Improved Techniques for Training Single-Image GANs

Preprint · March 2020

CITATIONS
0

READS
43

4 authors, including:

Tobias Hinz
University of Hamburg
13 PUBLICATIONS **48** CITATIONS
SEE PROFILE

Stefan Wermter
University of Hamburg
460 PUBLICATIONS **3,648** CITATIONS
SEE PROFILE

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
[2] Adobe Research

Abstract. Recently there has been an interest in the potential of learning generative models from a *single* image, as opposed to from a large dataset. This task is of practical significance, as it means that generative models can be used in domains where collecting a large dataset is not feasible. However, training a model capable of generating realistic images from only a single sample is a difficult problem. In this work, we conduct a number of experiments to understand the challenges of training these methods and propose some best practices that we found allowed us to generate improved results over previous work in this space. One key piece is that unlike prior single image generation methods, we *concurrently* train several stages in a sequential multi-stage manner, allowing us to learn models with fewer stages of increasing image resolution. Compared to a recent state of the art baseline, our model is up to six times faster to train, has fewer parameters, and can better capture the global structure of images.

1 Introduction

Generative Adversarial Networks (GANs) [13] are capable of generating realistic images [6] that are often indistinguishable from real ones in limited domains [27]. The resulting models can be used for many different tasks, such as unconditional and conditional image synthesis [26,19], image inpainting [10], and image-to-image translation [22,50]. However, these GANs are trained on large datasets, typically consisting of tens of thousands of images. Training generative models on such large datasets can be time-consuming and expensive.

In some cases, it might be preferable to train a generative model on a very small number of images or, in the limit, on a single image. This is useful if we want to obtain variations of a given image, work with a very specific image or style, or only have access to very little training data. The recently proposed SinGAN [38] introduces a GAN that is trained on a single image for tasks such as unconditional image generation and image harmonization. In this work, we start with this approach as a baseline and conduct a number of experiments to see how we can improve training single image GANs.

We find that exactly how multi-stage and multi-resolution training is handled is critical. In particular, training only one stage at a given time limits interactions between different stages, and propagating images instead of feature maps from one generator stage to the next negatively affects the learning process. Conversely, training all stages end-to-end causes overfitting in the single image scenario, where the network collapses to generating only the input image. We experiment with this balance, and find a promising compromise, training multiple stages in parallel with decreased learning rates, and find

that this improves the learning process, leading to more realistic images with less training time. Furthermore, we show how it is possible to directly trade-off image quality for image variance, where training more stages in parallel means a higher global image consistency at the price of less variation.

We also conduct experiments over the choice of rescaling parameters, i.e. how we decide at which image resolution to train at each stage. We observe that the quality of the generated images, especially the overall image layout, quickly degrades when there are not enough training stages on images with small resolution. Since the receptive field of the discriminator stays the same throughout training, lower stages with smaller resolutions are important for the overall image layout, while higher stages with larger resolution are important for the final image texture and color. We find that we only need relatively few training stages with high-resolution images in order to still generate images with the correct texture. As a consequence, we put a higher weight on smaller resolution images during training while using fewer of the stages to train on high-resolution images which improves the synthesized image quality.

Finally, since our model trains several stages in parallel, we can introduce a *task-specific fine-tuning stage* which can be performed on any trained model. For several tasks, such as e.g. image harmonization, we show that after training an initial model, it is possible to fine-tune it on a given specific image to further improve results. This fine-tuning shows benefits with as few as 500 additional training iterations and is therefore very fast (less than two minutes on our hardware).

Combining these proposed architecture and training modifications enables us to generate realistic images with fewer stages and significantly reduced overall training time (20-25 minutes versus 120-150 minutes in the original SinGAN work). To summarize, our main contributions are:

1. we train several stages in parallel and through this can trade-off the variance in generated images vs. their conformity to the original training image;
2. we do not generate images at intermediate stages but propagate the image features directly from one stage to the next;
3. we improve the rescaling approach for multi-stage training, which enables us to train on fewer stages than before;
4. we introduce a fine-tuning phase which can be used on pre-trained models to obtain optimal results for specific images and tasks.

2 Related Work

Learning the statistics and distribution of patches of a single image has been known to provide a powerful prior for various vision tasks since the empirical entropy of patches inside a single image is smaller than the empirical entropy of patches inside a distribution of images [52]. By using this prior, many tasks such as inpainting [43,46], denoising [53], deblurring [37], retargeting [34,35], and segmentation [11] can be solved with only a single image. In particular, image super-resolution [44,20,12,40,4] and editing [7,9,15,36,42,33] from a single image have been shown to be successful and a large body of work focuses specifically on this task. Furthermore, recent work shows that training a

model on a single image with sufficient self-supervision and data augmentation can be enough to learn powerful feature extraction layers [1].

However, approaches that train GAN models on single images are still relatively rare. Most existing GAN approaches for single images do not use natural images, but instead train only on texture images [23,49,5,30]. At this time, two GAN models that are trained on a single 'natural' image, have been proposed, namely SinGAN [38] and InGAN [39]. Both of these approaches are based on a bidirectional similarity measure for image summarization which can be used for tasks such as retargeting, image synthesis, and automatic cropping [41].

The work most relevant to our approach is SinGAN [38] which trains a generative model to perform unconditional image generation on only a single image. As the empirical entropy of patches inside a small part of an image is less than across different parts, it is useful to learn statistics of image patches across different image scales [3]. Therefore, SinGAN trains both the generator and the discriminator over multiple stages, corresponding to different image resolutions. Each stage is trained individually and previous stages are kept fixed when higher stages are trained. The output at each stage is an image which is then given as input to the next higher stage. While SinGAN trains each stage in isolation, keeping the generators at all lower stages fixed, our experiments found that we can get improved results by training several, but not all, stages simultaneously.

Another single, natural image approach InGAN [39] follows a somewhat similar training procedure to learn the internal patch statistics of an image. However, InGAN's focus is retargeting and it cannot perform other tasks such as unconditional image generation. Furthermore, unlike SinGAN and our model, InGAN's generator and discriminator architectures stay fixed during training.

3 Methodology

We now describe our findings in more detail, starting with the training of a multi-stage architecture, followed by best practices we found for scaling learning rate and image resolutions at different stages during training.

Multi-stage Training It is clear that multi-scale image generation is of critical importance [38], however, there are many ways in which this can be realized. SinGAN only trains the current (highest) stage of its generator and freezes the parameters of all previous stages. ProGAN [25] presents a progressive growing scheme that adds levels with all weights unfrozen, and more recently [24,26] train the entire pyramid jointly.

In this work, we investigate whether the model can be trained end-to-end, rather than with training being fixed at intermediate stages, even in the single image task. However, we find that training all stages leads to *overfitting* (see Figure 4), i.e. the generator only generates the original training image without any variation or diversity. Reducing the learning rate at lower stages during training somewhat alleviates this problem, but does not solve it entirely. Instead, we develop a novel progressive growing technique that trains multiple, *but not all*, stages concurrently while simultaneously using progressively smaller learning rates at lower stages. Since we train several stages of our model concurrently for a single image we refer to our model as 'Concurrent-Single-Image-GAN' (ConSinGAN).

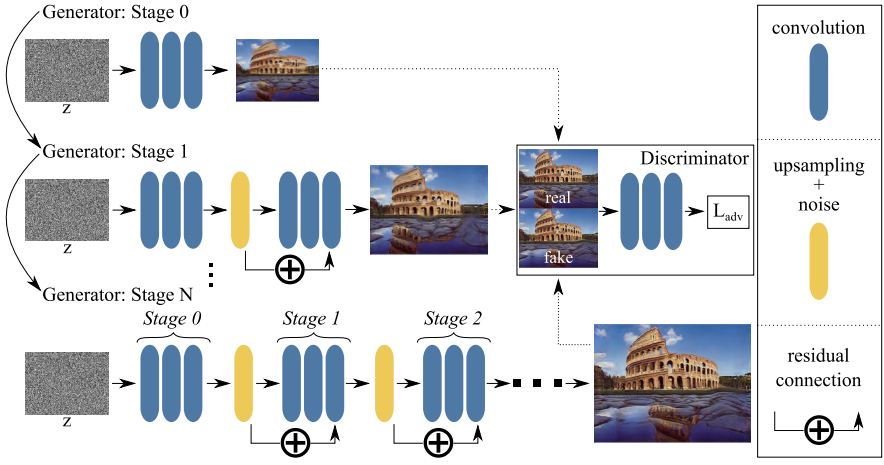

Fig. 1. Overview of our model. We start training at 'Stage 0' with a small generator and small image resolution. With increasing stages both the generator capacity and image resolution increase

Training ConSinGAN starts on a coarse resolution for a number of iterations, learning a mapping from a random noise vector z to a low-resolution image (see "Generator: Stage 0" in Figure 1). Once training of stage n has converged, we increase the size of our generator by adding three additional convolutional layers. In contrast to SinGAN, each stage gets the raw features from the previous stage as input, and previous layers are not fixed. We add a residual connection [17] from the original upsampled features to the output of the newly added convolutional layers (see "Generator: Stage 1" in Figure 1). We repeat this process N times until we reach our desired output resolution.

Like SinGAN we also add additional noise to the features at each stage [22,51] to improve diversity. In our default setting, we jointly train the last three stages of a generator (see "Generator: Stage N" in Figure 1), meaning the final three blocks of three convolutional layers. While it is possible to train more than three stages concurrently, we observed that this rapidly leads to severe overfitting (Figure 4).

