Object Recognition and Computer Vision 2019 - Final Project

Antoine Yang
Ecole Polytechnique - ENS Paris-Saclay
antoineyang3@gmail.com

Abstract

In the last few years, GANs have a made a fantastic breakthrough thanks to their astonishing ability to generate realistic images. However, it is well known that they require a great amount of training data to learn high dimensional distributions of visual data. SinGAN (4) proposes an approach that can effectively deal with complex natural images, training on a single image, using a pyramid of fully convolutional light-weight GANs that capture distribution of patches of the image at different scales. After understanding the paper, I reproduced its key results (quantitative as qualitative) on a various set of images. Then to extend this work, I implemented and experimented scalable approaches on Image Inpaiting in 4 *different settings: small hole, big hole, multiple (big) holes, as well as videos.*

1. Reproduction of results

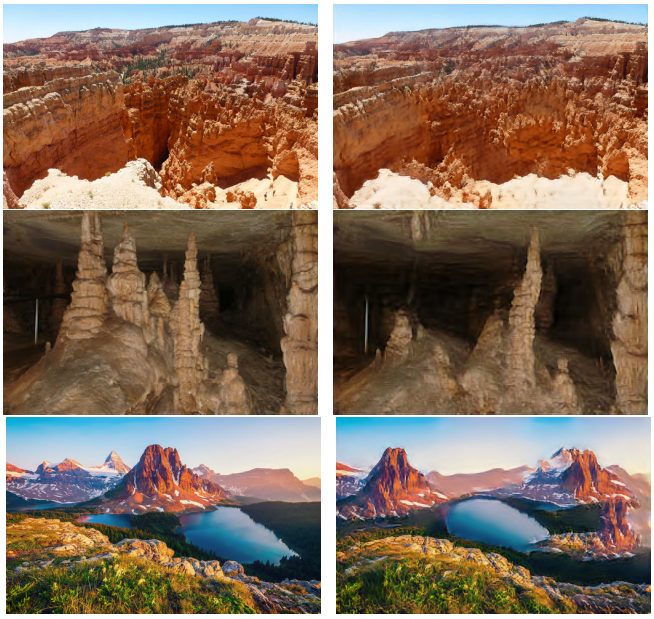
In this part, I present results of SinGAN I reproduced on various applications on Web images and my own images, chosen substantially different from those presented in the paper and its appendix but mostly from nature, most often of size varying from 256 to 512 for height and width.

1.1. Random Samples

After training on a single natural image, I generate random samples from it. Experimentally, I observe that generated samples at start scale higher than 0 are very similar to the original image. At start scale 0, I found pretty realistic results. I tuned the minimal size for training: the lower it is, the higher the number of scales, which enables to capture better large structures and not only textures.

However, fine details are only fuzzily captured at times, and vision effects like water reflection or sunlight are often not captured correctly (see Figure 1).

Furthermore, the method has substantial memory requirements especially when dealing with big images: using google colab GPUs (11 to 16 GB), I could not train on images bigger than about 750×750. To overcome that, authors propose to combine Generation and Super-Resolution.



(a) Original images (b) Generated examples

Figure 1: Random Sample Examples

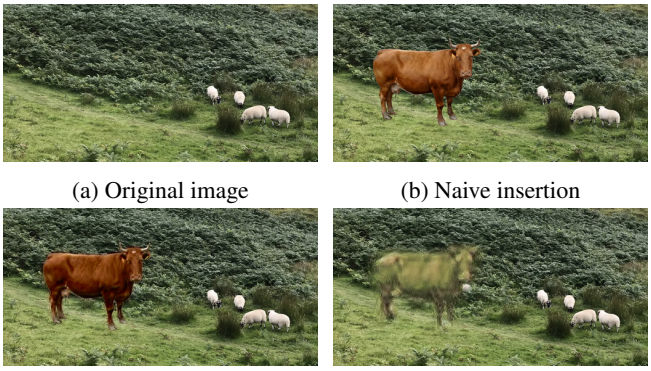
1.2. SIFID and AMT user study

The authors introduce a metric that measures the deviation between the distribution of deep features of generated patches and that of real patches. I experimented on my own set of 12 images, and found significantly lower values as well as values for scale N closer (but still higher) to scale N-1 than in the paper. I also report confusion rates (CR) results of a small survey done with 10 persons, and correlation (I can verify anticorrelation) with SIFID (Corr), in the following table.

