Image Inpainting and Object Removal with Deep Convolutional GAN

Qiwen Fu
Stanford University
qiwenfu@stanford.edu

You Guan
Stanford University
you17@stanford.edu

Yuxin Yang
Stanford University
yuxiny@stanford.edu

Abstract

In this project, we implemented Deep Convolutional Generative and Adversarial Neural Network to solve landscape picture inpaint problem. The resulting model beats most of the existing published solutions so far.

1. Introduction

1.1. Motivation

Image inpainting has always been a challenging and ongoing field of exploration for consumers. With the rising demand of taking nice pictures, lots of effort has been invested in building better tools for users to take athsetically beautiful pictures. But sometimes, users could take undesirable pictures when their object of interest is obstructed by the unrelated. For example

Motivated by these could-be-improved pictures, in our project, we are going to attempt to tackle the problem of getting rid of these undesirable objects in these pictures by using generative and adversarial nets.

1.2. Objective

The objective of this project is to implement the recently published Deep Conditional Generative and Adversarial Network (DCGAN) on image inpainting. GANs incorporates two neural networks - a generator which takes in the incomplete image as the input and generates a fake full image with the missing segment filled, and a discriminator which tries to distinguish generated images from original full images. The generator and the discriminator are trained at the same time and trying to beat each other.

Base on Yeh's work, we intend to overcome their shortcoming of using some too specific datasets with a relatively simple latent spaces. With all of the existing work considered, we intend to design a new model to fulfill such tasks like inpainting for some more general purposes. One application would be to develop mobile Apps to erase tourists from the landscape we care about. Some other applications might include restoring old or corrupted photographs, or lowering the cost of creating huge sceneries in movies and so on.

2. Related Work

As image inpainting being an ongoing challenge for quite a long time, several traditional methods that do not involve the usage and training of CNN have been developed. In 2003, Bertalmio, Bertozzi and Sapiro published an approach based on the "Navier-Stokes" equations for fluid dynamics, which used ideas from classical fluid dynamics to propagate isophote lines continuously from the exterior into the region to be inpainted [2]. And in 2004, Telea introduced an algorithm based on the fast marching method for level set applications, which was claimed to be simple to implement, fast, and able to produce similar results to more complex and slower know methods [7]. We have experimented with both of these two traditional methods, and the comparison between them and our method will be analyzed in later sections.

The state-of-the-art learning based method requires specific information about the holes in the training phase. And there are some existing CNN methods trained to generate an arbitrary missing segment based on its surroundings in the image with an algorithm driven by context-based pixel prediction[5]. Pathak experimented with two different loss models: a standard pixel-wise reconstruction loss, and a reconstruction plus an adversarial loss, with the second one giving better result than the first one. But their final results were still unsatisfactory as the generated fillings for the missing segment were mostly blurry and distorted. And we think it was because of that, as we mentioned in the motivation, this method extracts information from only one single image. This defect motivates us to implement a method that would generate the missing content by conditioning on the set of available data. Another inspiration from this work is that we could try different loss models if we are not getting satisfactory results.

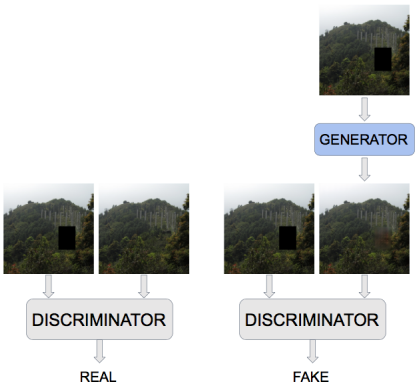


Figure 1. Conditional GAN

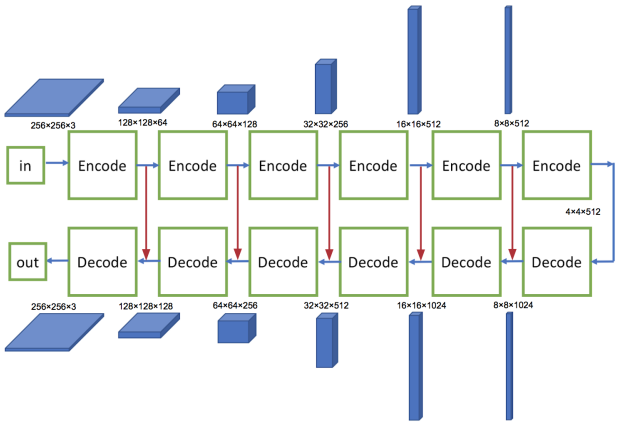


Figure 2. Network Architecture of the Generator

Based on those state of art works, there were some recent break-through. In 2016, Yeh and his group from UIUC claiming that their model outperforms the state-of-the-art methods by innovatively implementing GAN. And their outcome achieves pixel-level photorealism.