We use the same patch discriminator [22] architecture and loss function as the original SinGAN. This means that the receptive field in relation to the size of the generated image gets smaller as the number of stages increases, meaning that the discriminator focuses more on global layout at lower resolutions and more on texture at higher resolutions. In contrast to SinGAN we do not increase the capacity of the discriminator at higher stages, but use the same number of parameters at every stage. As a consequence, we initialize the discriminator for a given stage n with the weights of the discriminator of the previous stage $n+1$, whereas SinGAN increases the discriminator's capacity every four stages and has to initialize the weights randomly at these stages. At a given stage n, we optimize the sum of an adversarial and a reconstruction loss:

$$\min_{G_n} \max_{D_n} \mathcal{L}_{adv}(G_n, D_n) + \alpha \mathcal{L}_{rec}(G_n). \tag{1}$$

$\mathcal{L}_{adv}(G_n, D_n)$ is the WGAN-GP adversarial loss [14], while the reconstruction loss is used to improve training stability ($\alpha = 10$ for all our experiments). For the reconstruction loss the generator G_n gets as input a downsampled version (x_0) of the original image (x_N) and is trained to reconstruct the image at the given resolution of stage n:

$$\mathcal{L}_{rec}(G_n) = ||G_n(x_0) - x_n||_2^2. \tag{2}$$

The discriminator is always trained in the same way, i.e. it gets as input either a generated or a real image and is trained to maximise $\mathcal{L}_{adv}$. Our generator, however, is trained slightly differently depending on the final task.

Task Specific Generator Training For each task we use the original (downsampled) image x_n for the reconstruction loss $\mathcal{L}_{rec}$. The input for the adversarial loss $\mathcal{L}_{adv}$, however, depends on the task we train for. For unconditional image generation (and as was done in SinGAN for all tasks), the input to the generator is simply a randomly sampled noise vector for $\mathcal{L}_{adv}$.

Fig. 2. Visualization of the harmonization augmentations techniques for images we use as input for the generator

However, we found that if the desired task is known beforehand, better results can be achieved by training with a different input format. For example, for image harmonization, we can instead train using the original image with augmentation transformations applied as input. The intuition for this is that a model that is used for image harmonization does not need to learn how to generate realistic images from random noise, but rather should learn how to harmonize different objects and color distributions into the image background. To simulate this task, we apply random combinations of augmentation techniques to the original image x_N at each iteration, sampled from the following methods: additive noise, color transforms, cutout holes into the image filled with a random color (see Figure 2). The generator then gets the augmented image as input and needs to transform it back to an image that should resemble the original distribution, which is trained through the adversarial loss.

Learning Rate Scaling The space of all learning rates for each stage is large and has a big impact on the final image quality. At any given stage n, we found that instead of training all stages (n, $n-1$, $n-2$, ...) with the same learning rate, using a lower learning rate on earlier stages ($n-1$, $n-2$, ...) helps reduce overfitting. If the learning rate at lower stages is too large (or too many stages are trained concurrently), the model generator quickly collapses and only generates the training image (Figure 4). Therefore, we propose to scale the learning rate η with a factor δ. This means that for generator G_n stage n is trained with learning rate $\delta^0\eta$, stage $n-1$ is trained with a learning rate $\delta^1\eta$, stage $n-2$ with $\delta^2\eta$, etc. In our experiments, we found that setting $\delta = 0.1$ gives a good trade-off between image fidelity and diversity (see Figure 4 and Figure 5).

Improved Image Rescaling Another critical design choice is around what kind of multiscale pyramid to use. SinGAN originally proposes to downsample the image x by a factor of r^{N-n} for each stage n where r is a scalar with default value 0.75. As a result, eight to ten stages are usually needed to generate images with a resolution of 250 width

or height. When the images are downsampled more aggressively (e.g. $r = 0.5$) fewer stages are needed, but the generated images lose much of their global coherence.

We observe that this is the case when there are not enough stages at low resolution (roughly fewer than 60 pixels at the longer side). When training on images with a high resolution, the global layout is already "decided" and only texture information is important since the discriminator's receptive field is always 11×11. Consequently, to achieve a certain global image layout we need a certain number of stages (usually at least three) at low resolution, but we do not need many stages a high resolution. We adapt the rescaling to not be strictly geometric (i.e. $x_n = x_0 \times r^{N-n}$), but instead to keep the density of low-resolution stages higher than the density of high-resolution stages:

$$x_n = x_N \times r^{((N-1)/log(N))*log(N-n)+1} \text{ for } n = 0, ..., N-1 \tag{3}$$

For example, with the original rescaling method and a rescaling scalar $r = 0.55$ we get six stages with resolutions $25 \times 34, 38 \times 50, 57 \times 75, 84 \times 112, 126 \times 167, 188 \times 250$. In contrast to this, using our rescaling method with $r = 0.55$ leads to six stages with resolutions $25 \times 34, 32 \times 42, 42 \times 56, 63 \times 84, 126 \times 167, 188 \times 250$.

To summarize our main findings, we produce feature maps rather than images at each stage, we train multiple stages concurrently, we propose a modified rescaling pyramid, and we present a task-specific training variation.

4 Implementation

We next describe the key parts of our implementation, and refer to the supplementary material for further details.[1] 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. Each stage of our generator consists of three convolutional layers, with 64 filters per layer and a filter size of 3×3. The discriminator uses the same architecture, i.e. three convolutional layers with 64 filters and filter size 3×3.

At any given time, we train the last three stages of the model with the learning rate scaling $\delta = 0.1$. Each stage is trained for 2,000 iterations with an initial learning rate of $\eta = 0.0005$ which gets reduced by a factor of 10 after 1600 iterations in each stage. We optimize the networks with the ADAM optimizer [28]. For unconditional image generation, we find that training without batch normalization (BN) [21] speeds up training considerably without negatively affecting the convergence. For other tasks we find that BN is still useful, however, we only use it in the generator.

We use the Leaky ReLU (LReLU) activation function [32]. 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.

Training one of our models takes about 20-30 minutes on an NVIDIA GeForce GTX 1080Ti. On the other hand, training the SinGAN model takes on average 120-140 minutes on the same hardware.

[1] Our implementation is available here: `https://github.com/tohinz/ConSinGAN`

Fig. 3. Example of unconditionally generated images showing complex global structure generated by ConSinGAN, trained on a single image shown in the left column

5 Results

We evaluate ConSinGAN on unconditional image generation and image harmonization in detail. For space reasons we focus on these two applications but note that all other applications presented by SinGAN are also possible with ConSinGAN. We show examples of other tasks in the supplementary material.

5.1 Unconditional Image Generation

In this task, images are generated from a randomly sampled latent vector. Since our architecture is completely convolutional we can change the size of the input noise vector to generate images of various resolutions at test time. Figure 3 shows an overview of results from our method on a set of challenging images that require the generation of *global* structures for the images to seem realistic. We observe that ConSinGAN is successfully able to capture these global structures, even if we modify the image resolution at test time. For example, in the Stonehenge example, we can see how "stones" are added when the image width is increased. Similarly, our model adds additional "layers" to the aqueduct image when the image height is increased.

Ablation We conduct an ablation study to examine the interplay between the learning rate scaling and the number of concurrently trained stages (Figure 4) and to evaluate how varying the scaling δ (section 3) for the learning rate of lower stages affects training (Figure 5). As we can see in Figure 4, training with a $\delta = 0.1$ leads to diverse images for almost all settings, with the diversity slightly decreasing with a larger number of concurrently trained stages. When training with $\delta = 0.5$, however, we quickly observe a

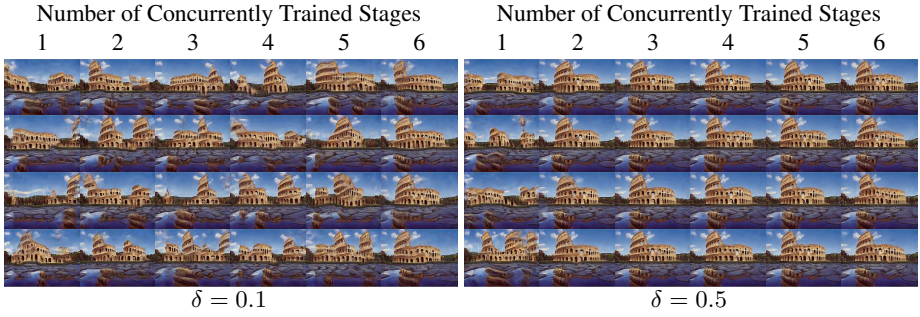

Fig. 4. Effect of the learning rate scale δ and the number of concurrently trained stages for a model with six stages. Images are randomly selected.

large decrease in image diversity even when only training two stages concurrently. As such, the number of concurrently trained stages and the learning rate scaling parameter δ give a trade-off between diversity and fidelity of the generated images.

Figure 5 visualizes how the variance in the generated images increases with decreasing δ for a model with three concurrently trained stages. For example, when we look at the top left example (Marina Bay Sands), we observe that for a $\delta = 0.5$ the overall layout of the image stays the same, with minor variations in, e.g., the appearance of the towers. However, with a $\delta = 0.1$, the appearance of the towers changes more drastically and sometimes even additional towers are added to the generated image. The same is true for the other examples, e.g. observe the differences in the layout of the bridge in the top right example. Unless otherwise mentioned, all illustrated examples and all images used for the user study where generated by models for which we trained three stages concurrently with $\delta = 0.1$.

Baseline comparisons We compare our model to the original SinGAN [38] paper in Figure 6. For the SinGAN model, we show the results of both the default rescaling method

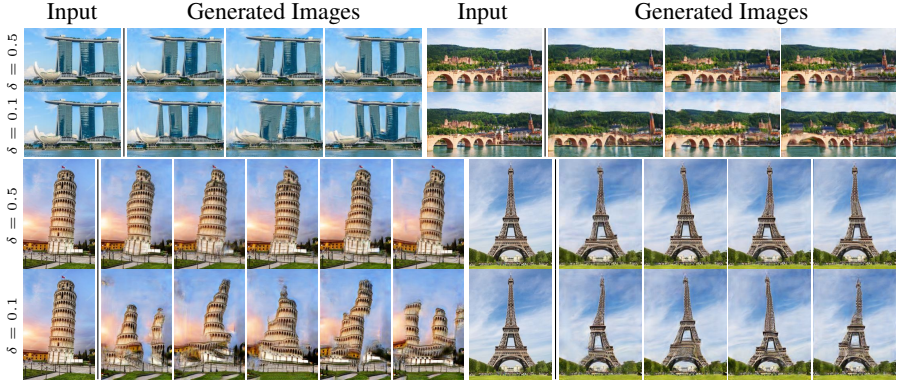

Fig. 5. Effect of the learning rate scale δ during training of ConSinGAN

with 8-10 stages, with our rescaling method with 5-6 stages, and all our modifications, trained with 5-6 stages. In the first example (Mount Rushmore) we can observe that SinGAN struggles to model recurring structures (faces) in the images it generates. This is the case for both numbers of stages during training, with slightly better results when training for the full eight stages. In the second example (Leaning Tower of Pisa) we again observe a loss of global structure independent of the number of stages trained. Our multi-stage training helps ensure a more consistent global structure.

