAN models were trained on six stages, all SinGAN models were trained on eight – ten stages. We can see how ConSinGAN is able to model the global image structure better in most cases, despite being trained on fewer stages than SinGAN.

1. Unconditional Generation

Figure 1 shows more examples of unconditional image generation for both ConSinGAN and SinGAN [11]. Our ConSinGAN models were trained on six stages while the SinGAN models were trained with the default scaling factor of 0.75 for eight – ten stages per image. Despite this we can see that the ConSinGAN is capable of modeling the global image layout better than SinGAN in most cases. Training a ConSinGAN model takes roughly 20-25 minutes on our hardware (NVIDIA GeForce GTX 1080Ti), while training a SinGAN model takes roughly 2 hours.

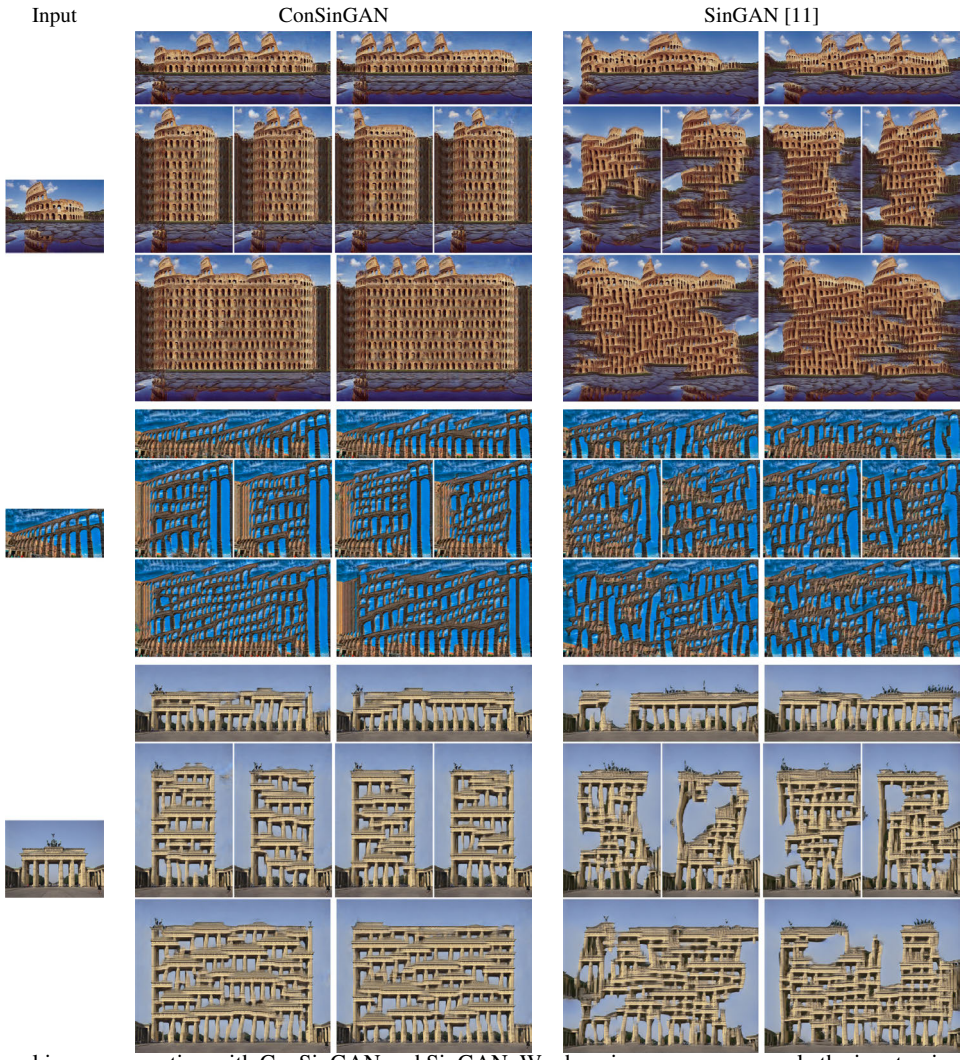
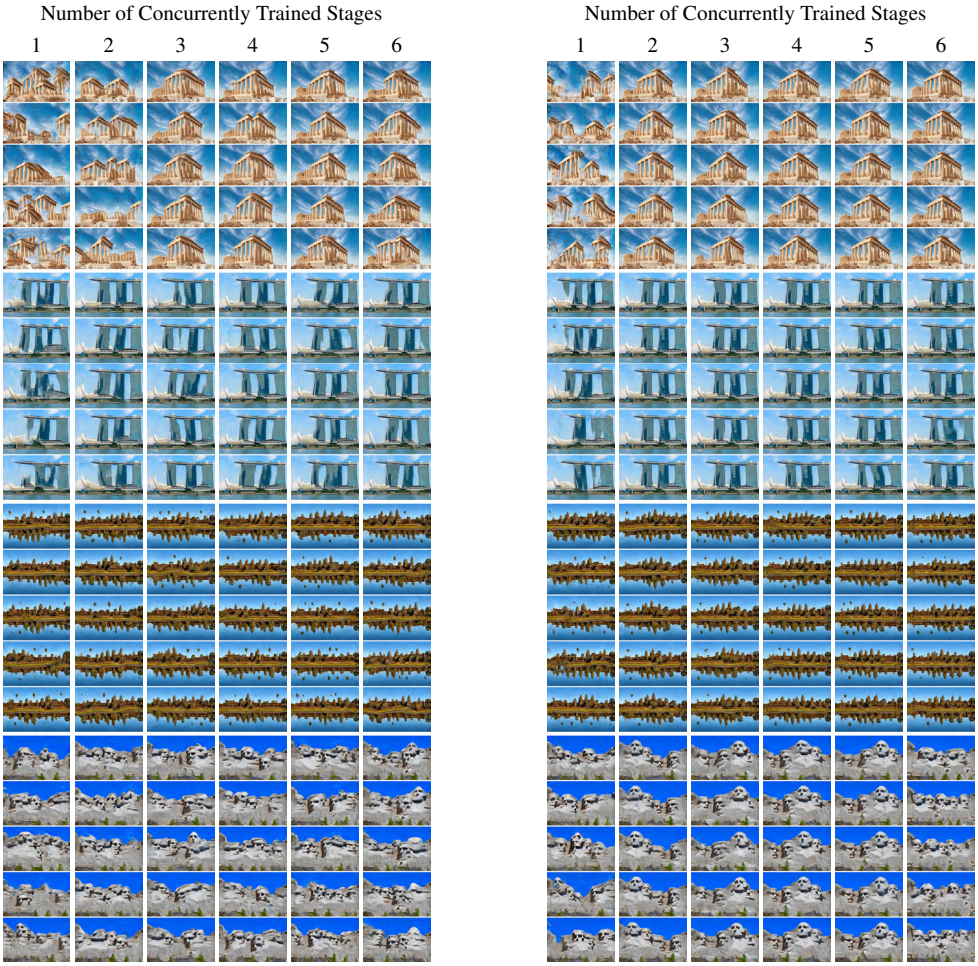


