INF 573 Image Analysis Project: Object Detection and Removal by InPainting

Erwan Umlil
Ecole Polytechnique, Paris
erwan.umlil@polytechnique.edu

Rebecca Jaubert
Ecole Polytechnique, Paris
rebecca.jaubert@polytechnique.edu

1 INTRODUCTION

A problem that we often encounter as an amateur photograph is the unwanted presence of some objects or people on pictures. To deal with this issue, there exist techniques which work very well to remove and then generate the parts of images. However, they are usually based on Generative Adversarial Networks (GAN) and require lots of data. In general, if someone takes a picture of a scenery that is not very famous he or she will struggle to find a dataset corresponding to his/her needs.

We were curious to see if there existed a method that could fill in holes in a realistic way without requiring data we do not have. That is exactly what **SinGAN** [5] model does. It trains on a single image but it is still able to do many image processing tasks very well. It can produce a new image based on a color map, render an rough edited image into a realistic one, harmonize an image after adding elements of different texture, improve resolution or even animate an image. We explored different ways to achieve the best inpainting result. Our goal was to find how we could achieve the best inpainting using SinGAN, and which task with which scale produces the best realistic result.

First, we will explain the object detection procedure we followed so that the user can choose what to remove. Then we will present the different attempts we have made with SinGAN to generate the missing parts of the image. Finally, we will briefly discuss on another attempt we made with another network: EdgeConnect.

2 OBJECT DETECTION AND ERASING

To tackle our problem of an unwanted element on an image, the first step consists of detecting this object and then removing it.

2.1 Detection through segmentation

To perform this task, we use a segmentation method based on the pre-trained net ResNet101 [3] which can be downloaded directly from torchvision [2]. We have to preprocess the input image so that it fits resnet's input shape: we crop and resize the input image and we keep track of the transformation to apply the inverse one later. Then we perform a forward pass on it and as a result obtained the labels (the class category) associated to every pixel. Since we can now group the pixels by category, it is easy to proceed to the removal of any object (see section 2.3 for the choice of the object).

2.2 Erasing

Our "erasing" is actually a simple recolouring of the object's pixels in white. We iterate over all the pixels in the image and check if they belong to the object and if they do, we set their RGB values to $(255, 255, 255)$. To finish this part, we post-process our output to give it back its original shape. The result is presented Figure 1.

(a) Initial image **(b) Post segmentation**

Figure 1: Segmentation step of our pipeline

We notice that segmentation using ResNet is not perfect: a few features of the objects are remaining, such as wing mirrors and car's shadow. A way to improve these results would be to dilate the mask and to set the colour of hole's neighbours to white too.

2.3 Choice of the object

We initially had implemented this algorithm such that every detected object would be erased, but in a real world setting it is often only one object that is desired to be removed. Therefore, we decided to offer the user the possibility to choose which object he wants to crop. To make this functionality available, we pass the list of objects' code numbers (each label is represented by a number) to remove as an argument of the segmentation function. We chose to restrict our application to twenty categories (input as 1,..,20). A table showing the encoding between categories and numbers is provided in the README of our github and in Appendix (Table 1). Default is set to 15, which corresponds to a human.

3 PARTIAL IMAGE GENERATION USING SINGAN

After the object removal process, we have access to three different images: the original, the edited one with white pixels where the object has been removed and its associated mask. This section describes our attempts to render the best generated output.

3.1 SinGAN: brief explanation

Its name "**Sin**gle Image **GAN**" is quite explicit on what SinGAN[5] does. Given a single training image, it is able to learn how to do image manipulation tasks. Its five main tasks are transforming a paint into a realistic photo, rearranging and editing objects in the image, harmonizing a new object into an image, image super-resolution and creating an animation from a single input. However, it is powerful enough that one can use it for other tasks, in our case : InPainting.
It creates diverse samples at various scales such that they carry the same visual content but by mapping random noise they contain new structures. This process generates lots of images resembling the original and thus not requiring any class labels, yet the model now has a dataset it can train on. The higher the number of scales we train on, the larger the structures it can capture. The first scales correspond to the global architecture such as texture whereas the last ones comprehend fine details. Therefore depending on the dimensions of the object removed, the scale chosen to inject when manipulating our input image should differ.

3.2 Training

Since SinGAN learns from a single image, we have to repeat the training process for every picture we wish to process. There are pros and cons: we do not need to download a pre-trained model which could be heavy in memory and not adapted to our data, but we have to launch a training phase for each new image. Thus, the choice of the image we train on among the three available possibilities is important as in our case the original image might contain patterns we do not want, since we remove part of its structure intentionally. So we decided to train SinGAN on the initial image, containing the object we want to remove.