Figure 7 further highlights the advantages of our approach by showing a detailed comparison of the kinds of images each model generates after being trained on a given number of stages with the new or old rescaling technique. Each column depicts four randomly sampled images from each model. We can see the positive effect of the rescaling technique for both models, regardless of the number of trained stages. Furthermore, we can see that our model retains better global coherence in both cases.

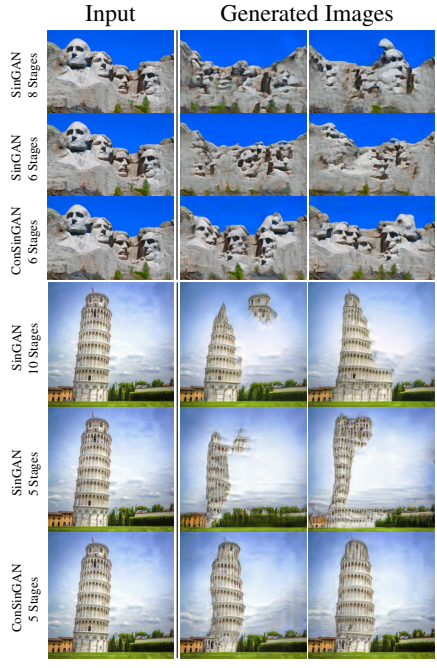

Fig. 6. Comparison of images generated by SinGAN and ConSinGAN

Quantitative evaluation The Fréchet Inception Distance (FID) [18] compares the distribution of a pre-trained network's activations between a sets of generated and real images. Prior work proposed the Single Image FID (SIFID) metric to evaluate generation quality [38]. SIFID is an adaptation of the FID to the single image domain and compares the statistics of the network's activations between two individual images (generated and real). In our experiments, we found that SIFID exhibits very high variance across different images (scores ranged between $1e-06$ to $1e01$) without a clear distinction of which was "better" or "worse". In this work, we focus mostly on qualitative analyses and user studies for our evaluation but also report SIFID for comparison.

To quantitatively evaluate our model, we performed evaluations on two datasets. The first dataset is the same as the one used by SinGAN, consisting of 50 images from several categories of the 'Places' dataset [48]. However, many of these images do not exhibit a global layout or structure. Therefore, we also construct a second dataset, where we take five random samples from each of the ten classes of the LSUN dataset [45]. This dataset contains classes such as "church" and "bridge" which exhibit more global structures. We train both the SinGAN model and our model for each of the 50 images in both datasets and use the results for our evaluation. As we can see in Table 1 and Table 2 our model was trained on fewer stages on all images and was also faster during training.

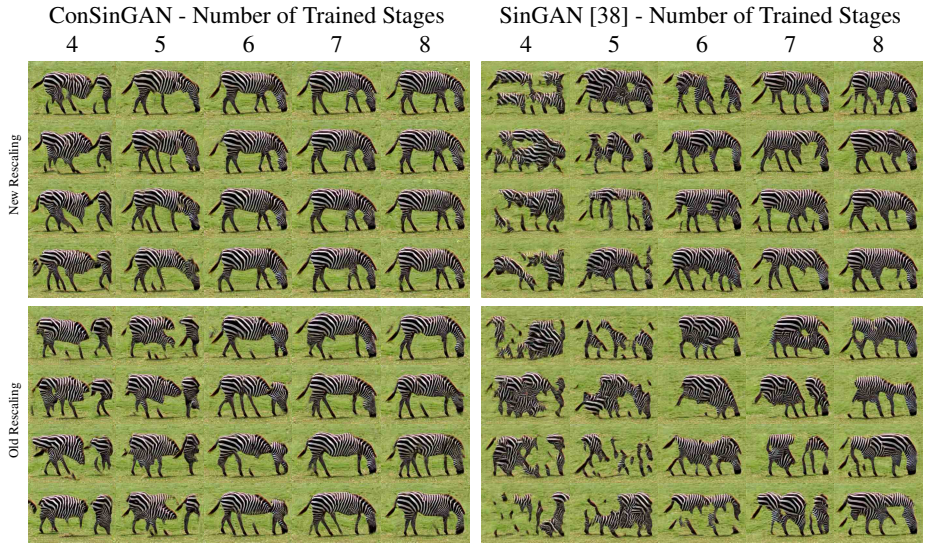

Fig. 7. Detailed comparison of the effect of the number of trained stages and rescaling method during training. Images are randomly selected

Image Diversity First, we evaluate the diversity in our images compared to the original SinGAN model using the measure as SinGAN: for a given training image we calculate the average of the standard deviation of all pixel values along the channel axis of 100 generated images. Then, we normalize this value by the standard deviation of the pixel values in the training image. On the data from the 'Places' dataset, SinGAN obtains a diversity score of 0.52, while our model's diversity is similar with a score of 0.50. When we increase the learning rate on lower stages by setting $\delta = 0.5$ instead of the default $\delta = 0.1$ we observe a lower diversity score of 0.43 as the model learns a more precise representation of the training image (see Figure 5 for a visualization). On the LSUN data, SinGAN obtains a much higher diversity score of 0.64. This is due to the fact that it often fails to model the global structure of the training image and the resulting generated images differ greatly from the training image. Our model, on the other hand, obtains a diversity score of 0.54 which is similar to the score on the 'Places' dataset and indicates that our model can indeed learn the global structure of even complex images.

User Study: 'Places' We follow the same evaluation procedure as previous work [22,38,47] to compare our model with SinGAN on the same training images that were used previously in [38]. Users were shown our generated image and its respective training image for one second each and were asked to identify the real image. We reproduced the user study from the SinGAN paper with our own trained SinGAN and ConSinGAN models. As we can see in Table 1 our model achieves results similar to the SinGAN model (in the original work, SinGAN reports a confusion of 21.45% with their models/images). However, our model is trained on fewer stages and with fewer parameters and our model obtains a better SIFID score of 0.06, compared to SinGAN's 0.09. Furthermore, the

Table 1. Results of our user study and SIFID on images from the **Places** dataset.

Model	Confusion ↑	SIFID ↓	Train Time	# Stages	# Params
ConSinGAN	$16.0\% \pm 1.4\%$	0.06 ± 0.03	24 min	5.9	~660,000
SinGAN	$17.0\% \pm 1.5\%$	0.09 ± 0.07	152 min	9.7	~1,340,000

images generated by ConSinGAN often still exhibit a better global structure, but one second is not enough time for users to identify this.

User Study: 'LSUN' Since the images from the LSUN dataset are much more challenging than the images from the 'Places' dataset we do not compare the generated images against the real images, but instead compare the images generated by SinGAN to the ones generated by ConSinGAN. We generate 10 images per training image, resulting in 500 generated images each from SinGAN and ConSinGAN. We use the generated images to compare the models in two different user studies.

In both versions, the participants see the two images generated by the two models next to each other and need to judge which image is better. We do not enforce a time limit, so participants can look at both images for as long as they choose. The difference between the two versions of the user study is how we sample the generated images. In the first version ("random") we randomly sample one image from the set of generated images of SinGAN and ConSinGAN each. This means that the two images likely come from different classes (e.g. 'church' vs. 'conference room'). In the second version ("paired") we sample two images that were generated from the same training image. We perform both user studies using Amazon Mechanical Turk, with 50 participants comparing 60 pairs of images for each study.

Table 2 shows how often users picked images generated by one model or the other for each of the two settings. We can see that users prefer the images generated by ConSinGAN in both settings and that, again, our model achieves a better SIFID score than SinGAN. This is the case even though our model only trains on six stages, has fewer parameters than SinGAN, and takes less time to train. The images from the LSUN data vary in difficulty and global structure. This might be the reason why our model performs even better in the paired setting since this setting guarantees that we always compare the two models on images of the same difficulty. In the random setting, however, it can happen that we compare a SinGAN model from an 'easy' image with a ConSinGAN model from a 'difficult' image.

Table 2. Results of our user studies and SIFID on images from the **LSUN** dataset.

Model	Random ↑	Paired ↑	SIFID ↓	Train Time	# Stages	# Params
ConSinGAN	$56.7\% \pm 1.9\%$	$63.1\% \pm 1.8\%$	0.11 ± 0.06	20 min	5.9	~660K
SinGAN	$43.3\% \pm 1.9\%$	$36.9\% \pm 1.8\%$	0.23 ± 0.15	135 min	9.1	~1.0M

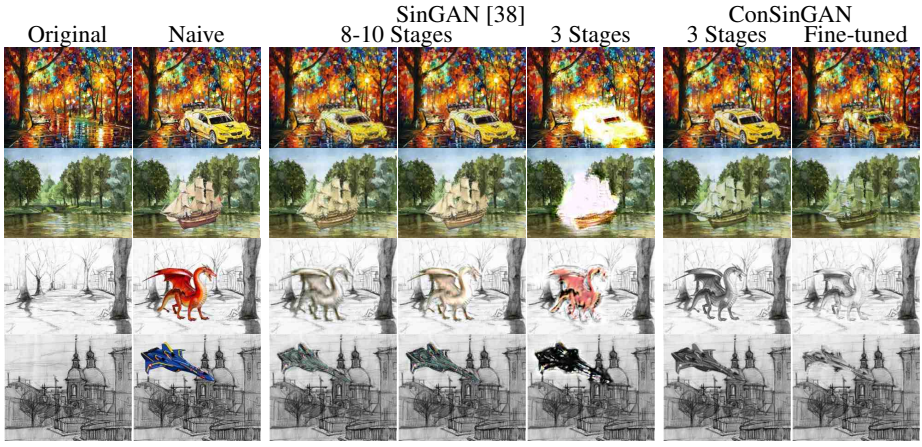

Fig. 8. Image harmonization with SinGAN and ConSinGAN

Overall, our experiments show that the modifications we made in ConSinGAN allow for the generation of more believable images, especially when they exhibit some degree of global structure, with less training time and a smaller model in general.