Figure 2. Unconditional image generation with ConSinGAN and SinGAN. We show images were we scale the input noise map by a factor of two along each side and along both sides. The ConSinGAN models were trained on six stages, while the SinGAN models were trained on eight – ten stages.

Figure 2 and Figure 3 show results of unconditional image generation with different aspect ratios and resolutions. Scaling in the horizontal direction usually works better for both models. Scaling in the vertical direction is much more challenging for both models, however, the ConSinGAN handles this better than the SinGAN.

Figure 3. Unconditional image generation with ConSinGAN and SinGAN. We show images were we scale the input noise map by a factor of two along each side and along both sides. The ConSinGAN models were trained on six stages, while the SinGAN models were trained on eight – ten stages.



$\delta = 0.1$ $\delta = 0.5$

Figure 4. Interplay between learning rate scaling δ and the number of concurrently trained stages for models that were trained on a total of six stages on different images. We can see how the image diversity usually decreases with increasing δ or increasing number of concurrently trained stages. All images are randomly sampled.

2. Number of Concurrently Trained Stages and Learning Rate Scaling

Figure 4 shows more visualizations of the interplay between the learning rate scaling δ and the number of concurrently trained stages. Again, we observe that image diversity decreases with increasing δ and increasing number of concurrently trained stages, leading to complete overfitting when trained end-to-end.

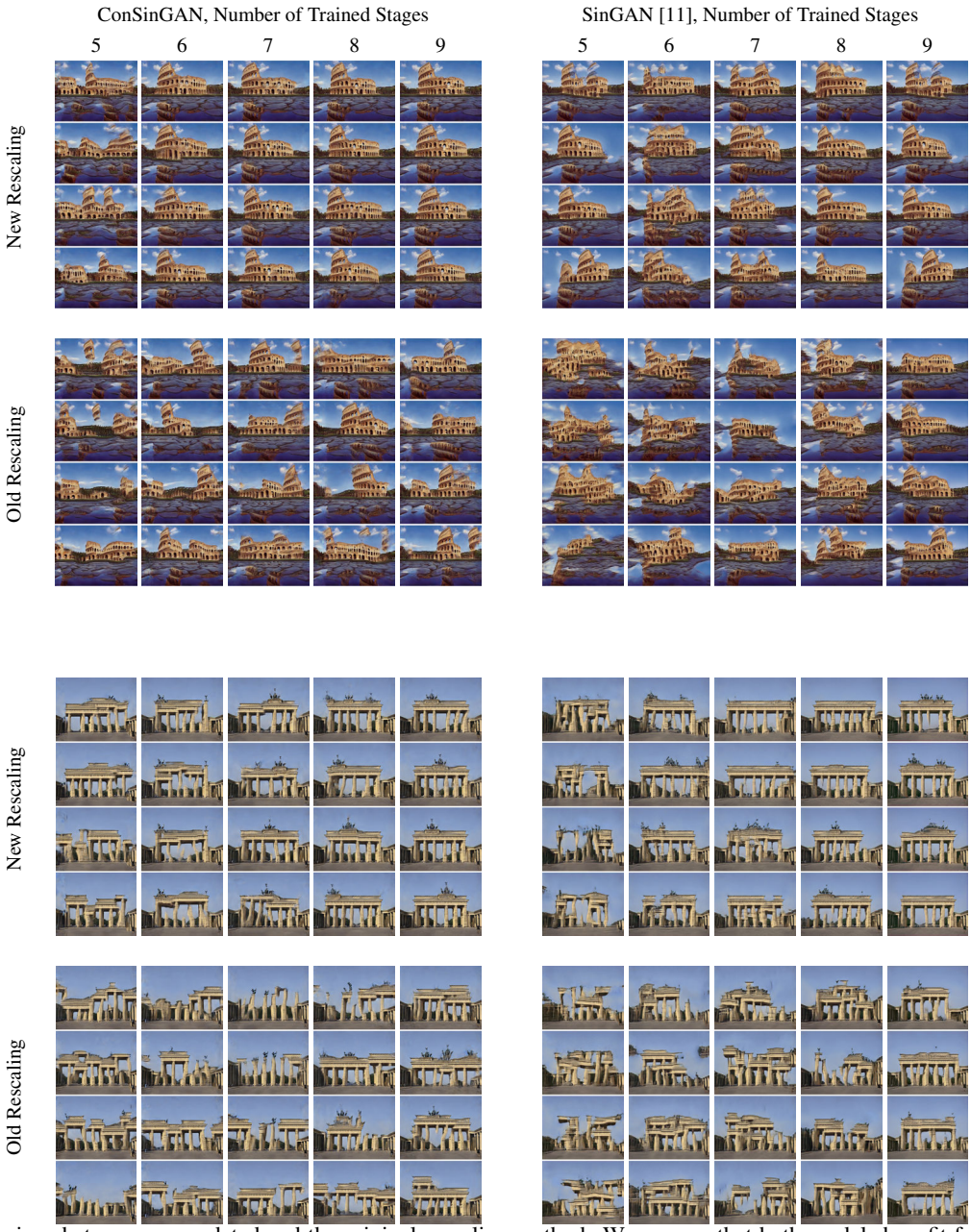


Figure 5. Comparison between our updated and the original rescaling method. We can see that both models benefit from the updated rescaling method and can generate realistic images with fewer stages. All images are randomly sampled.

3. Comparison of Original and Improved Rescaling Method

Figure 5 and Figure 6 show more examples of how the image quality develops with an increasing number of stages for the original and our improved rescaling method. Note that the ConSinGAN starts to generate realistic images after already 5-6 stages while the SinGAN usually needs more than 7 stages.