1st scale	SIFID	Survey	CR	Corr
N	8.3×10^{-6}	paired	16.67%	−0.14
		unpaired	23.33%	−0.18
N-1	$4.3 \times 10{-}6$	paired	46.67%	−0.25
		unpaired	50.25%	−0.04

1.3. Harmonization

After training SinGAN on a background image, I insert a downsampled version of the image with a naively pasted cow and obtain with start scale 9 a darker cow, with a tone fitting the background, mainly composed of dark green plants (Figure 2). As I lower the start scale, the cow becomes greener, distorted and mixed with the background.



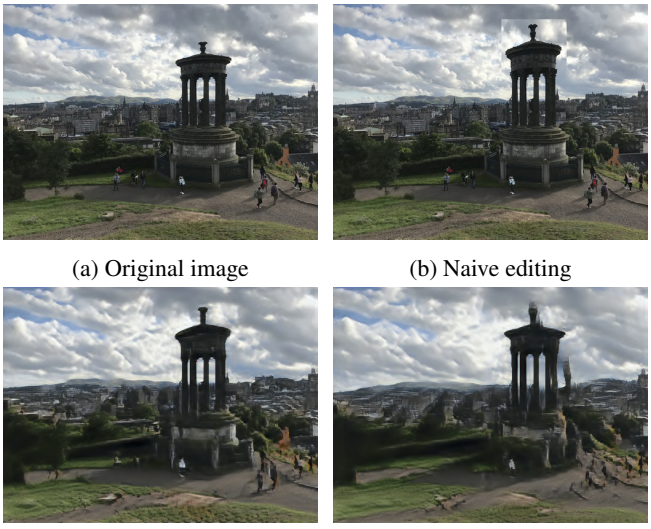
(a) Original image (b) Naive insertion

(c) Output image, start scale 9 (d) Output image, start scale 5

Figure 2: Harmonization example

1.4. Editing

After training SinGAN on a non-edited image, I inject a downsampled version of the image with a tower of naively increased size at start scale 5 and observe that the clouds are nicely harmonized with the background (Figure 3). As I lower the start scale, the image gets distorted and the tower is combined with the clouds.



(a) Original image (b) Naive editing

(c) Output image, start scale 5 (d) Output image, start scale 2

Figure 3: Editing example

1.5. Super-Resolution

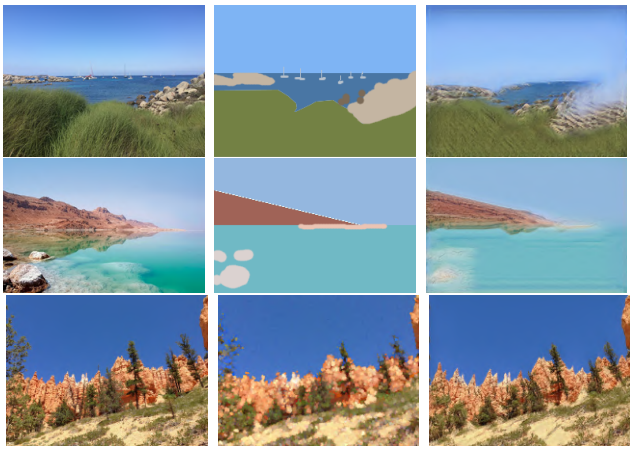
After training the model on a low resolution image, I repetively (4 times) upsample it and inject it to the finest scale generator. The output image has indeed a better resolution than the original one resized with bilinear resampling (Figure 4) but often contains artefacts. I compute PSNR (measure of the reconstruction quality commonly used) using cv2, and report it in Figure 4.

(a) Original image (b) Output image, PSNR 19.307

Figure 4: Super-Resolution example

1.6. Paint-to-Image

After training SinGAN on an image, I inject a painted version with a start scale 2 and recover an image with details close to the original one (Figure 5). The task is clearly harder when the painting is simpler (as in the first 2 cases of Figure 5). As I lower the start scale, the details are finer, but there are more artefacts and blurry parts.