5.2 Image Harmonization

We now show our results on several image harmonization examples. We compare our results to SinGAN and Deep Painterly Harmonization [31] for high-resolution images.

Training Details We train the ConSinGAN with the same hyperparameters for all images without any fine-tuning of hyperparameters for the different images. The general architecture is the same as for the unconditional image generation, however, we only train the model for exactly three stages per image. During the general training we train for 1,000 iterations per stage and we randomly sample from some different data augmentation techniques to obtain a "new" training image at each iteration as described in section 3. When we fine-tune a model on a given specific image (last column in all figures) we use a model trained on the general style image and use the target image directly as input (instead of the style image with random augmentation transformations) and then train the model for an additional 500 iterations.

Comparison with SinGAN Figure 8 shows comparisons between SinGAN and ConSinGAN on the image harmonization task. The first two columns of each figure show the original image we trained on and the naive cut-and-paste image that is the input to our trained model. The next three images show the results of a trained SinGAN model, where the first two are the results of a fully trained model. We insert the naive image at all stages of the model and choose the two best results, while the third image is the result when we train SinGAN on only three stages. The final two columns show the results of the ConSinGAN. Training the ConSinGAN model takes less than 10 minutes for a given image when the coarse side of the image has a resolution of 250 pixels. Fine-tuning a model on a specific image takes roughly 2-3 minutes. In contrast to this, training the

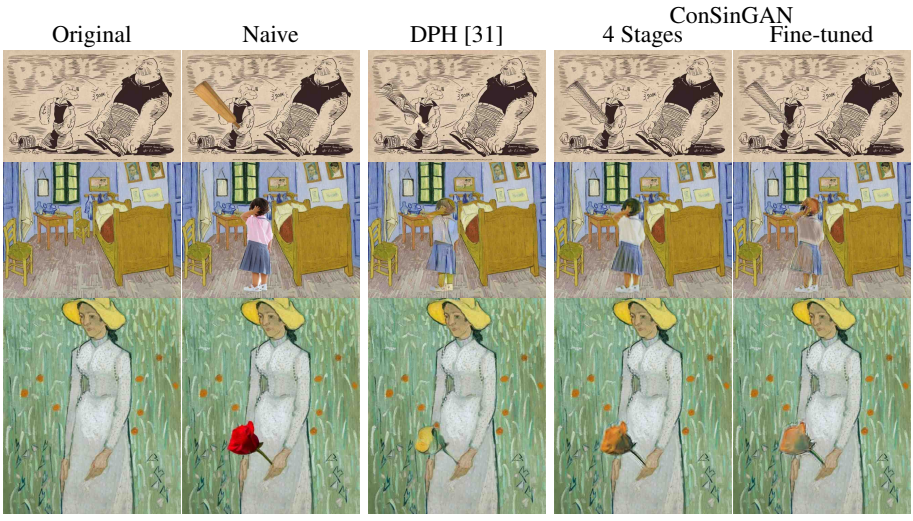

Fig. 9. Image harmonization comparison with Deep Painterly Harmonization (DPH) and ConSinGAN on high resolution images

SinGAN model takes roughly 120 minutes as before, since we need to train the full model, even if only some of the later stages are used at test time.

We can see that the ConSinGAN performs similarly or better to SinGAN, even though we only need to train the ConSinGAN for 3 stages. The ConSinGAN also generally introduces fewer artifacts into the harmonized image, while SinGAN often changes the surface structure of the added objects. See for example the first row, where SinGAN adds artifacts onto the car, while ConSinGAN keeps the original objects much more consistent, while still adapting it to the overall image style. When we fine-tune the ConSinGAN model on specific images we can get even more interesting results. For example, the car gets absorbed much more by the colors of the overall background, compared to the results of SinGAN or the ConSinGAN which was not fine-tuned.

The bottom two rows of Figure 8 show results when we add colorful objects to black-and-white paintings. When training SinGAN on only three stages like ConSinGAN it usually fails completely to harmonize the objects at test time. Even the images harmonized after training on 8-10 stages SinGAN often still contain some of the original colors, while the ConSinGAN manages to completely transfer the objects to black-and-white versions. Again, further fine-tuning the ConSinGAN on the specific images leads to an even stronger "absorption" of the objects into the original image in most cases.

Comparison with DPH Figure 9 shows comparisons between the ConSinGAN, adapted to harmonize high-resolution images, and the results by Deep Painterly Harmonization (DPH) [31]. The images have a resolution of roughly 700 pixels on the longer side, as opposed to the images with 250 pixels used by the SinGAN examples. In order to produce these high-resolution images, we add another stage to our ConSinGAN architecture. This means that we now train on four stages, instead of three stages as before, otherwise the training procedure stays the same. Since we use images with a

high-resolution, training time increases to roughly 30-40 minutes per image. Again, we use the same architecture, hyperparameters, and training method for each image. This is in contrast to many style-transfer approaches and also DPH, which have additional hyperparameters such as the style and content weight which need to be fine-tuned for a specific style image for optimal results.

We can see that, while the outputs of ConSinGAN usually differ from the outputs of DPH, they are still realistic and visually pleasing. This is even the case when our model has never seen the naive copy-and-paste image at train time, but only uses it at test time. In contrast to this, DPH requires as input the style input, the naive copy-and-past input, and the mask which specifies the location of the copied object in the image. DPH is then trained for this specific combination to obtain pleasing results, whereas our model sees the copy-and-paste image only at test time, but not at training time.

Again, fine-tuning our model sometimes leads to even better results, but even the model trained only with random image augmentations performs well. While our training time is quite long, we only need to train our model once for a given image and can then add different objects at different locations of the image at test time. This is not possible with DPH, which requires to be retrained whenever the copied object changes.

6 Conclusion

Finally, Figure 10 shows some examples where our model fails to learn the global layout of the image. We see that the model still learns many of the features that are present in the original training example. However, it is not able to structure them in a way that reflects the global layout of the original image. Instead, the order of the individual "parts" of the object get "shuffled" around (top two rows) or we have floating pieces that are not connected when they should be (bottom row).

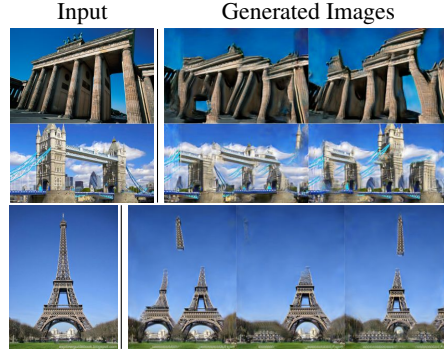

Fig. 10. Example of failure cases of ConSinGAN

We introduced ConSinGAN, a GAN inspired by a number of best practices discovered for training single image GANs. Our model is trained on sequentially increasing image resolutions, to first learn the global structure of the image, before learning texture and stylistic details later. Compared to other models, our approach allows for control over how closely the internal patch distribution of the training image is learned by adjusting the number of concurrently trained stages and the learning rate scaling of lower stages. Through this, we can decide how much diversity we want to observe in the images generated with the trained model. We show that our approach can be trained on a single image and can be used for tasks such as unconditional image generation and image harmonization. Our model generates more globally coherent results and is smaller and more efficient to train than previous models.

Acknowledgements

The authors gratefully acknowledge partial support from the German Research Foundation DFG under project CML (TRR 169). We also thank the NVIDIA Corporation for their support through the GPU Grant Program.

References

1. Asano, Y.M., Rupprecht, C., Vedaldi, A.: Surprising effectiveness of few-image unsupervised feature learning. In: International Conference on Learning Representations (2020)
2. Ba, J.L., Kiros, J.R., Hinton, G.E.: Layer normalization. arXiv preprint arXiv:1607.06450 (2016)
3. Bagon, S., Boiman, O., Irani, M.: What is a good image segment? a unified approach to segment extraction. In: European Conference on Computer Vision. pp. 30–44. Springer (2008)
4. Bell-Kligler, S., Shocher, A., Irani, M.: Blind super-resolution kernel estimation using an internal-gan. In: Advances in Neural Information Processing Systems. pp. 284–293 (2019)
5. Bergmann, U., Jetchev, N., Vollgraf, R.: Learning texture manifolds with the periodic spatial gan. In: International Conference on Machine Learning. pp. 469–477. JMLR. org (2017)
6. Brock, A., Donahue, J., Simonyan, K.: Large scale gan training for high fidelity natural image synthesis. In: International Conference on Learning Representations (2019)
7. Cho, T.S., Butman, M., Avidan, S., Freeman, W.T.: The patch transform and its applications to image editing. In: Computer Vision and Pattern Recognition. pp. 1–8. IEEE (2008)
8. 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)
9. Dekel, T., Michaeli, T., Irani, M., Freeman, W.T.: Revealing and modifying non-local variations in a single image. ACM Transactions on Graphics (TOG) **34**(6), 1–11 (2015)
10. Demir, U., Unal, G.: Patch-based image inpainting with generative adversarial networks. arXiv preprint arXiv:1803.07422 (2018)
11. Gandelsman, Y., Shocher, A., Irani, M.: Double-dip": Unsupervised image decomposition via coupled deep-image-priors. In: Computer Vision and Pattern Recognition. vol. 6, p. 2 (2019)
12. Glasner, D., Bagon, S., Irani, M.: Super-resolution from a single image. In: International Conference on Computer Vision. pp. 349–356. IEEE (2009)
13. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: Advances in Neural Information Processing Systems. pp. 2672–2680 (2014)
14. 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)
15. He, K., Sun, J.: Statistics of patch offsets for image completion. In: European Conference on Computer Vision. pp. 16–29. Springer (2012)
16. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: International Conference on Computer Vision. pp. 1026–1034 (2015)
17. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Computer Vision and Pattern Recognition. pp. 770–778 (2016)
18. Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., Hochreiter, S.: Gans trained by a two time-scale update rule converge to a local nash equilibrium. In: Advances in Neural Information Processing Systems. pp. 6626–6637 (2017)