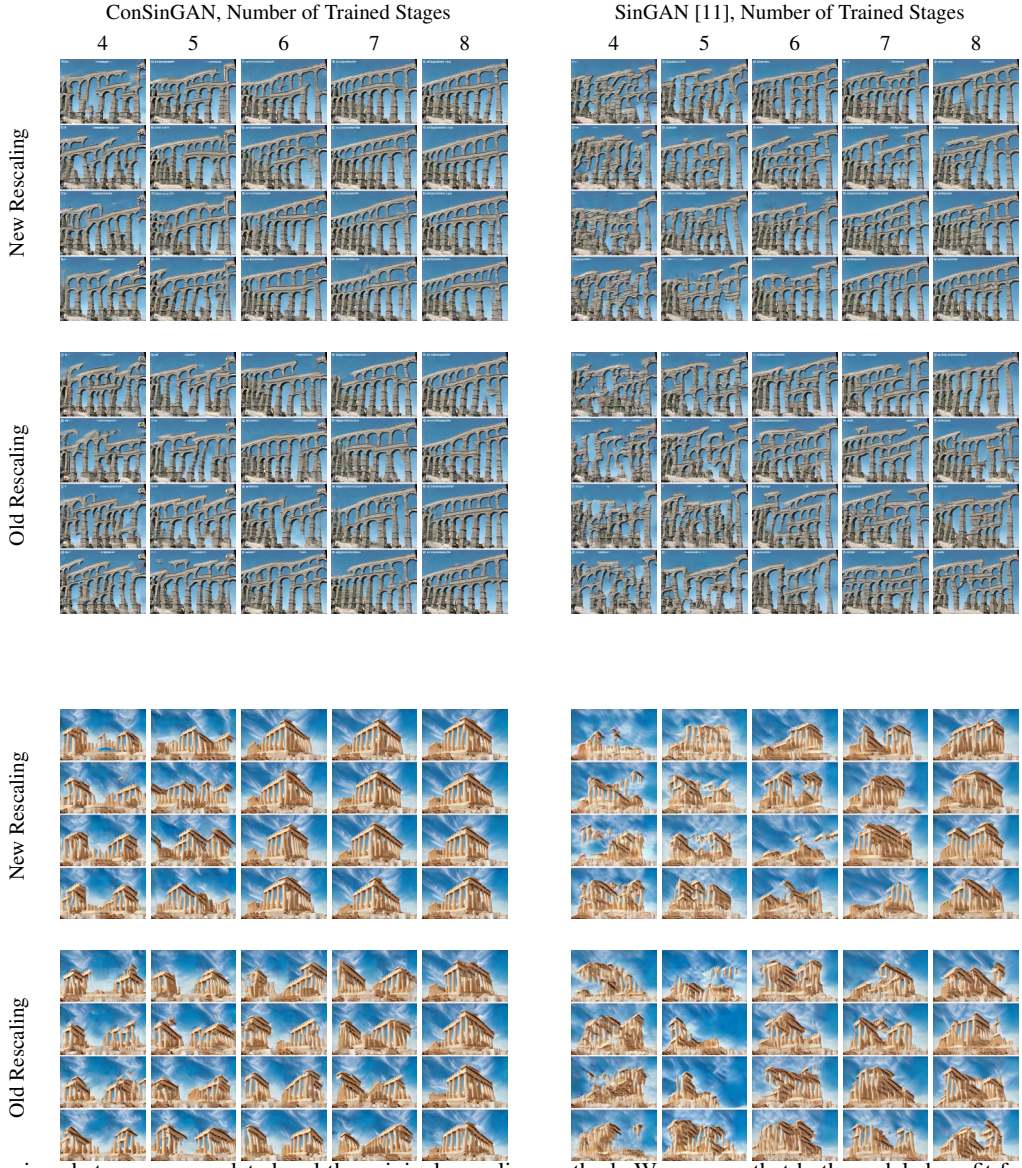


Figure 6. Comparison between our updated and the original rescaling method. We can see that both models benefit from the updated rescaling method and can generate realistic images with fewer stages. All images are randomly sampled.