(a) Original Images (b) Painted images (c) Output images

Figure 5: Paint-to-Image examples (start scale 2, 3, 2)

1.7. Animation

After training with noise padding mode on an image, I synthesize a motion with a random walk in z-space with start scale 2. I observe a continuous smooth animation of the leaves of the tree and the blades of grass. As I lower the start scale, the movements get more magnitude (Figure 6).

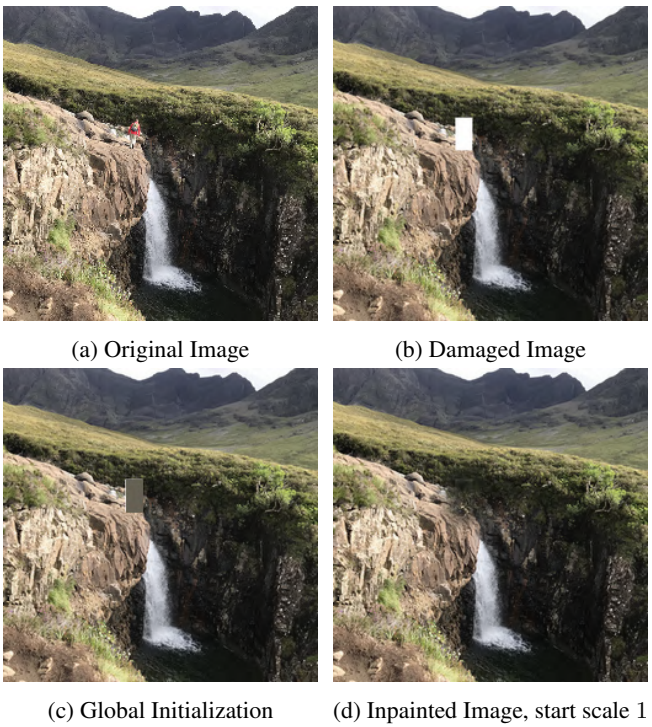
(a) Original Image (b) Start scale 2 (c) Start scale 0

Figure 6: Animation Example ((b),(c): averaged frames)

2. Extension: Scalable image/video inpainting

2.1. Inpainting of a small hole

I imagine that a user crops out an undesired part of the image and wants to recover a realistic piece. During training, I limit all losses to valid pixels to learn on the clean pixels of the damaged image. At test time, I initialize the value on the missing pixels by the mean value of all pixels in the non occluded area (Figure 7). Then I inject a downsampled version of the image at a coerse scale and combine the generated image with the original background.



(a) Original Image (b) Damaged Image

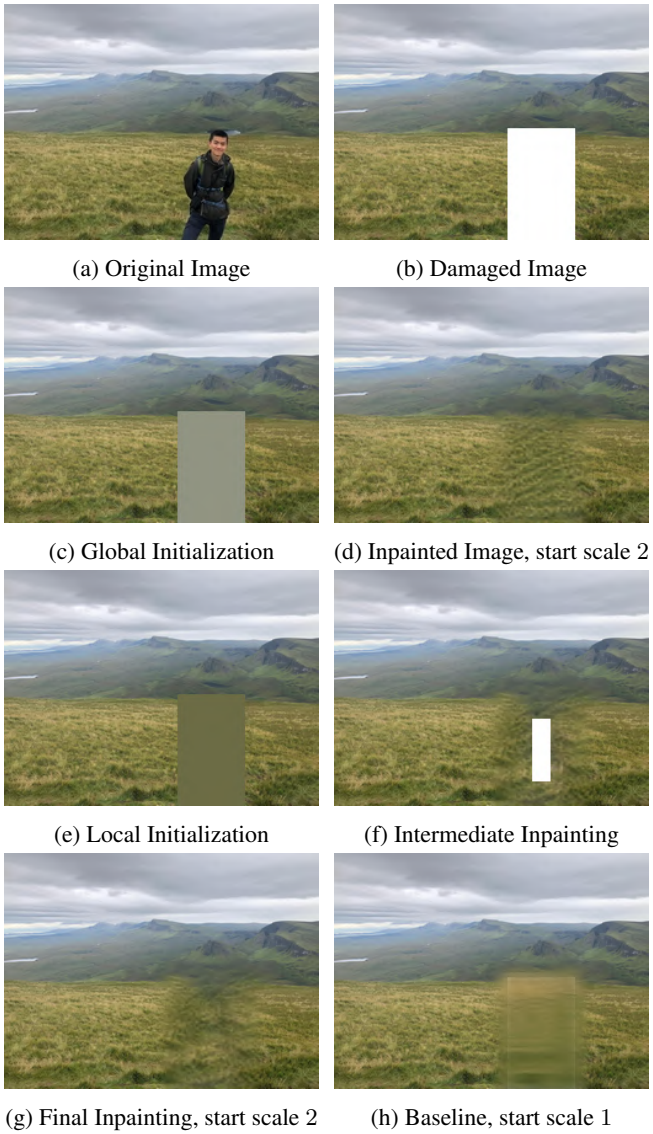
(c) Global Initialization (d) Inpainted Image, start scale 1