Yeh and his team proposes to use deep generative models for image inpainting [8]. They conduct experiments on three datasets, including the CelebFaces Attributes Dataset (CelebA), the Street View House Numbers (SVHN) and the Stanford Cars Dataset. They show generative results by GANs with high image quality, compared with TV inpainting [1], LR inpainting [4] and nearest neighbor inpainting [3], which are three relatively mature methods to take care of this problem.

However, the dataset they used are too specific with a relatively simple latent spaces, making the inpainting task easy. This defect motivates us to research deeper into the use of GANs in inpainting. We are going to experiment on a more diverse dataset, such as MIT_places, and try to train a model that is more general.

3. Methodology

3.1. Conditional GANs

Generative Adversarial Nets (GAN) contains two competing neural network models, including a generator and a discriminator, which are two players in a game and trying to beat each other. Generally speaking, the discriminator tries to tell the fake images generated by the generator from real images, while the generator tries to generate good-looking images to fool the discriminator. Specifically for our model, the generator takes in the input cropped images and generates fake recovered images. The discriminator takes in both generated images from the generator and the ground truth real images, along with the cropped images, and tries to discriminates real images from fake generated images. Fig.1 depicts this process. During the training process, the generator and the discriminator are playing a continuous game. At each iteration, the generator is trained to produce more realistic images, while the discriminator is getting better at distinguishing fake images. Both models are trained together in a minimax fashion and the goal is to train a generator to be indistinguishable from real data.

3.2. Network Architectures

The architecture of the generator network we built shown in Fig.2 is based on the U-Net[6], which incorporates convolution layers in an encoder-decoder fashion to generate recovered images from cropped images.

The generator consists of an encoder, which is a contracting network, and a decoder, which is an expanding network. The input to the encoder is the input tensor of cropped images. The encoder shrinks the size of the tensor layer by layer, with several convolutional layers whose strides are larger than one. However, the depth of the tensor, e.g. the last dimension of the tensor is increased layer by layer with an increasing number of filters used each layer. The output of the decoder is a small tensor $(4 \times 4 \times 512)$ which is the encoded embeddings in the latent space, containing the context information of the original images. The output of the encoder is the input to the decoder, which expands the tensor layer by layer with conv2d_transpose and construct the recover images that are of the original image size. Using this model can lead to much more compact feature learning in the middle of the layers without consuming large memory.

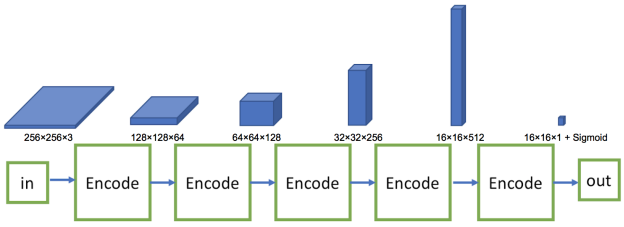


Figure 3. Network Architecture of the Discriminator

The encoder and the decoder is basically symmetrical: there are 6 layers of encoding and 6 layers of decoding. The number of filters in the encoder increases layer by layer, while the number of filters in the decoder decreases number of filters layer by layer. Each encoding layer consists of a 2D-convolution for down-sampling, batch normalization, and leakly relu activations; each decoding layer consists of a transpose convolution for up-sampling, batch normalization and relu activations. To allow the network to skip layers, we concatenate the mirroring layer from encoder at each decoding layer. With skipped layers, the model can learn weights to ignore deeper layers. This can help model to retain components from original input more easier in deep CNN, which is particularly useful in segmentation task.