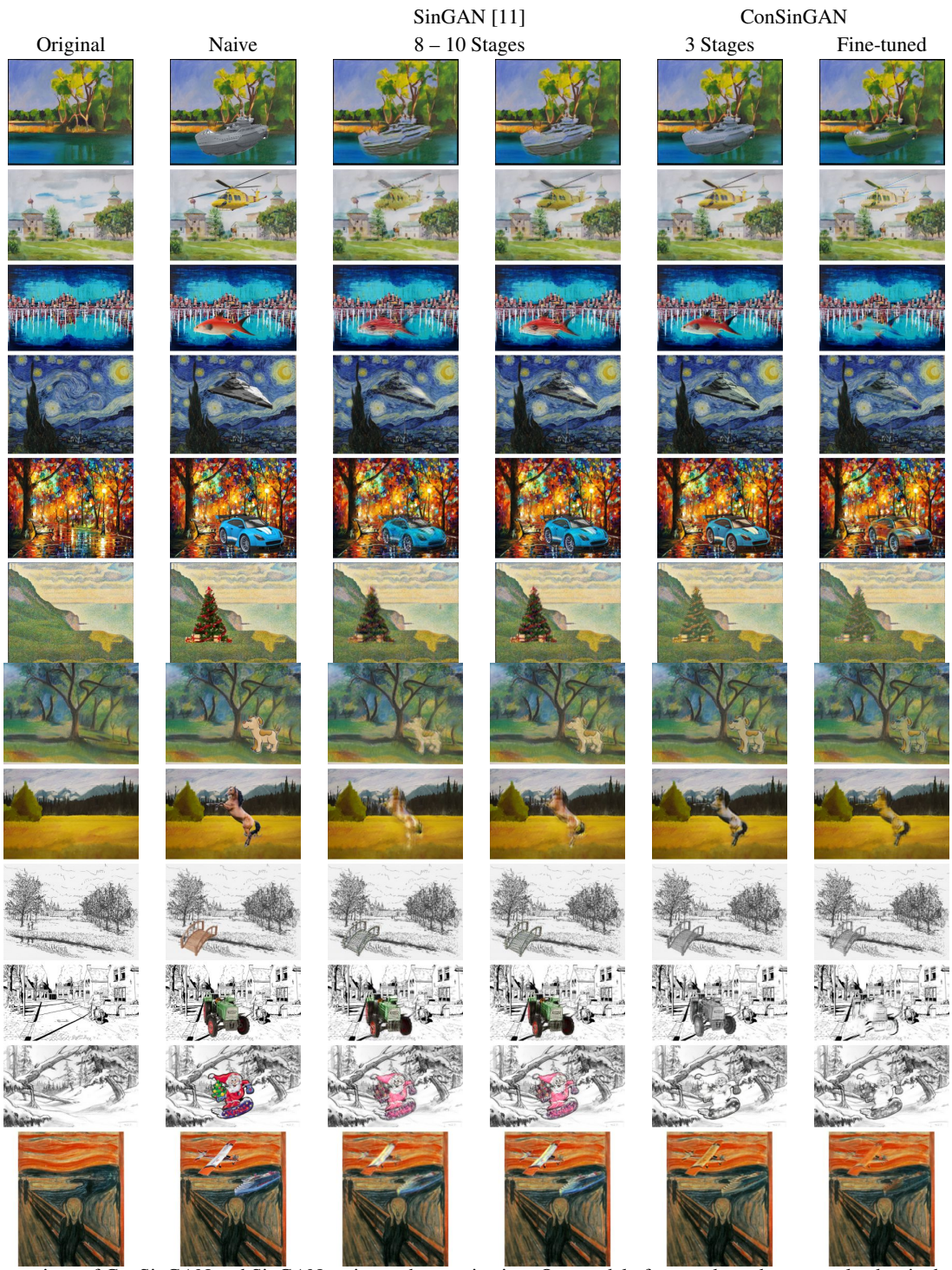


Figure 7. Comparison of ConSinGAN and SinGAN on image harmonization. Our model often produces better results despite being trained on fewer stages.

4. Image Harmonization

Figure 7 shows further comparisons between SinGAN and ConSinGAN on the image harmonization task. We can see that ConSinGAN often performs better despite being trained on fewer stages than SinGAN. SinGAN is also not able to transform color objects into black-and-white images, while this is no problem for ConSinGAN. Figure 8 and Figure 9 show more

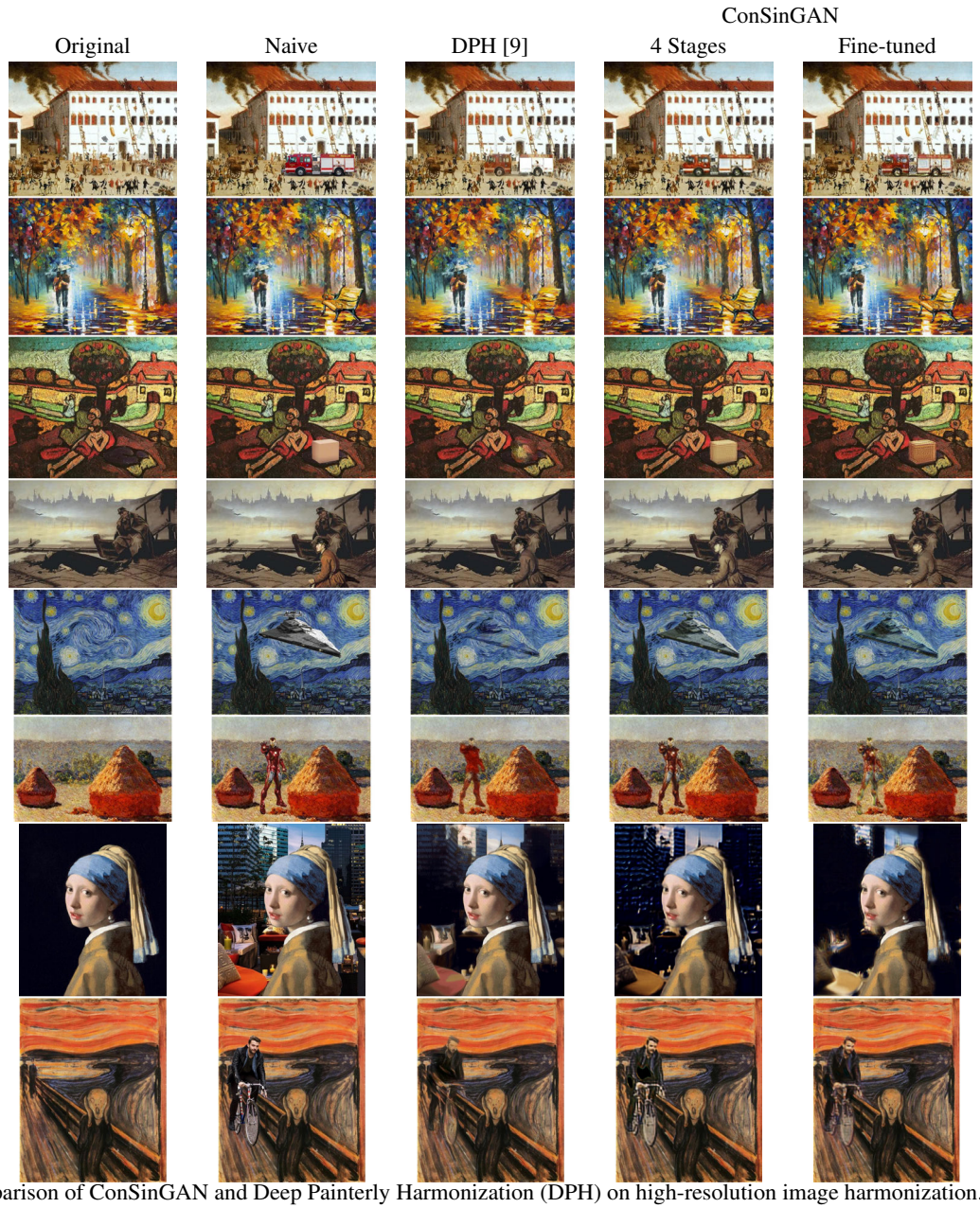


Figure 8. Comparison of ConSinGAN and Deep Painterly Harmonization (DPH) on high-resolution image harmonization. Images from DPH taken from their official Github implementation. In contrast to DPH, our model does only see the naive image during training if we fine-tune it (last column), but not for our general training procedure (fourth column).

comparisons between ConSinGAN and Deep Painterly Harmonization (DPH) [9] on image harmonization tasks of images with higher resolution. Note that DPH needs the target image and mask as input for its algorithm, while ConSinGAN is trained with randomly augmented images and only sees the target image at test time (except for the fine-tuned case).