Figure 7: Inpainting example with global initialization

2.2. Inpainting of a big hole

As SinGAN only trains on an image, it is challenging to inpaint big holes. To avoid being sensible to pixels far from the hole, I further modify the global initialization, only taking the average color on a neighbourhood of 10 pixels around the hole (local initialization). I also tune the radius of mask dilatation to 10 to smooth the corners. Inspired by (1), at test time, I repetively recover only the 10 pixels at the boarder until recovering the hole, to benefit from better initialization for the most inner parts (progressive inpainting).

I observe decent results recovering texture and smooth transitions, despite a small blur. I compare my results to a normal training+harmonization baseline (min size 15; if not tuned so, the hole is not filled entirely) in Figure 8.



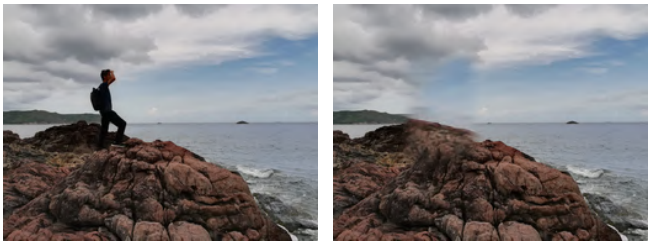
(a) Original Image (b) Damaged Image

(c) Global Initialization (d) Inpainted Image, start scale 2

(e) Local Initialization (f) Intermediate Inpainting

(g) Final Inpainting, start scale 2 (h) Baseline, start scale 1

Figure 8: Progressive Inpainting and local initialization



(a) Original Image (b) Inpainted image

(c) Original Image (d) Inpainted image (e) Progressive inpaint

Figure 9: Additional inpainting examples (start scale 1, 2)

Another strategy for big holes is initializing the hole with the corresponding part of a generated. Additionally, if an inpainting is not satisfatory, I can do a second inpainting, or do an editing (Figure 10), but this is not scalable.

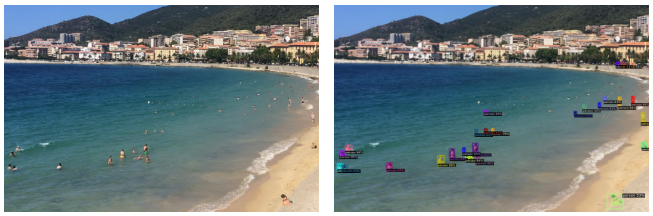
(a) Original Image (b) Inpainting, generative init.

(c) Second Inpainting (d) Inpainting+Editing

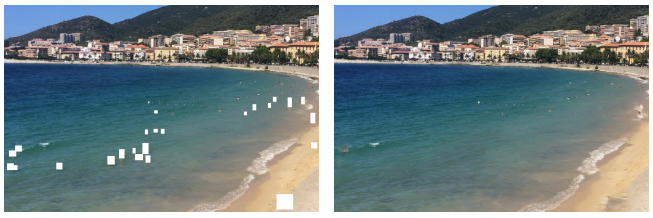
Figure 10: Generative init, editing, multiple inpaintings

2.3. Automated inpainting of multiple (big) holes

I also demonstrate that the previous method can also be applied to multiple holes efficiently. Additionally, I implemented an algorithm automatically cropping out people detected by a Detectron2 (5) Mask-RCNN model (2), and creating the associated binary mask, which enables to have a pipeline automated image inpainting of all instances of a given class. This is also compatible with the previous strategies for big holes. One has to be careful when automatically cropping out, e.g. in (e) Figure 11, I also have to crop out the non-detected cats'shadows otherwise SinGAN is going to learn from it. I also observe some failures at the boundaries e.g. in (h), Figure 11.