The discriminator is a simple decoder classifier network, and its architecture is shown in figure3. The input of the discriminator is the concatenation of the cropped image with either the ground truth images or the recovered images generated by the generator. The discriminator is consisted of 5 layers of encoder, which is similar to the encoder of the generator: each encoding layer is consisted of a convolution operation with stride greater than 1, a batch normalization, and a leaky relu activation. The last layer then goes through a sigmoid activation to return a number from 0 to 1 that can be interpreted as the probability of the input being real or fake.

3.3. Loss Functions and Objective

With conditional GAN, both generator and discriminator are conditioning on the input x. Let the generator be parameterized by θ_g and discriminator be parameterized by θ_d.The minimax objective function can be expressed as:

$$\min_{\theta_g}\max_{\theta_d}\Big[\mathbb{E}_{x,y\sim p_{data}}\log D_{\theta_d}(x,y)+ E_{x\sim p_{data}}\log(1-D_{\theta_d}(x,G_{\theta_g}(x))\Big]$$

The objecitve function consists two parts. THe first part represents the the loss coming from the discriminator. If the discriminator did well in distinguishing the generated picture and target, this gives the generative model a high loss, vice versa. while second part represents the L1 loss the difference between the generated picture and the target. Note that we did not introduce noise in our generator because we do not find it working better. Also, we consider L1 difference between input x and label y in generator. At each iteration, the discriminator is trained to maximize θ_d according to the above expression and generator is trained to θ_g in the following way:

$$\min_{\theta_g}\Big[-\log(D_{\theta_d}(x,G_{\theta_g}(x)))+\lambda\big\|G_{\theta_g}(x)-y\big\|_1\Big]$$

With GAN, if the discriminator considers the pair of images generated by the generator to be fake (not well recovered), the loss will be back-propagated through discriminator and through generator. Therefore, the generator can learn how to recover the image to make it look real.

4. Dataset

We trained and tested our model using the MIT Places dataset [9] by MIT Computer Science and Artificial Intelligence Laboratory, as well as some pictures taken by ourselves. The MIT Places dataset contains in total 250 million 256×256 pictures under different categories, with about 10-15K pictures under each category. We used the pictures under categories mountain, snowfield and pasture for training because they have relatively low variance compared to other categories and are more straightforward for our GANs to learn.

For dataset preprocessing, we first manually filtered and removed the unsuitable pictures whose major focus was either humans or some sharp objects such as buildings or vehicles instead of natural landscapes as we desired. About 15000 pictures were left after this step. Then we cropped out a rectangle portion with random size (within a pre-determined range of 30 - 80 pixels in height or length) on a random position from each training image by making that portion black. The DCGAN will get trained by learning how to fill in those cropped out regions.

5. Experiments and Comparisons

5.1. Baseline

For our baseline, we firstly implemented Bertalmio, Bertozzi and Sapiro's work with "Navier-Stokes", and Telea's algorithm from OpenCV library inpaint target pictures. Results are shown in Fig.4,and Fig.5

Original Cropped Cropped Mask

”Navier-Stoke”, PSNR = 31.8328 Telea, PSNR = 32.5291

Figure 4. ”Navier-Stoke” algorithm and Telea’s algorithm performance in landscape inpainting for mountains

Original Cropped Cropped Mask

”Navier-Stoke”, PSNR = 37.1354 Telea, PSNR = 37.2827

Figure 5. ”Navier-Stoke” algorithm and Telea’s algorithm performance in landscape inpainting for trees

PS house, PSNR = 30.7228 PS Tree, PSNR = 34.8032

Figure 6. Photoshoped house and tree using cropped pictures.

We further measured the PSNR for the baseline rendered pictures for houses and trees, and captioned them under the rendered results in Fig.4, and Fig.5. We believe that the inpainting PSNR measurement for both algorithms are rather similar. Telea’s algorithm beats ”Navier Stoke” by a small number.

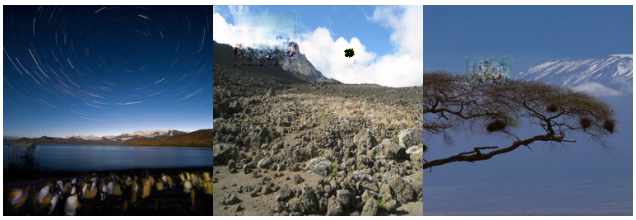
Apart from the two algorithms above, we also implemented another baseline by manually inpaint using Adobe Photoshop. And we measured their PSNR values and captioned them in Fig.6.