Figure 9. Comparison of ConSinGAN and Deep Painterly Harmonization (DPH) on high-resolution image harmonization. Images from DPH taken from their official Github implementation. In contrast to DPH, our model does only see the naive image during training if we fine-tune it (last column), but not for our general training procedure (fourth column).

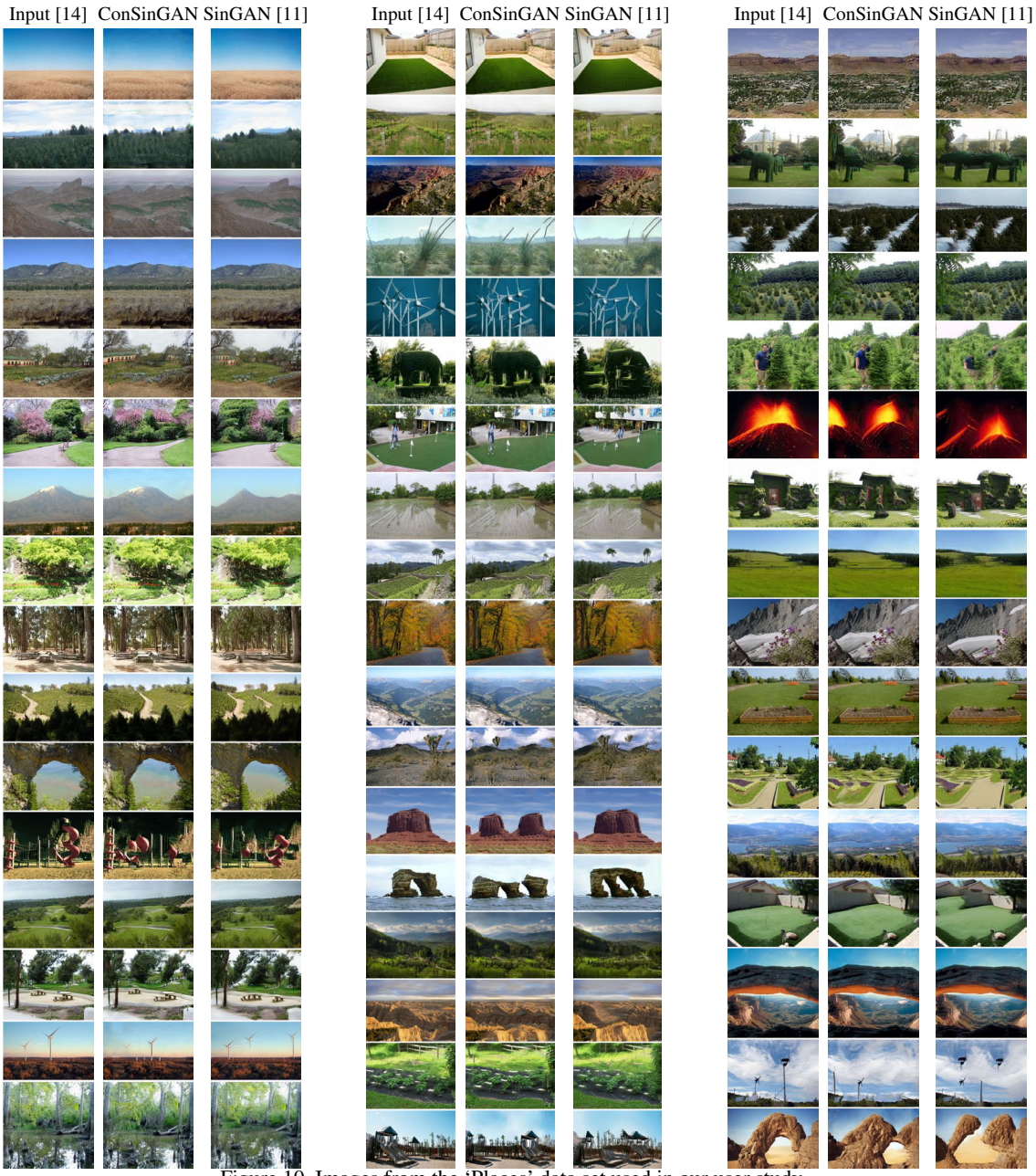


Figure 10. Images from the 'Places' data set used in our user study.

5. Images From User Studies

Figure 10 and Figure 11 show the images that were used in the two user studies on the 'Places' and 'LSUN' data sets respectively.