(a) Original Image (b) Detection with Mask R-CNN



(c) Automatic Cropping (d) Normal inpainting, start scale 1

(e) Original Image+BBoxes (f) Inpainting, start scale 2

(g) Original Image+BBoxes (h) Inpainting, start scale 1

Figure 11: Multiple Holes and Automatic Detection

2.4. Video Inpainting

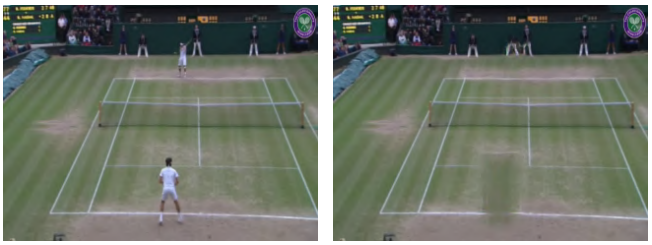
Now, I explore how the previous methods can be generalized to video inpainting for small videos of about 5 seconds (24 frames/s) with no viewpoint change. In that case, I can train on the first frame and use the trained model to inpaint the following frames, as the distributions of their patches are very similar. This results in a very lightly computational method, as inpainting is very fast once training is achieved.

I imagine that an user gives as input the zone (e.g. in Figure 12, below the tennis player on the opposite side of the net) in which he wants to crop out humans. Note that this interactive setting could have also been considered in the previous section. First, I extract frames of the video,

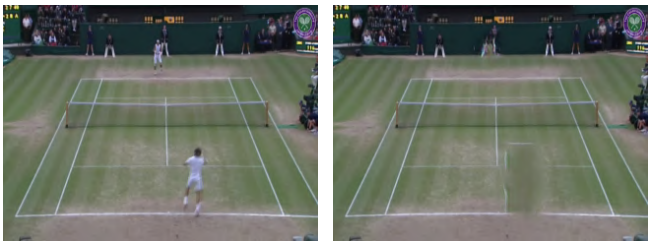
and automatically crop out humans as previously. Second, I train SinGAN on the first frame as in previous inpaintings. After tuning the start scale at the first frame, I inpaint all the frames with the trained SinGAN, and extract a video using cv2. On parallel, I extract audio from the original video using moviepy and reinsert it to the inpainted video.