19. Hinz, T., Heinrich, S., Wermter, S.: Generating multiple objects at spatially distinct locations. In: International Conference on Learning Representations (2019)
20. Huang, J.B., Singh, A., Ahuja, N.: Single image super-resolution from transformed self-exemplars. In: Computer Vision and Pattern Recognition. pp. 5197–5206 (2015)
21. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: International Conference on Machine Learning. pp. 448–456 (2015)
22. Isola, P., Zhu, J.Y., Zhou, T., Efros, A.A.: Image-to-image translation with conditional adversarial networks. In: Computer Vision and Pattern Recognition. pp. 1125–1134 (2017)
23. Jetchev, N., Bergmann, U., Vollgraf, R.: Texture synthesis with spatial generative adversarial networks. arXiv preprint arXiv:1611.08207 (2016)
24. Karnewar, A., Wang, O.: Msg-gan: Multi-scale gradients for generative adversarial networks. In: Computer Vision and Pattern Recognition (2020)
25. 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)
26. Karras, T., Laine, S., Aila, T.: A style-based generator architecture for generative adversarial networks. In: Computer Vision and Pattern Recognition. pp. 4401–4410 (2019)
27. Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J., Aila, T.: Analyzing and improving the image quality of stylegan. arXiv preprint arXiv:1912.04958 (2019)
28. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: International Conference on Learning Representations (2015)
29. Klambauer, G., Unterthiner, T., Mayr, A., Hochreiter, S.: Self-normalizing neural networks. In: Advances in Neural Information Processing Systems. pp. 971–980 (2017)
30. Li, C., Wand, M.: Precomputed real-time texture synthesis with markovian generative adversarial networks. In: European Conference on Computer Vision. pp. 702–716. Springer (2016)
31. Luan, F., Paris, S., Shechtman, E., Bala, K.: Deep painterly harmonization. Computer Graphics Forum **37**(4), 95–106 (2018)
32. 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)
33. Mao, J., Zhang, X., Li, Y., Freeman, W.T., Tenenbaum, J.B., Wu, J.: Program-guided image manipulators. In: International Conference on Computer Vision. pp. 4030–4039 (2019)
34. Mastan, I.D., Raman, S.: Multi-level encoder-decoder architectures for image restoration. arXiv preprint arXiv:1905.00322 (2019)
35. Mastan, I.D., Raman, S.: Dcil: Deep contextual internal learning for image restoration and image retargeting. In: The IEEE Winter Conference on Applications of Computer Vision. pp. 2366–2375 (2020)
36. Mechrez, R., Shechtman, E., Zelnik-Manor, L.: Saliency driven image manipulation. Machine Vision and Applications **30**(2), 189–202 (2019)
37. Michaeli, T., Irani, M.: Blind deblurring using internal patch recurrence. In: European Conference on Computer Vision. pp. 783–798. Springer (2014)
38. Shaham, T.R., Dekel, T., Michaeli, T.: Singan: Learning a generative model from a single natural image. In: International Conference on Computer Vision. pp. 4570–4580 (2019)
39. Shocher, A., Bagon, S., Isola, P., Irani, M.: Ingan: Capturing and retargeting the "dna" of a natural image. In: International Conference on Computer Vision (2019)
40. Shocher, A., Cohen, N., Irani, M.: zero-shot super-resolution using deep internal learning. In: Computer Vision and Pattern Recognition. pp. 3118–3126 (2018)
41. Simakov, D., Caspi, Y., Shechtman, E., Irani, M.: Summarizing visual data using bidirectional similarity. In: Computer Vision and Pattern Recognition. pp. 1–8. IEEE (2008)
42. Tlusty, T., Michaeli, T., Dekel, T., Zelnik-Manor, L.: Modifying non-local variations across multiple views. In: Computer Vision and Pattern Recognition. pp. 6276–6285 (2018)

43. Ulyanov, D., Vedaldi, A., Lempitsky, V.: Deep image prior. In: Computer Vision and Pattern Recognition. pp. 9446–9454 (2018)
44. Yang, W., Zhang, X., Tian, Y., Wang, W., Xue, J.H., Liao, Q.: Deep learning for single image super-resolution: A brief review. IEEE Transactions on Multimedia **21**(12), 3106–3121 (2019)
45. 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)
46. Zhang, H., Mai, L., Xu, N., Wang, Z., Collomosse, J., Jin, H.: An internal learning approach to video inpainting. In: International Conference on Computer Vision. pp. 2720–2729 (2019)
47. Zhang, R., Isola, P., Efros, A.A.: Colorful image colorization. In: European Conference on Computer Vision. pp. 649–666. Springer (2016)
48. 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)
49. Zhou, Y., Zhu, Z., Bai, X., Lischinski, D., Cohen-Or, D., Huang, H.: Non-stationary texture synthesis by adversarial expansion. ACM Transactions on Graphics (TOG) **37**(4), 1–13 (2018)
50. Zhu, J.Y., Park, T., Isola, P., Efros, A.A.: Unpaired image-to-image translation using cycle-consistent adversarial networks. In: International Conference on Computer Vision. pp. 2223–2232 (2017)
51. Zhu, J.Y., Zhang, R., Pathak, D., Darrell, T., Efros, A.A., Wang, O., Shechtman, E.: Toward multimodal image-to-image translation. In: Advances in Neural Information Processing Systems. pp. 465–476 (2017)
52. Zontak, M., Irani, M.: Internal statistics of a single natural image. In: Computer Vision and Pattern Recognition. pp. 977–984. IEEE (2011)
53. Zontak, M., Mosseri, I., Irani, M.: Separating signal from noise using patch recurrence across scales. In: Computer Vision and Pattern Recognition. pp. 1195–1202 (2013)

A Unconditional Generation

Figure 11 shows more examples of unconditional image generation for both ConSinGAN and SinGAN [38]. 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.

B Unconditional Generation at Arbitrary Sizes

Figure 12 and Figure 13 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.

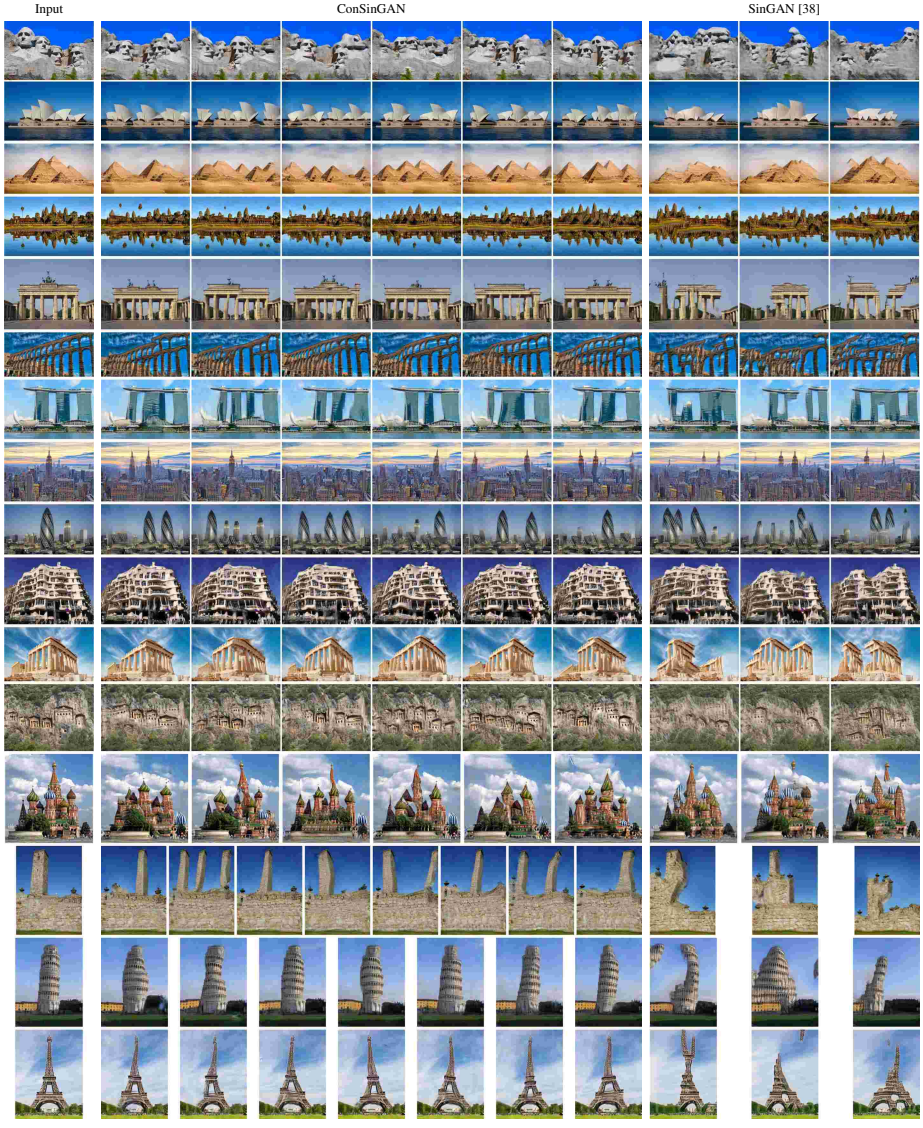

Fig. 11. Unconditional image generation with ConSinGAN and SinGAN. All ConSinGAN models were trained on six stages, all SinGAN models were trained on eight – ten stages. We can see how ConSinGAN is able to model the global image structure better in most cases, despite being trained on fewer stages than SinGAN