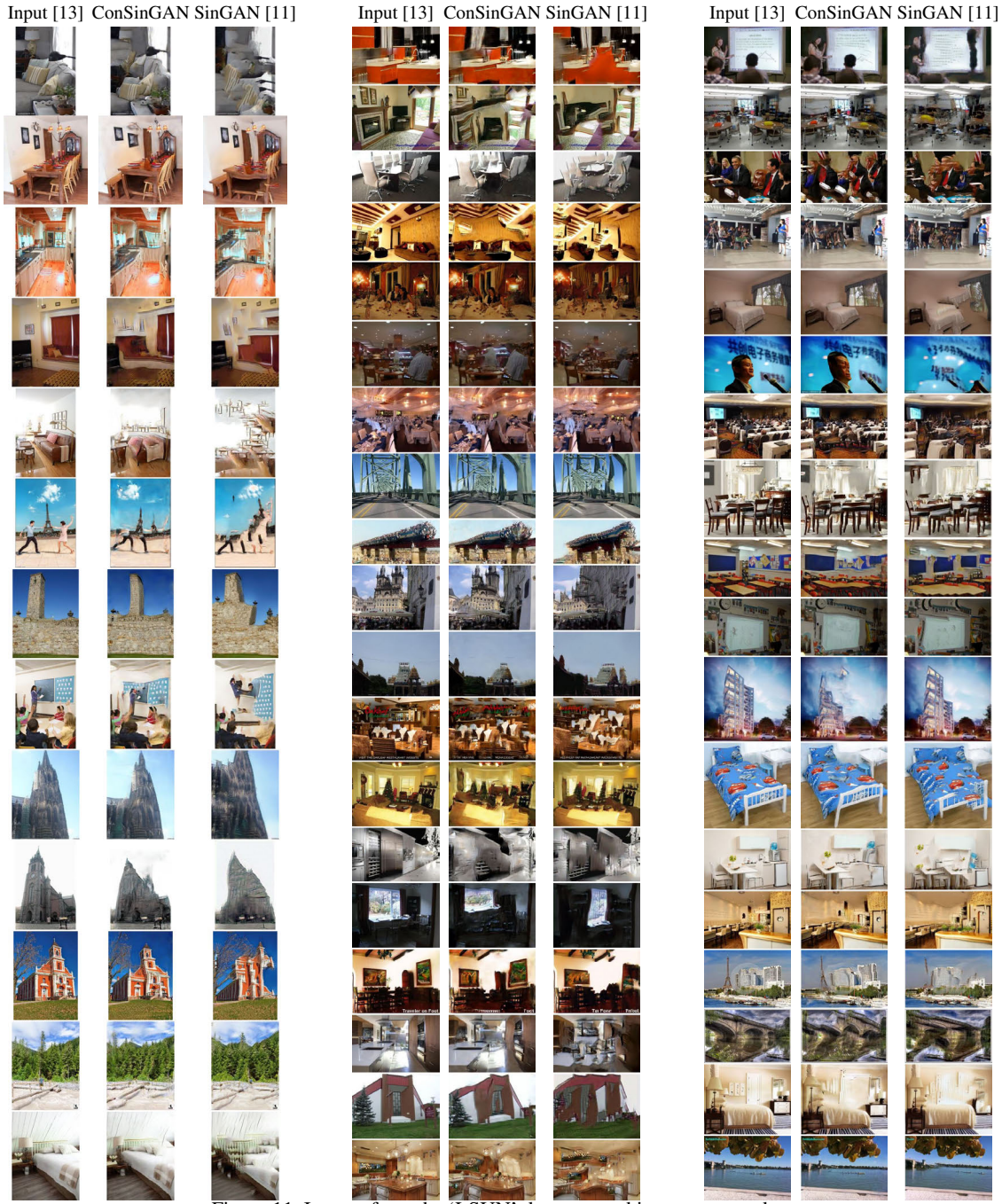


Figure 11. Images from the 'LSUN' data set used in our user study.

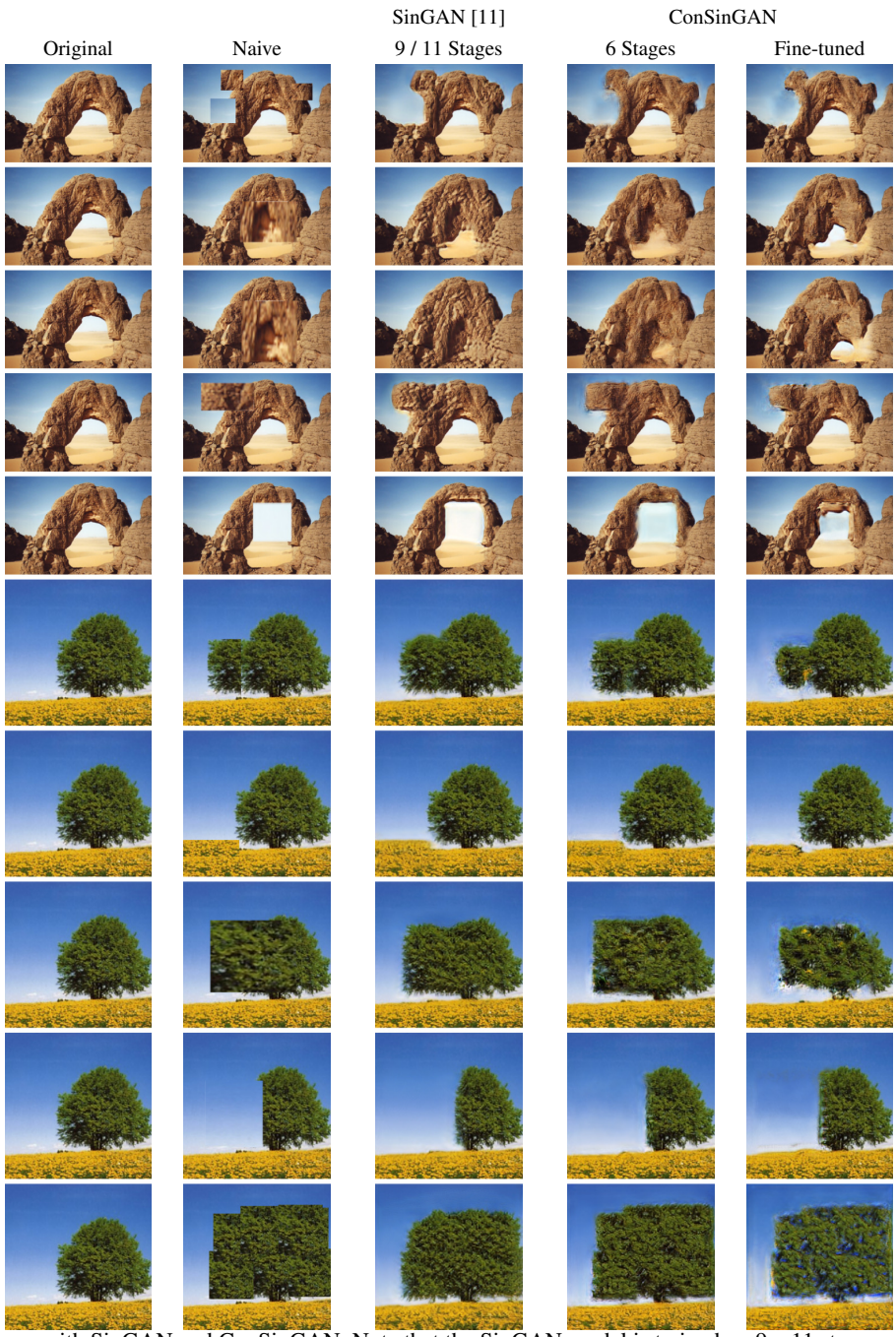


Figure 12. Editing images with SinGAN and ConSinGAN. Note that the SinGAN model is trained on 9 – 11 stages, while ConSinGAN is only trained for 6 stages

6. Image Editing

Figure 12 shows some examples of the image editing task with SinGAN and ConSinGAN. We trained the full SinGAN model (9 and 11 stages respectively) and chose the best results. The ConSinGAN was trained on only six stages and with only 1,000 iterations per stage, as opposed to 2,000 iterations per stage at the SinGAN. We can observe that both models have strengths and weaknesses. SinGAN is usually better at merging background objects (e.g. the sky in the stone image) but also introduces many artifacts, e.g. it changes the texture of the stone in many cases even in places where no editing took place. Furthermore, its texture is very repetitive when large ares are changed, e.g. the leaves in the changed areas of the tree.