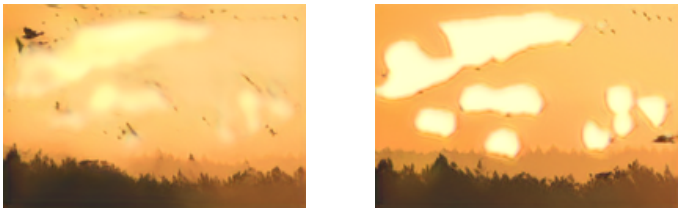
Training can be quite time consuming depending to the hardware used. On our school devices using GPU, training needs about ten to thirty minutes. Fortunately, the training phase is done scale by scale, up to eight scales, and in most cases it is unnecessary to train the network on all scales.

At first, the training algorithm consider global features and progressively, it delves into a lot of details, examining small details. Experiments we did in section 3.3.2 showed that it is unnecessary to train the network on all scales, which can save a lot of time since the last scales are the one which spend the most of time.

3.3 InPainting tests and results

There are three main aspects that can be modified during our generating process: the image we pass as input to SinGAN, the scale used as discussed in the previous section and the mode chosen. In this section, we present our results in the same order as we attempted them. It may not seem logical at first glance, however since those three parameters are heavily correlated it is the easiest way to explain our thought process for each attempt.

3.3.1 ***Edit Mode: raw input.*** Our first guess to generate the missing regions was to use the Edit mode of SinGAN since it is presented in the paper as the most similar to Photoshop. Thus, we trained the model on birds.png (Figure 2a) and we passed birds_seg.png (Figure 2b) obtained after removing the birds through segmentation to SinGAN using Edit mode. We tried several scales with those same parameters to compare. We obtained an output that despite leaving the the white zones noticeable is quite promising as it is already achieving some sort of inpainting (Figure



(a) scale=1 (b) scale=5

Figure 3: Edit in SinGAN on raw segmented input

3). In this case, larger scales appear to work better but higher ones manage to remove the remaining pixels from the object that had survived the segmentation process. That is due to a better understanding of the global structure of the image, contrary to smaller scales which concentrate on details and pixels which are neighbours. Moreover, in this case our picture does not require much detail, instead it is almost a uniform colouring.

(a) Input (b) After segmentation

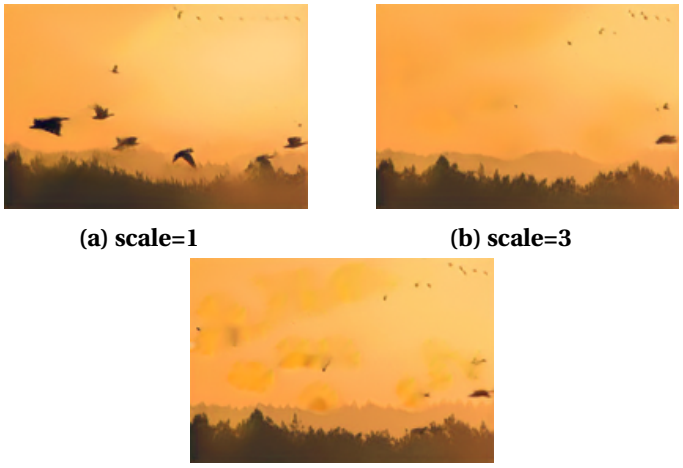
Figure 2: Effect of segmentation on birds.png

Our main conclusion from this test was that we had to add an extra pre-processing step to the segmented image. Indeed, SinGAN can make a modified region appear as natural only if we provide an adapted input image. Running SinGAN on the raw image with white pixels was not concluant (Figure 3). So we must try to approximate the render of the erased region before passing the image through SinGAN.

3.3.2 ***Pre-processing of the image: inpainting colouring.*** Actually, we wanted to verify if our assumption that editing on non white holes would work better than editing on the raw segmented image. We explored three alternatives to do this.

cv2. At first, we used *cv2.inpaint* to have fast results and to be able to compare our further results to a standard function. We obtained the images presented in Figure 5.

Figure 4: Output of cv2 inpainting on segmented birds image



(a) scale=1 (b) scale=3

(c) scale=6

Figure 5: Edit Mode with cv2 inpainted image as input

We can notice that the best scale seems to be scale 3. Indeed, scale 1 understands too much the global structure of the image, including the presence of birds, since we trained SinGAN on the input image with those birds. So when we ask it to edit the naively inpainted image, it draws the birds we wanted to remove. Scale 6 is too concentrated on details and we have the same issue we had with the raw segmented image (Figure 3). Scale 3 gives very satisfying results: all the birds that were captured by segmentation have been removed and the final image is quite realistic.