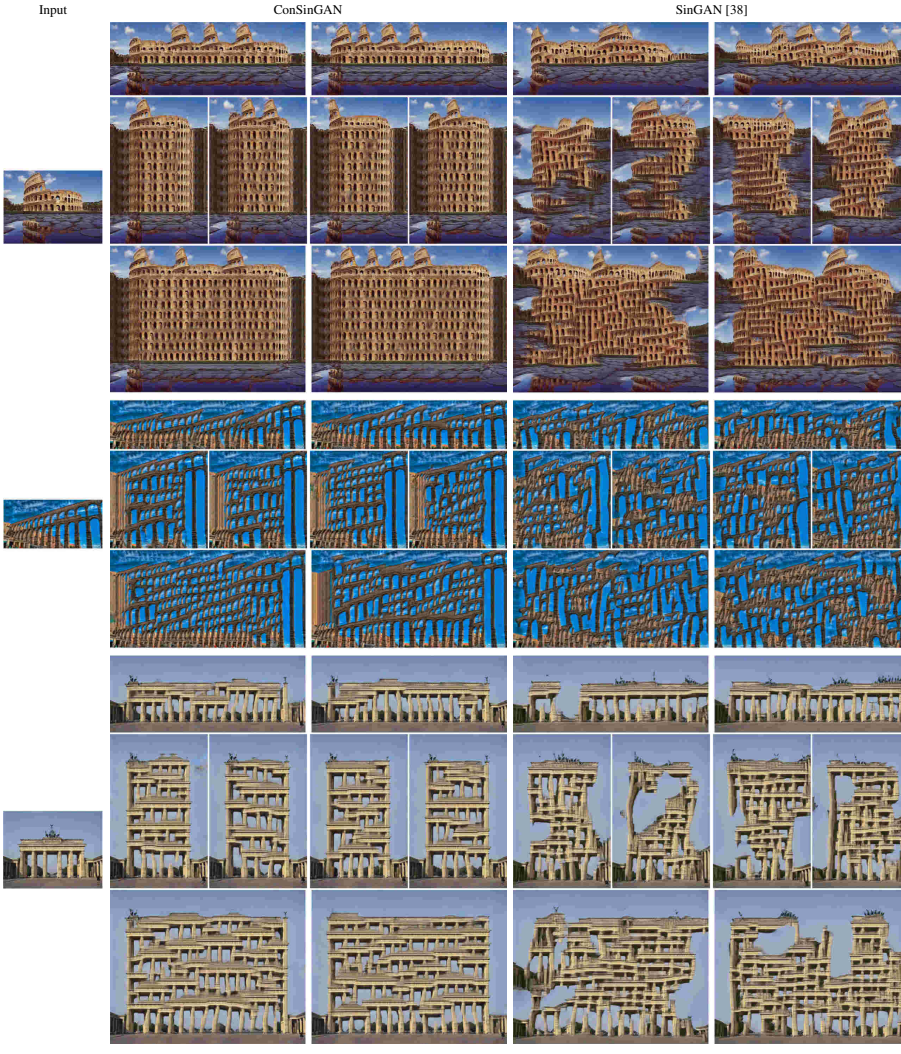

Fig. 12. Unconditional image generation with ConSinGAN and SinGAN. We show images were we scale the input noise map by a factor of two along each side and along both sides. The ConSinGAN models were trained on six stages, while the SinGAN models were trained on eight – ten stages

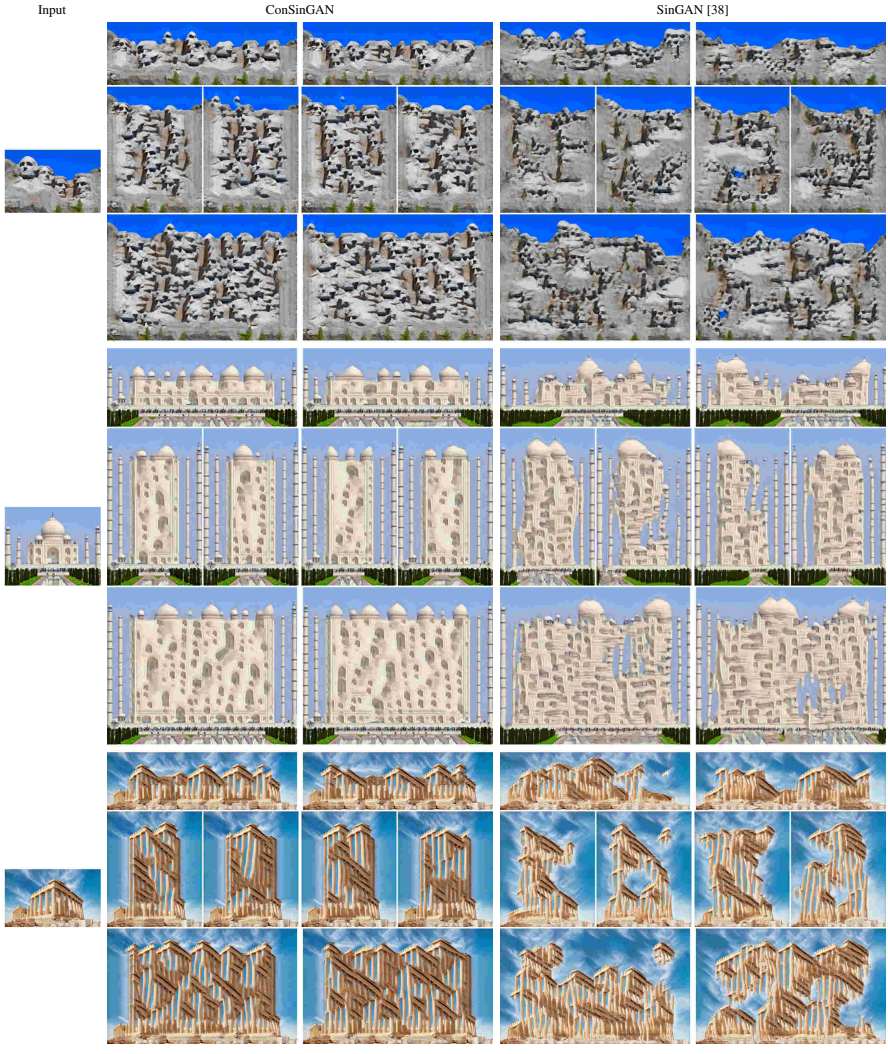

Fig. 13. Unconditional image generation with ConSinGAN and SinGAN. We show images were we scale the input noise map by a factor of two along each side and along both sides. The ConSinGAN models were trained on six stages, while the SinGAN models were trained on eight – ten stages

C Number of Concurrently Trained Stages and Learning Rate Scaling

Figure 14 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.

D Comparison of Original and Improved Rescaling Method

Figure 15 and Figure 16 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.

Fig. 14. 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

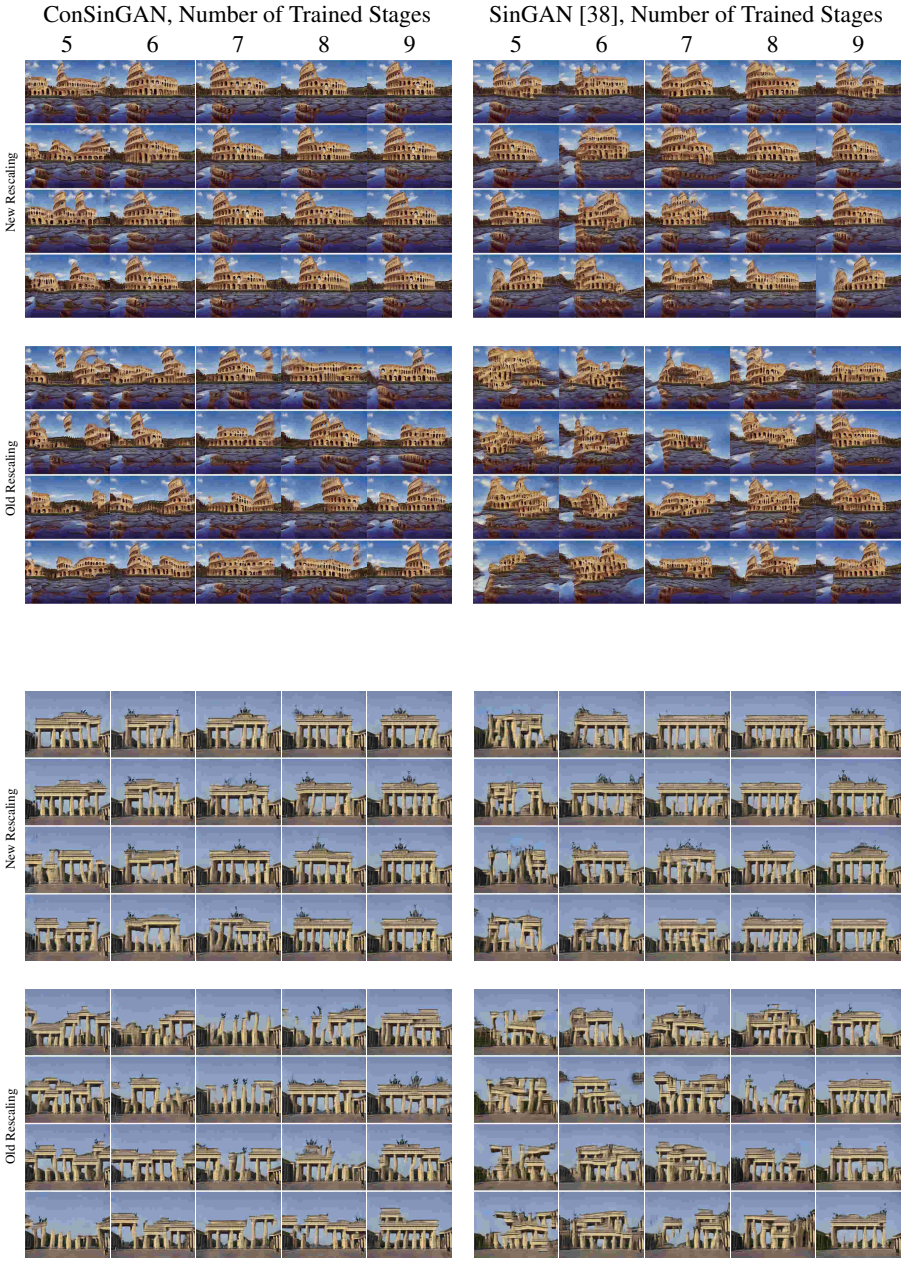

Fig. 15. Comparison between our updated and the original rescaling method. We can see that both models benefit from the updated rescaling method and can generate realistic images with fewer stages. All images are randomly sampled