ConSinGAN, on the other hand, does not change the structure in areas that are not edited and exhibits none of the repetitiveness in the features. However, it sometimes fails to merge the background as successfully as SinGAN does, see e.g. again the sky in the stone image. We can also see that ConSinGAN tends to adhere more closely to the layout of the edited image and mainly rounds and smooths the edges along edited areas. SinGAN changes more of the image which can sometimes look more realistic, but might not always be desired if the changes were done carefully and the artist wants the model to adhere to his changes as exactly as possible. Also note that we did not experiment with any hyperparameters for ConSinGAN for the image editing task and it might be possible to achieve better results by finding better hyperparameters.

7. Other Approaches We Explored

Multiple Discriminators at Each Stage Similar to other work, e.g. InGAN [12], we tried using several discriminators at each stage. This can potentially be helpful for tasks such as unconditional image generation at different resolutions (see Figure 2 and Figure 3). To test this we adapted our model so that the generator is trained with multiple discriminators at each stage. We generate images at different fixed resolutions, e.g. by scaling the width and height of the input noise map by factors 0.5, 1.0, 1.5, and 2.0. As a result, the generator generates several images at each iteration which are used as inputs for several discriminators (one for each resolution).

Each of the discriminators is trained on only one specific resolution where the 'real' image is a rescaled version of the original image. We observed that this does indeed often lead to improved results for the unconditional generation of various resolutions. However, for other applications, the results were inconsistent – sometimes it improved things, sometimes it did not. Since training several discriminators at each stage increases the training time considerably we decided to not use this approach in the default settings of our method.

Adaptive Number of Training Iterations at Each Stage At the moment, each stage is trained for a pre-defined number of iterations, e.g. 2,000 iterations for unconditional image generation and 1,000 iterations for image harmonization. We believe it is unlikely that each stage has to learn the same amount of information, especially since each stage is initialized with the weights of the previous stage. It should therefore, in theory, be possible to train each stage only for as long as necessary, thereby potentially reducing the training time even more. We tried this by measuring how much the generated image changes during the training of each stage (similar to what [6] did in Fig. 8) and to stop training a given stage when it does not change the output of a given image much compared to previous iterations. Again, this approach sometimes led to good results with reduced training iterations, but the results were inconsistent. However, we believe that this is a worthwhile direction and better approaches might make it possible to achieve good results with considerably fewer iterations on some (all) stages of training.

Further Improve Global Image Layout We also tried several other approaches to further improve the global image layout, especially from complex images. One idea is to add a second task for the discriminator, so that it not only has to decide whether a given image patch is real or not, but also where in a given image the patch is located (roughly). We implemented this by adding a "location loss" to the discriminator so that it also had to predict the location of a given image patch. To prevent overfitting we split the input image into nine equal rectangles (3×3) and the discriminator had to predict (for the real image only) where a given image patch is from. The generator was then trained to fool the discriminator into both predicting that the image patches of the generated images are real and to being able to correctly predict their location, too.

Our second approach was to add a second discriminator with increased receptive field to the higher stages. The idea is that this second discriminator could then still judge the global layout (and not only texture/style) even at higher resolutions. To reduce the computational burden we did not increase the convolutional filter as such, but used dilated convolutions instead. However, training still takes longer since we have to train a second discriminator at higher stages.

Both approaches had mixed results, with the added discriminator with dilated convolutions overall performing better than the location loss. The location loss often did not clearly improve the global layout and sometimes led to reduced diversity in the generated images. On the other hand, using an additional discriminator with dilated convolutions on higher stages often did actually improve the global consistency, but on average the improvements were not big enough to warrant the extra training time. We still feel that these, or similar, avenues should be further studied, since enforcing better global layout in this manner might enable us to train on even fewer stages, thereby negating the more expensive training and possibly even speeding up the overall training time.

Data Augmentation To increase the diversity in the generated images we experimented with applying basic augmentation techniques to the original training image. At each iteration we applied simple transformations such as horizontal mirroring, taking a random crop (consisting of at least 95% of the original image), slight rotation (± 5 degrees), slight zooming, etc. However, this considerably worsened the final results in our tests, since the discriminator was apparently not able to learn a

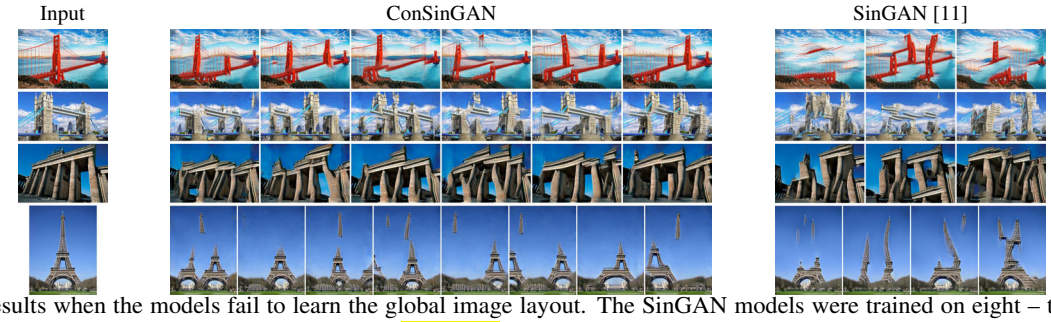


Figure 13. Results when the models fail to learn the global image layout. The SinGAN models were trained on eight – ten stages, the ConSinGAN models were trained on three – five stages. Note that increasing the number of stages and/or the learning rate scaling δ for the ConSinGAN models improves the learned image layout.

good image representation and the generated images showed several artifacts (uneven textures/surfaces, no straight lines, etc). Improving/fine-tuning the used augmentation techniques or finding a more appropriate (sub-)set of augmentation techniques for this task might improve results.