Naive diffusion. The second algorithm we implemented was a naive one. We diffuse the color of the pixels which are adjacent to a white hole by setting, at each time step, the colour of an originally white pixel to the mean of the non-white neighbours. We repeated this a given number of epochs. We obtained quite disappointing results (Figure 6) compared to *cv2* algorithm, but this naive inpainting allows better results than no inpainting at all.

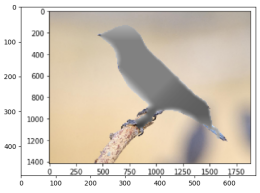


Figure 6: Result of naive diffusion on bird.png

Heat diffusion. Finally, we implemented an inpainting based on a physical phenomenon. Our thought was to notice that the best way to inpaint a hole in an image was to have a smooth transition with the edges of the hole. A good idea to achieve this is to diffuse the colour of non white pixels by following a physical principle. We used the heat diffusion equation:

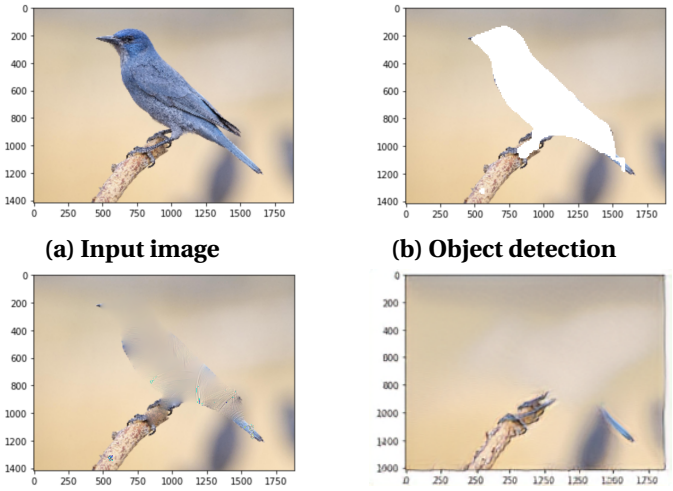
$$\frac{\partial T}{\partial t} = \kappa \Delta T + S$$

where T is temperature (here the colour of pixels), κ (set to 5e-1) is the diffusion coefficient and S the production of the source (here $S = 0$). As this equation comes from physics, we hope that it will provide a result that seems natural to the eye.

So we implemented this idea. For a given number of time steps (epochs), we perform one step of the diffusion equation:

$$T_{i,j}^{n+1} = T_{i,j}^{n} + \Delta t \kappa \left[\frac{T_{i-1,j}^{n} - 2T_{i,j}^{n} + T_{i+1,j}^{n}}{\Delta x^2} + \frac{T_{i,j-1}^{n} - 2T_{i,j}^{n} + T_{i,j+1}^{n}}{\Delta y^2} \right]$$

We apply this formula to the neighbouring of each originally white pixel. We know which pixel was originally white thanks to the mask we produced during the segmentation step. Step by step, white pixels become non white.



(a) Input image **(b) Object detection**

(c) Heat diffusion **(d) SinGAN scale=2**

Figure 7: Results on bird.png

With this method, we obtained very satisfying results, such as those presented in Figure 11.

3.3.3 ***Paint2image Mode*** . Until now, we used the Edit mode of SinGAN. However, a second mode which is implemented in SinGAN could lead us to a good final result: the Paint2image mode. Indeed, this algorithm allow any user to paint something and it will make the painting look like the original image. This is quite close to our task: we *in*painted something and we want to smooth the result with the remaining of the input picture.

So we follow the same pipeline than with Edit mode, and we use Paint2image instead of Edit when we arrive to the SinGAN. A final image we obtained is presented Figure 8. There are more residual pieces of bird because the parts that the segmentation did not detect are spread by diffusion, then the Paint2image mode interpret them as a drawing and give them the appearance of the original image. However, the background seems more natural than the one we got using Edit mode. So with a perfect object detection, this mode might outperform Edit mode.