(a) Frame 0 (b) Inpainted Frame 0

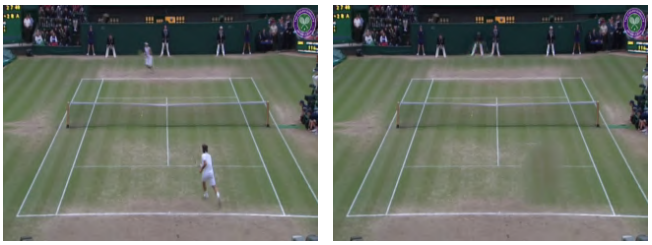
(c) Frame 58 (d) Inpainted Frame 58



(e) Frame 94 (f) Inpainted Frame 94

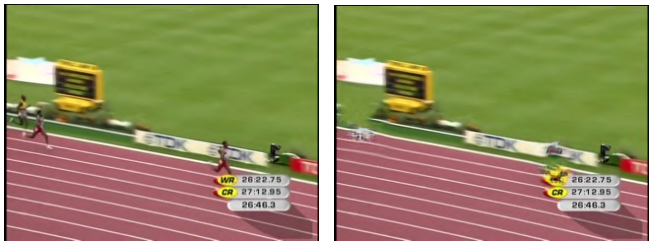


(g) Frame 126 (h) Inpainted Frame 126

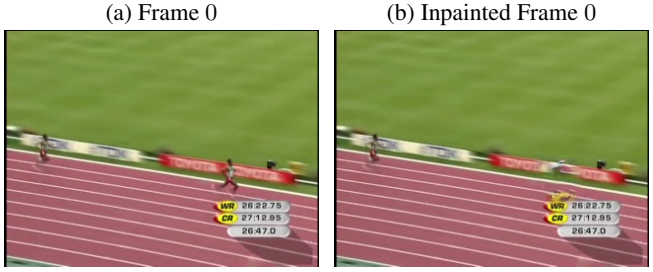


(i) Frame 139 (j) Inpainted Frame 139

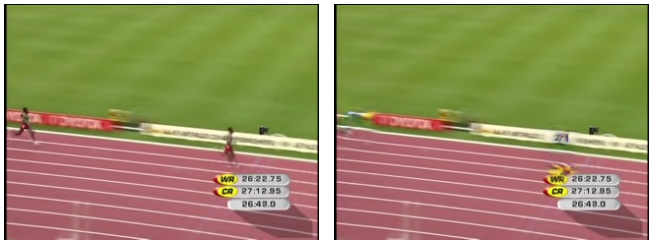
Figure 12: Video Inpainting, start scale 4



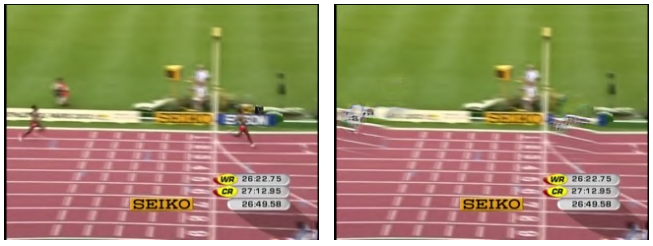
(a) Frame 0 (b) Inpainted Frame 0



(c) Frame 17 (d) Inpainted Frame 17 (The second runner is not detected)



(e) Frame 40 (f) Inpainted Frame 40



(g) Frame 80 (h) Inpainted Frame 80

(i) Frame 104 (j) Inpainted Frame 104

Figure 13: Video Inpainting, camera motion, start scale 3

In Figure 13, I explore the inpainting results with a slow camera motion. The movement causes a blur that makes the Mask R-CNN occasionnaly fails (about 10 frames out of 104), which results in humans reappearing suddenly at times. The movement also causes distortions at later

frames: the lines of the track being leaning on the first frame, given that I only train on this one, the inpainted includes similar leaning lines in all frames.

These results could be improved by taking into account time consistency (proximity of frames close in time) to train and by retraining SinGAN when the camera has moved of a certain distance.

2.5. Evaluation

Finally, I propose an evaluation of inpainting results on the 10 images shown previously, comparing visually with SN-PatchGAN. I use its tensorflow implementation given at `https://github.com/JiahuiYu/generative_inpainting` provided with model weights, based on (7) and (6). Note that it has been trained images from the places dataset (8) of resolution 256×256, 256 being the largest dimension of my test images, and largest hole size 128×128, my holes being of lower size. Figure 14 demonstrates the competitivity of the proposed method, especially for big holes.