Different Normalization Approaches and Activation Functions We also experimented with different normalization approaches, both for the input image/noise and for the network architecture. Our generator gets as input either the original training image (normalized to a range $[-1, 1]$) for the reconstruction loss and noise sampled from a random normal distribution ($\mathcal{N}(0, 1)$) for the unconditional image generation. As such, the input to our generator comes from two slightly different distributions. We tried both normalizing the input image so that it more closely resembles a normal distribution and sampling the noise so that it follows more closely the image distribution (e.g. by sampling from $\mathcal{U}(-1, 1)$. However, both approaches did not improve the image quality or training progress and the training does, in fact, not seem to suffer from the two slightly different input distributions.

We also tried using other normalization techniques besides batch normalization in the network architecture. We tried both layer normalization [1] and pixel normalization [5], where layer normalization did not improve the results and pixel normalization made the results considerably worse. In the end we were able to completely remove any normalization layers for the unconditional image generation, which had the positive benefit of further speeding up the training. We also experimented with other activation functions besides leaky ReLU [10], such as ELU [2], SELU [8], and PRELU [4]. PRELU usually led to similar or better results, but also slowed down training since it introduces an additional parameter. Both ELU and SELU had negative effects on the final result and in the end we kept the leaky ReLU activation function, since it works almost as well as PRELU but does not slow down the training.

8. Failure Cases

Figure 13 shows examples of trained models that did not learn a correct global layout of the training image. Note that the SinGAN models were trained for the full eight – ten stages, while the ConSinGAN models were only trained for three – five stages. Increasing the number of trained stages to the default of six for the ConSinGAN increases the image quality.

9. Optimization and Implementation Details

Preprocessing We first rescale the input image so that its longer side has a resolution of 250 pixels for stage N and its shorter side a resolution of 25 pixels for stage 0. We then resize the orginal image to the resolutions of the intermediate stages according to Equation 3 of the original paper. The image is normalized to values in range $[-1, 1]$ during training.

Network Architecture Our discriminator and each stage of our generator consist of three convolutional layers with 64 filters each and a filter size of 3×3. Both the discriminator and the generator have two additional layers taking an image (or the noise mask) as input and extracting features, and mapping features to images (generator) or loss space (discriminator) respectively.

We use the Leaky ReLU (LReLU) activation function [10]. When BN is not used, we observe that the parameter $LReLU_\alpha$ which sets the negative slope in the LReLU does have some impact on the convergence and the final results for different tasks. For all unconditional image generation tasks we set $LReLU_\alpha = 0.05$, while setting $LReLU_\alpha = 0.3$ for other tasks such as image harmonization.

Optimization At each stage we optimize our model for 2,000 iterations (unconditional image generation) or 1,000 iterations (image harmonization and editing). We use the WGAN-GP loss [3] with a gradient penalty weight of 0.1. The learning rate starts with $\eta = 0.0005$ at each stage for both the generator and the discriminator and gets multiplied with 0.1 after 80% of iterations steps at each stage. Optimization is done with Adam [7] with $\beta_1 = 0.5$ and $\beta_2 = 0.999$. We train three stages concurrently with a learning rate scaling of $\delta = 0.1$ for the lower stages.

During each iteration we first perform three gradient steps on the discriminator. We found that we could speed up training by only performing one gradient update on the generator during each iteration, but scaling it by a factor of 3. The scalar for the reconstruction loss in the generator is $\alpha = 10.0$. We use the leaky ReLU activation with a negative slope of 0.05 for image generation and 0.3 for image harmonization. We do not use batch normalization for unconditional image generation, but found it useful in the generator (but not the discriminator) for other tasks such as image harmonization and editing.

Training Time Training takes around 20-25 minutes for unconditional image generation and 10 minutes for image harmonization for a 250×250 pixel image on an NVIDIA GeForce GTX 1080Ti compared to training a SinGAN model which takes on average 120-140 minutes on the same hardware. One of the reasons for this is that we only need to train our model on 5-6 stages for good results, while the original SinGAN needs 8-10 stages. However, even when both models are trained on the same number of stages we observe that it still takes less time to train our model, possibly because we do not generate an image after each stage (as SinGAN does) and do not rely as much on BN.

References

[1] Ba, J.L., Kiros, J.R., Hinton, G.E.: Layer normalization. arXiv preprint arXiv:1607.06450 (2016)

[2] Clevert, D.A., Unterthiner, T., Hochreiter, S.: Fast and accurate deep network learning by exponential linear units (elus). In: International Conference on Learning Representations (2016)

[3] Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., Courville, A.C.: Improved training of wasserstein gans. In: Advances in Neural Information Processing Systems. pp. 5767–5777 (2017)

[4] He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 1026–1034 (2015)

[5] Karras, T., Aila, T., Laine, S., Lehtinen, J.: Progressive growing of gans for improved quality, stability, and variation. In: International Conference on Learning Representations (2018)

[6] Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J., Aila, T.: Analyzing and improving the image quality of stylegan. In: Proceedings of the IEEE Computer Vision and Pattern Recognition (2020)

[7] Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: International Conference on Learning Representations (2015)

[8] Klambauer, G., Unterthiner, T., Mayr, A., Hochreiter, S.: Self-normalizing neural networks. In: Advances in Neural Information Processing Systems. pp. 971–980 (2017)

[9] Luan, F., Paris, S., Shechtman, E., Bala, K.: Deep painterly harmonization. Computer Graphics Forum **37**(4), 95–106 (2018)

[10] Maas, A.L., Hannun, A.Y., Ng, A.Y.: Rectifier nonlinearities improve neural network acoustic models. In: International Conference on Machine Learning. vol. 30 (2013)

[11] Shaham, T.R., Dekel, T., Michaeli, T.: Singan: Learning a generative model from a single natural image. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 4570–4580 (2019)

[12] Shocher, A., Bagon, S., Isola, P., Irani, M.: Ingan: Capturing and retargeting the "dna" of a natural image. In: Proceedings of the IEEE International Conference on Computer Vision (2019)

[13] Yu, F., Zhang, Y., Song, S., Seff, A., Xiao, J.: Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. arXiv preprint arXiv:1506.03365 (2015)

[14] Zhou, B., Lapedriza, A., Xiao, J., Torralba, A., Oliva, A.: Learning deep features for scene recognition using places database. In: Advances in Neural Information Processing Systems. pp. 487–495 (2014)