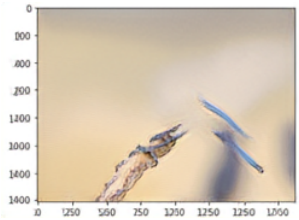


Figure 8: Result of paint2image on bird.png

4 OTHER ATTEMPT

InPainting using EdgeConnect[4]

EdgeConnect comprises an edge generator followed by an image completion network. The edge generator hallucinates edges of the missing region (both regular and irregular) of the image, and the image completion network fills in the missing regions using hallucinated edges as *a priori*. [1] We attempted to use this pre-trained model to fill our erased region in the image. The goal of this attempt was more to explore other available solutions, for example solutions that avoid a training for each picture. As we downloaded a pre-trained version of EdgeConnect, no training phase was required.

However, this method is designed for images with fine details because the picture to recover needs to have edges so that the network could hallucinate missing ones. So it doesn't perform optimally on a large hole like the ones we create by removing a full object. Moreover, most of images we used contained a landscape (for instance, a sky), so there is no edge to hallucinate. It was easy to use as it requires no extra training so we still did a few tests but as expected we obtained poor results, as shown in Figure 9.

Figure 9: InPainting with EdgeConnect

5 CONCLUSION

In this project, we developed a concrete pipeline to detect and remove an unwanted object from a picture. We used SinGAN network and designed a way to use it that is different from its official use. We also designed an adapted pre-processing by experimentation, and we implemented it to make this whole pipeline easy to test for any user.

The results we obtained are quite satisfying compared to what we expected, but they are still not perfect. More precisely, improvements can be done at the segmentation level, because the current implementation let parts of the object in the image.

A further work could be to improve this detection step. One might also try to optimize the heat equation inpainting, which involves several unexplored parameters. One could also try to use other diffusion equations. Finally, one might try to train SinGAN on the output of the heat diffusion inpainting step rather than on the input image.

Thanks to this work, we learned how to complete a full computer vision project, that means to make an existing code work and to implement a pipeline involving this code.

6 APPENDIX

Latest working version of the project with all required data is available on this **repository**.

Below you can find the results of two other experiments we made.

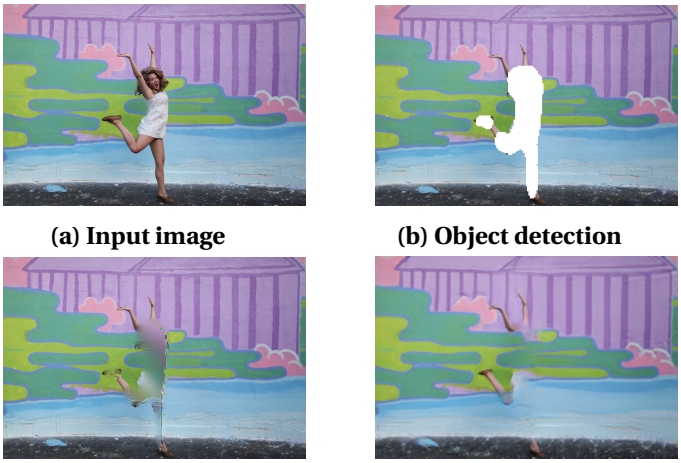
(a) Input image

(b) Object detection

(c) Heat diffusion

(d) SinGAN scale=2

Figure 10: Results on car.png



(a) Input image **(b) Object detection**

(c) Heat diffusion **(d) SinGAN scale=4**

Figure 11: Results on happy_woman.png

Code	Label
1	Aeroplane
2	Bicycle
3	Bird
4	Boat
5	Bottle
6	Bus
7	Car
8	Cat
9	Chair
10	Cow
11	Dining table
12	Dog
13	Horse
14	Motorbike
15	Person
16	Potted plant
17	Sheep
18	Sofa
19	Train
20	Tv/monitor

Table 1: Class labels encoding for segmentation

REFERENCES

[1] [n. d.]. EdgeConnect Github. ([n. d.]). https://github.com/knazeri/edge-connect

[2] [n. d.]. Torchvision. ([n. d.]). https://pytorch.org/vision/stable/index.html

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep Residual Learning for Image Recognition. (2015). arXiv:cs.CV/1512.03385

[4] Kamyar Nazeri, Eric Ng, Tony Joseph, Faisal Z. Qureshi, and Mehran Ebrahimi. 2019. EdgeConnect: Generative Image Inpainting with Adversarial Edge Learning. (2019). arXiv:cs.CV/1901.00212

[5] Tamar Rott Shaham, Tali Dekel, and Tomer Michaeli. 2019. SinGAN: Learning a Generative Model from a Single Natural Image. In *Computer Vision (ICCV), IEEE International Conference on*.