Interestingly, we can see that although the Photoshoped results are perceived much better than the output of ”Navier-Stoke” algorithm and Telea’s algorithm, the PSNR ratio is still lower for both of those pictures. We can conclude from this that PSNR is not a sufficient parameter for image inpainting.

5.2. Training Environment and Process

We built our DC-GAN model, under Tensorflow CUDA mode and Nvdia GeForce GPU. It took around 36 minutes to finish training each epoch. Since our generative model performed too well after on average 20 epochs and generative model tends to overfit Fig.7, we had to reinitiate all the parameters to restart the training process of it after GAN loss becomes stably high for our generative model. For an entire training cycle it takes around 12×3 hours to finish 3 sets of 12-hour 20-epoch training. The training loss for generative network stabilizes at around 3.5.

We fine-tuned our model by implementing different learning rate, loss functions (L1 and L2 loss, a combination of both, and arbitrary loss functions), batch size, number of hidden states, weights of GAN loss and L1 loss function. We particularly tried to come up with loss functions innovatively, because we found our generated pictures to be extremely patchy, especially using L2 loss Fig.8. This is



Example 1 Example 2 Example 3

Figure 7. Over-fitting examples

what we expected, since L2 loss punishes large values, but not sparsity. This leads to our image being too patchy or blurry, compared to L1 loss which encourages sparsity. We implemented a loss function on the sum of gradient of this patch, so we can further punish the patchiness. But still did not see significant improvements. We eventually decided to use the L1 loss which is also introduced in section "Loss and Objective function" above. The figure below compares L2 loss versus L1 loss.

L2 loss fine tuned L1 loss fine tuned

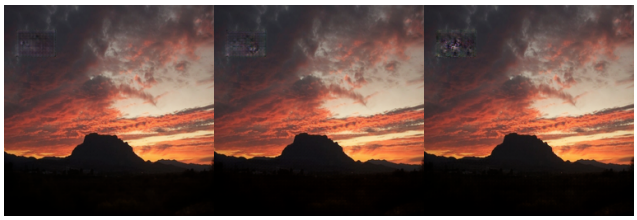
Figure 8. L2 loss versus L1 loss.

During our training process, we saved our model every epoch. We originally assumed that the quality of generated pictures will increase with number of epochs we train. But, we soon noticed that it starts to overfit aggressively after 20 epochs. The process is further illustrated using the following graph of transition Fig. 9.

During the training process, we also observed the outputs of training examples. Some of them are listed below 10.

6. Result and Analysis

For analysis purposes, we first chose two random pictures in test dataset with randomly cropped rectangular shaped patches to see the results (Fig. 11, 12).
PSNR ratios are computed and captioned. We can see that although our GAN results are visually more sharp and natural, Telea's naive inpainting method still holds



After 10 Epochs After 15 Epochs After 20 Epochs

Figure 9. Transition from under-fitting to over-fitting.

GAN training output1 GAN training output2 GAN training output3

GAN training output4 GAN training output5 GAN training output6

Figure 10. Transition from under-fitting to over-fitting.

Original Cropped Telea, PSNR 37.2827 =DC-GAN, PSNR = 35.4635

Figure 11. Sample Test Result: Trees

Original Cropped Telea, PSNR 32.5291 =DC-GAN, PSNR = 32.2640

Figure 12. Sample Test Result: Houses

are larger PSNR result. We also observe that GAN PSNR results beat manually photoshopped results for most cases.

After evaluating the test dataset, we manually cropped out a person out of a picture from dataset. We selected this picture with a person in it from the raw dataset without manual filtering, and cropped out a piece from this picture using Mac Preview.

Original Cropped Telea Recovered

Figure 13. Mountain Sample Test Result with manual cropping

Original Cropped Telea DC-GAN

Figure 14. Snow Sample Test Result with manual cropping

We observe a much better result with our generative model than Telea's naive inpainting.

Knowing that our GAN model runs well with MIT Places dataset, we began to wonder whether it still works well with our own pictures. We manually selected several pictures from iPhone with unwanted objects in it. And manually cropped out the unwanted objects using Mac Preview. We ran them through both Telea's naive algorithm and GAN, and obtained the following results in Fig. 15, 16, 17.