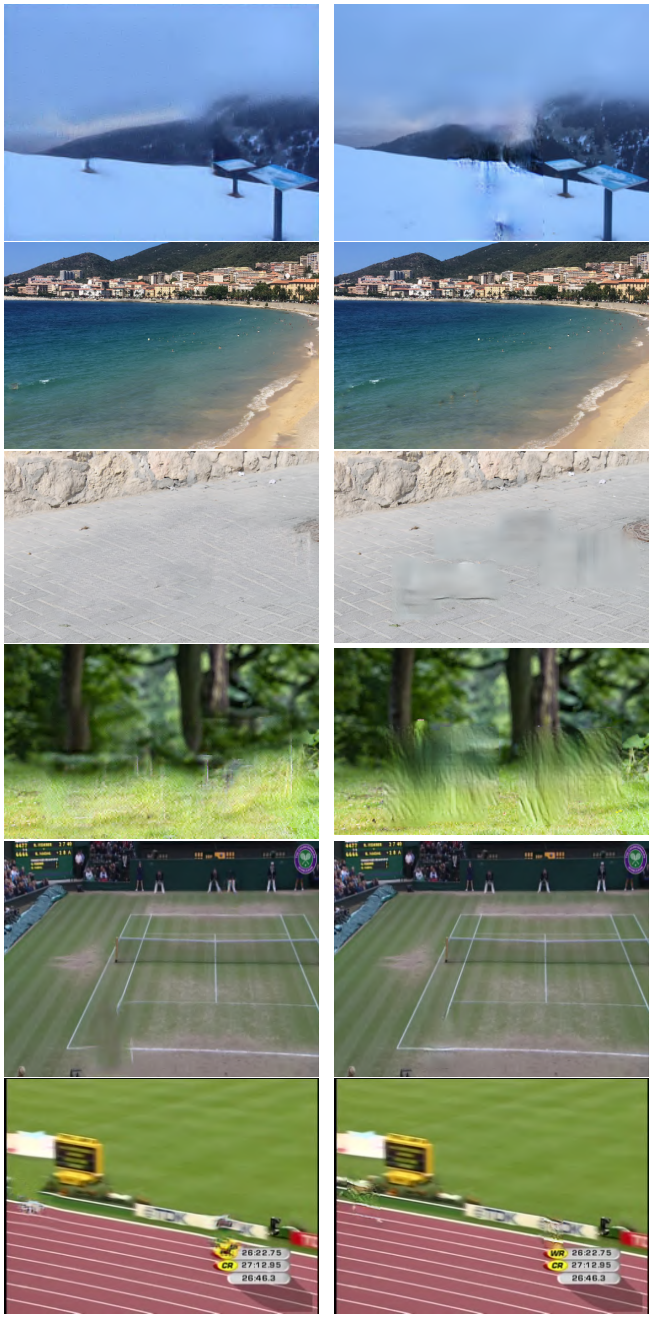


Figure 14: SinGAN (left) vs SN-PatchGAN (right)

3. Conclusion

After understanding the paper, I reproduced its key qualitative and quantitative results (Random Samples Generation, SIFID/AMT, Harmonization, Editing, Super-Resolution, Paint to Image, Animation) on my own images and found interesting results. I then moved on an extension on image inpainting first for small hole modifying the training process and harmonizing a globally initialized hole.

Then I considered image inpainting for a big hole proposing diverse strategies: a progressive inpainting with local initialization, generative initialization, and improving the result by further editing and or do a second inpainting. This work could be improved by further modifying the training, adding for instance a perceptual loss term to regularize the hole pixels, or leveraging partial convolutions to define a convolution inside the hole (3).

Then I proposed a pipeline for automated detection and inpainting of all instances of a given class (possibly in a given zone). Additionally, I worked on short videos inpainting, training SinGAN only on one frame, evaluating it with and without camera motion. I evaluated my results by comparing visually with SN-PatchGAN results. For a quantitative evaluation, I could think of extending the SIFID metric to measure how similar are the distributions of patches in the original image on its valid pixels and in the inpainted image (in the whole image), or do a poll.

While SinGAN exhibits impressive results to create realistic samples as well as for various image manipulation tasks with the same architecture, a recurrent problem I have faced is generating blurry parts and the training instability. Furthermore, in general, SinGAN does not work very well when the global structure fills a major part of the image, and is therefore less generalizable to line tasks.

Finally, I would like to thank Thomas Eboli for advising me during this project.

References

[1] Alexei A. Efros and Thomas K. Leung. Texture synthesis by non-parametric sampling. In *Proceedings of the seventh IEEE international conference on computer vision*, 1999.

[2] Kaiming He, Georgia Gkioxari, and Piotr Dollar. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, 2017.

[3] Guilin Liu, Fitsum A. Reda, and Kevin J. Shih. Image inpainting for irregular holes using partial convolutions. In *The European Conference on Computer Vision (ECCV)*, 2018.

[4] Shaham Tamar Rott, Tali Dekel, and Tomer Michaeli. Singan: Learning a generative model from a single natural image. In *ICCV*, 2019.

[5] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. `https://github.com/facebookresearch/detectron2`, 2019.

[6] Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas S Huang. Free-form image inpainting with gated convolution. *arXiv preprint arXiv:1806.03589*, 2018.

[7] Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas S Huang. Generative image inpainting with contextual attention. *arXiv preprint arXiv:1801.07892*, 2018.

[8] Bolei Zhou, Agata Lapedriza, and Aditya Khosla. Places: A 10 million image database for scene recognition. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.