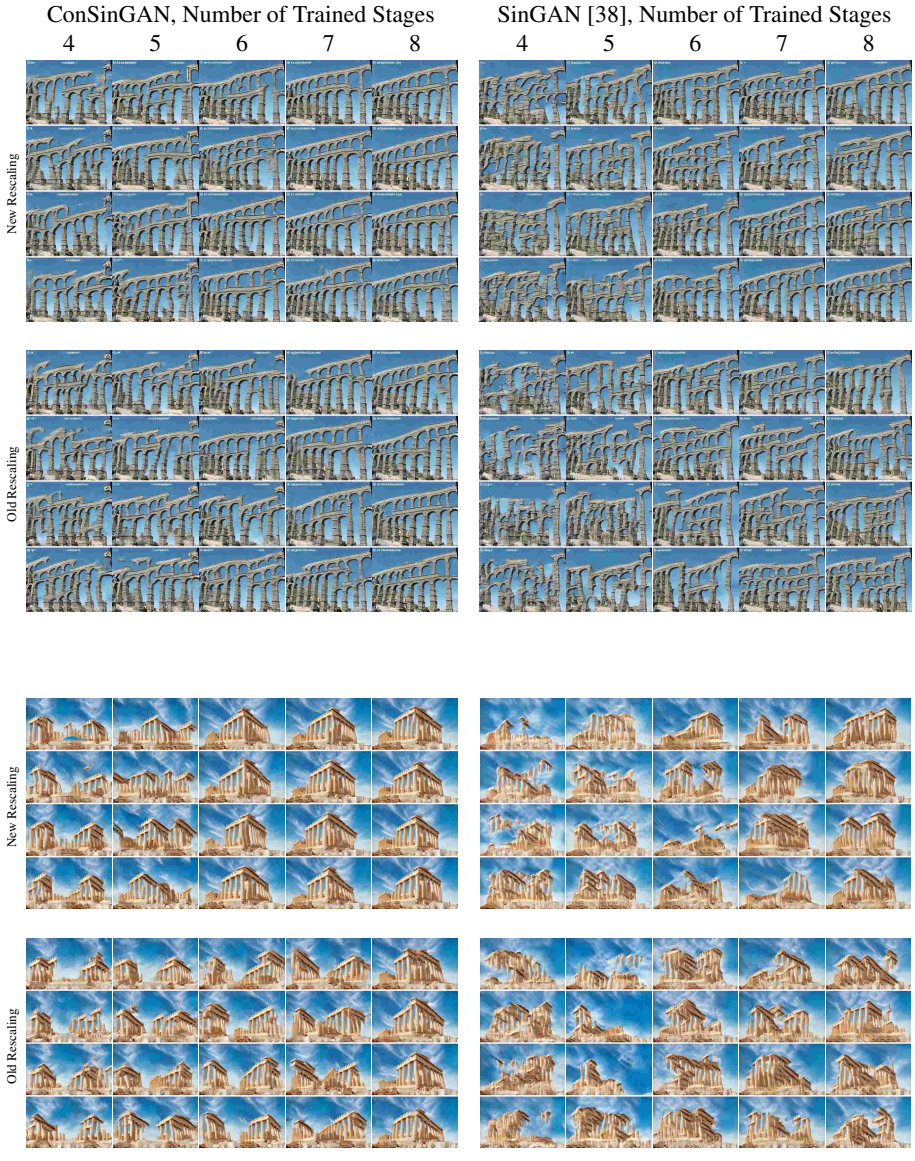

Fig. 16. Comparison between our updated and the original rescaling method. We can see that both models benefit from the updated rescaling method and can generate realistic images with fewer stages. All images are randomly sampled

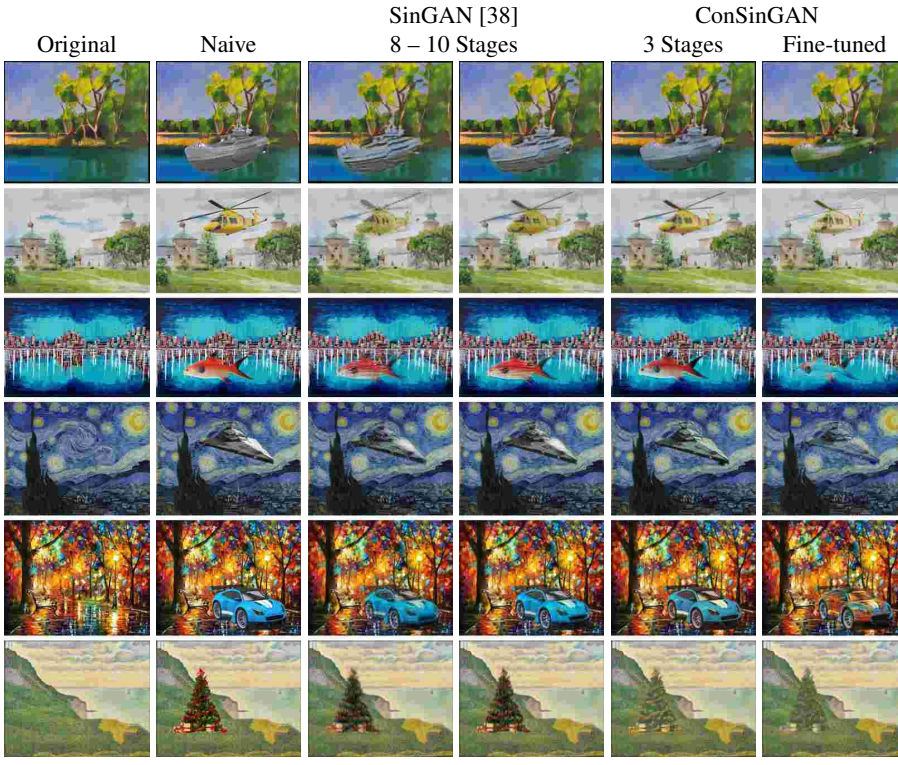

Fig. 17. Comparison of ConSinGAN and SinGAN on image /supp-harmonization. Our model often produces better results despite being trained on fewer stages

E Image Harmonization

Figure 17 and Figure 18 show 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 19, Figure 20, and Figure 21 show more comparisons between ConSinGAN and Deep Painterly Harmonization (DPH) [31] 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).

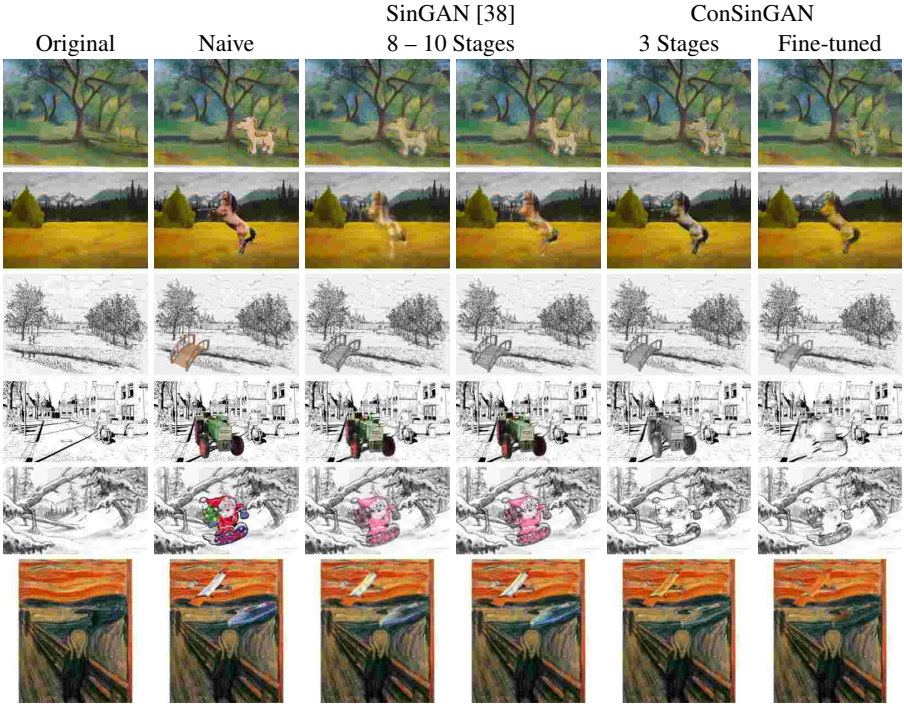

Fig. 18. Comparison of ConSinGAN and SinGAN on image supp-harmonization. Our model often produces better results despite being trained on fewer stages

Fig. 19. 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)

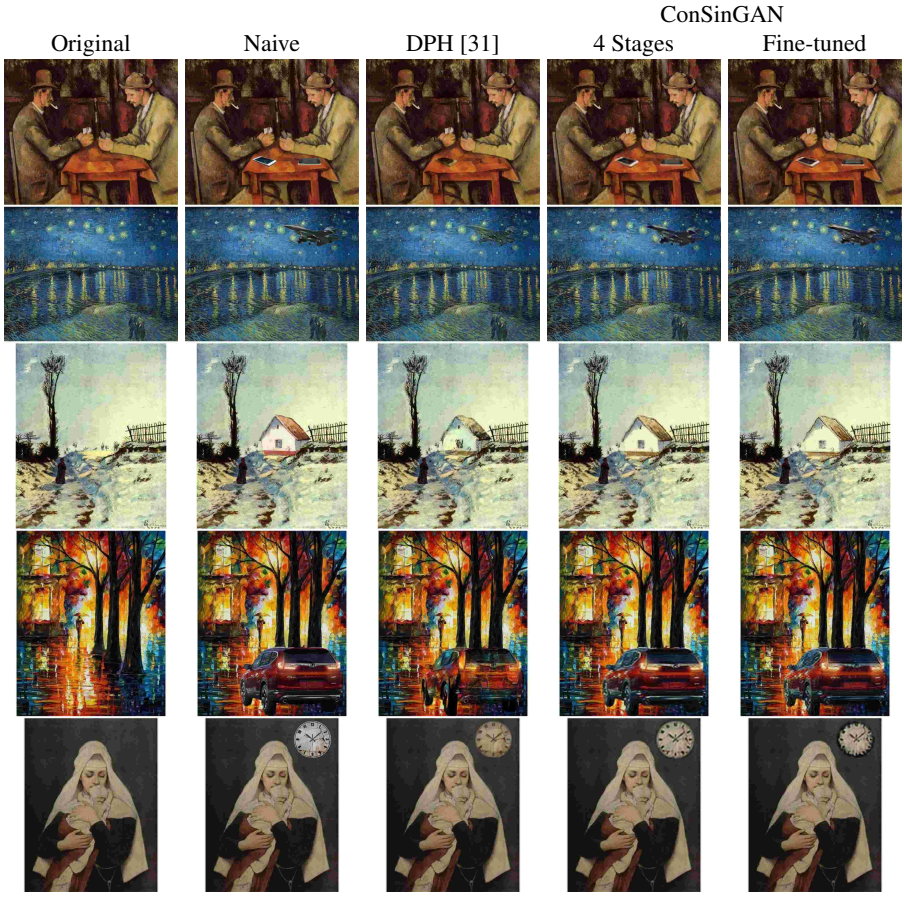

Fig. 20. Comparison of ConSinGAN and Deep Painterly Harmonization (DPH) on high-resolution image harmonization. Images from DPH taken from their official Github implementation. In contrast to DPH, our model does only see the naive image during training if we fine-tune it (last column), but not for our general training procedure (fourth column)