Original Cropped Telea DC-GAN

Figure 15. Picture Yuxin took in Zion mountain.

Original Cropped Telea DC-GAN

Figure 16. Picture Yuxin took in Tahoe on Gondola Express.

Original Cropped Telea DC-GAN

Figure 17. Picture Yuxin took during her road trip in Death Valley.

7. Conclusion

In conclusion, we obtained considerably visually pretty results using DC-GAN. Although the PSNR value is still on average lower than "Navier-Stoke" algorithm and Telea's inpainting algorithm, we received overwhelmingly more positive results by survey in friends. Since our GAN are specially trained for landscape, our model learned significant amount information during training process and seem to outperform all the existing solutions on market, except manually photoshoped ones. We see huge potential for our model to be implemented in mobile devices or industry due to its small storage consumption and speed in computing. We look forward to seeing our model scale out in industry in the future after more fine tuning and optimizing.

8. Challenges and Future Work

8.1. Challenges and Solutions

The challenges we encountered in this project mostly occurred either during the DCGAN training process or during the preprocessing of the dataset.

One major challenge that made DCGAN difficult to train was that discriminator and generator learn with different speed. In order to get satisfactory training results, both discriminator and generator has to be tuned such that they are learning at the same pace. We had to try and test with many different learning rates and manually retrain discriminator from scratch several times throughout the training, or it simply becomes too good. Another major challenge was the high cost in both time and memory for the training of DCGAN. The GAN model takes around 50 minutes to train 1 epoch, and it usually takes about

40 epochs in order to get acceptable results. At the same time, memory overhead is also very high. CUDA crashes from time to time, leading to wasted training efforts.

Preprocessing the dataset was also a major challenge for us. After yielding unsatisfactory training results for several rounds we noticed that part of the pictures in the dataset were not suitable for our training purpose at all. The major focus of those pictures were either humans or some sharp objects such as buildings and vehicles instead of our targeted natural landscape. To solve this problem we had to each go through the 10-15K pictures in one category manually to get rid of those unsuitable pictures, which cost large amount of time and effort.

8.2. Future Work

Several future steps could be taken to improve the results as well as put the trained model into practical usage. Since our current output images have some artificial square-shaped pattern on the filled portion, we could fine-tune the model to make the output image less artificial and more realistic, or also integrate our model with some traditional image inpainting methods to get better results. In addition, since we only trained the model on specific landscape datasets with relatively low variance, in the next step we could train on a compound dataset to get a more general model. Eventually we could also develop mobile applications for image inpainting and object removal based on our model and some traditional methods, to make this technique into practical usage.

References

[1] M. V. Afonso, J. M. Bioucas-Dias, and M. A. T. Figueiredo. An augmented lagrangian approach to the constrained optimization formulation of imaging inverse problems. *IEEE Transactions on Image Processing*, 20(3):681–695, March 2011.

[2] M. Bertalmio, A. L. Bertozzi, and G. Sapiro. Navier-stokes, fluid dynamics, and image and video inpainting. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, volume 1, pages I–355–I–362 vol.1, 2001.

[3] J. Hays and A. A. Efros. Scene completion using millions of photographs. In *ACM SIGGRAPH 2007 Papers*, SIGGRAPH '07, New York, NY, USA, 2007. ACM.

[4] C. Lu, J. Tang, S. Yan, and Z. Lin. Generalized nonconvex nonsmooth low-rank minimization. *CoRR*, abs/1404.7306, 2014.

[5] D. Pathak, P. Krhenbhl, J. Donahue, T. Darrell, and A. A. Efros. Context encoders: Feature learning by inpainting. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2536–2544, June 2016.

[6] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 234–241. Springer, 2015.

[7] A. Telea. An image inpainting technique based on the fast marching method. *J. Graphics, GPU, & Game Tools*, 9(1):23–34, 2004.

[8] R. A. Yeh, C. Chen, T. Lim, M. Hasegawa-Johnson, and M. N. Do. Semantic image inpainting with perceptual and contextual losses. *CoRR*, abs/1607.07539, 2016.

[9] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva. Learning deep features for scene recognition using places database. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 487–495. Curran Associates, Inc., 2014.