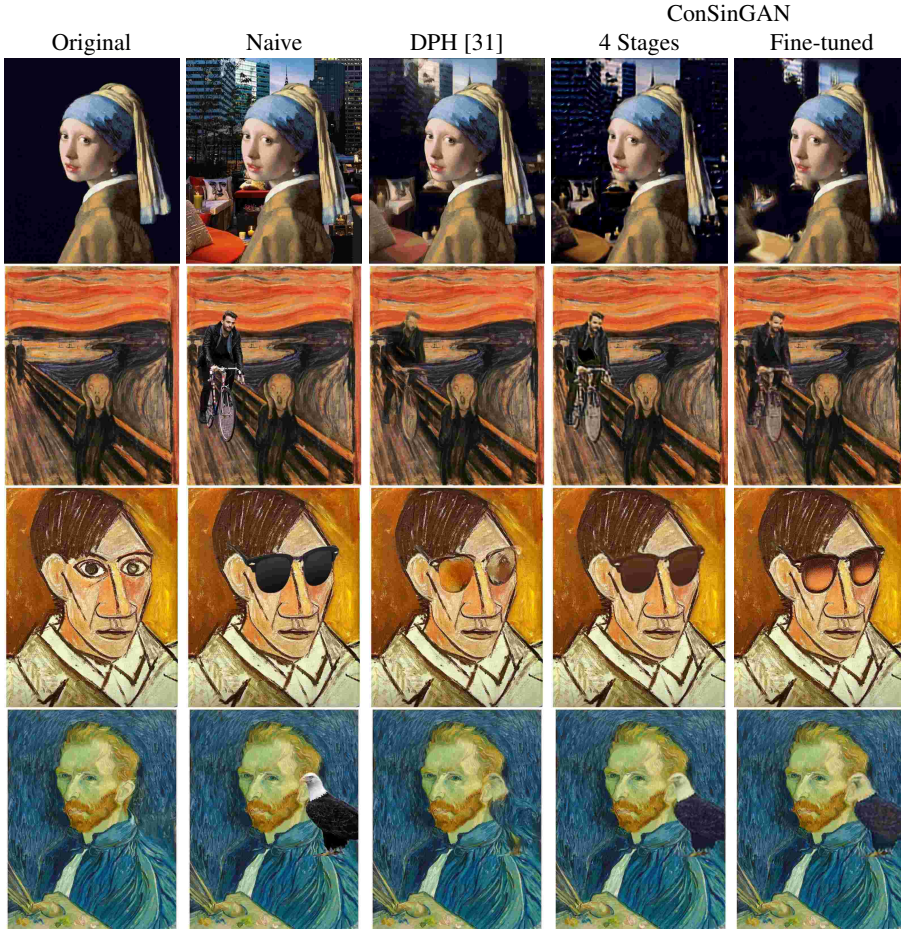

Fig. 21. Comparison of ConSinGAN and Deep Painterly Harmonization (DPH) on high-resolution image harmonization. Images from DPH taken from their official Github implementation. In contrast to DPH, our model does only see the naive image during training if we fine-tune it (last column), but not for our general training procedure (fourth column)

F Images From User Studies

Figure 22 and Figure 23 show the images that were used in the two user studies on the 'Places' and 'LSUN' data sets respectively.

G Image Editing

Figure 24 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.

Fig. 22. Images from the ‘Places’ data set used in our user study

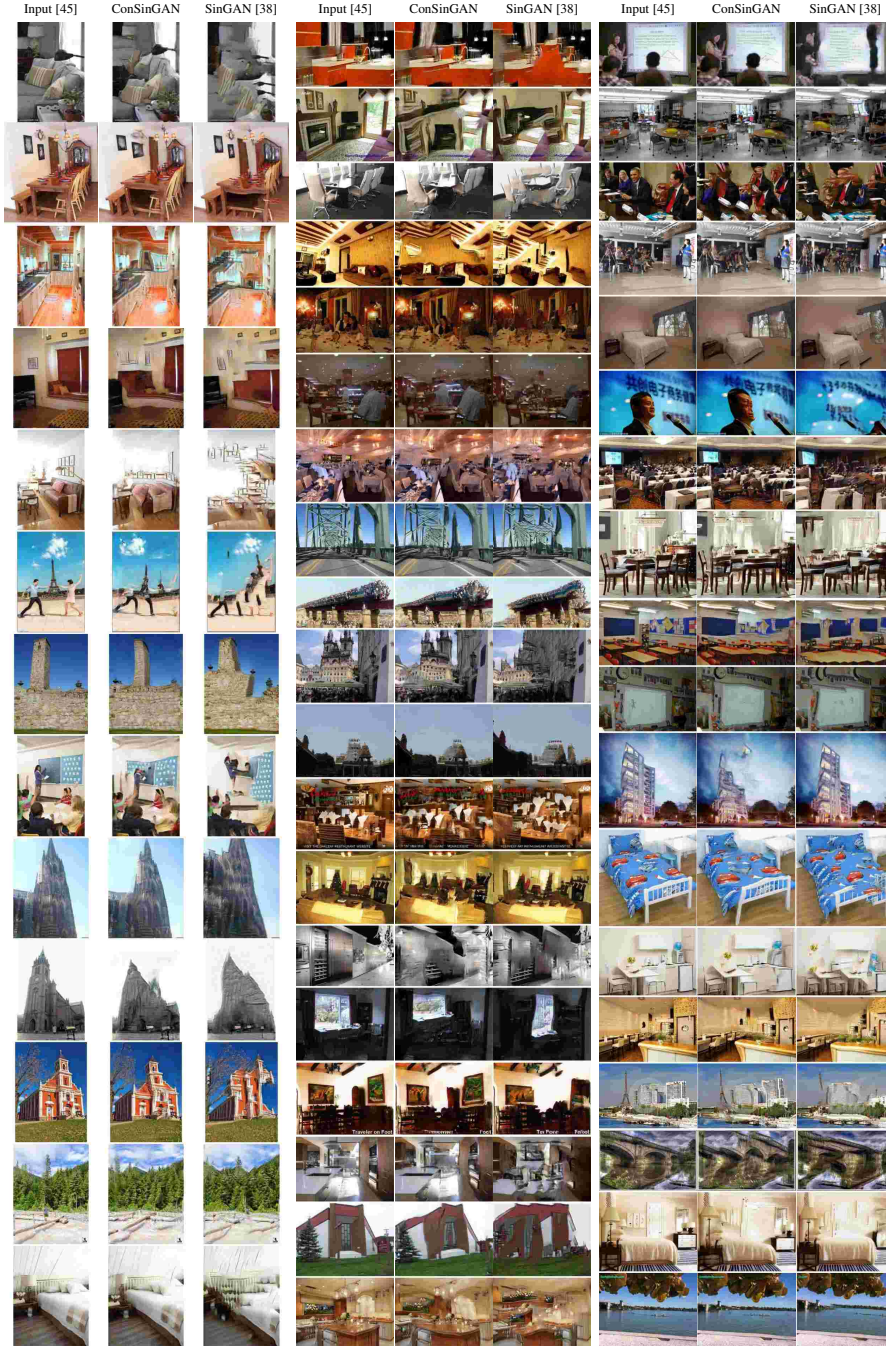

Fig. 23. Images from the 'LSUN' data set used in our user study

Fig. 24. 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

H Other Approaches We Explored

Multiple Discriminators at Each Stage Similar to other work, e.g. InGAN [39], 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 12 and Figure 13). 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 generators 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 [27] 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 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 [2] and pixel normalization [25], 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 [32], such as ELU [8], SELU [29], and PRELU [16]. 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.

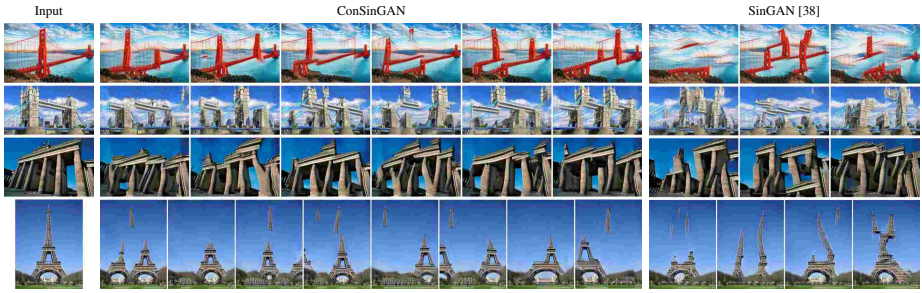

Fig. 25. Results when the models fail to learn the global image layout. The SinGAN models were trained on eight – ten stages, the ConSinGAN models were trained on three – five stages. Note that increasing the number of stages and/or the learning rate scaling δ for the ConSinGAN models improves the learned image layout

I Failure Cases

Figure 25 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.

J Optimization and Implementation Details

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 [14] 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 [28] 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. Our discriminator and each stage of our generator consists of three convolutional layers with 64 filters. 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. 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.