Comparing and Visualizing Image Inpainting Techniques

Alan Yue
GANs section

Vincent Nguyen
Diffusion section

Abstract

There is a common and increasing demand for photo editing and modification software to remove, replace, and fix broken images. In this paper, we will examine popular implementations of GAN and diffusion methods for image inpainting - the rebuilding and restoration of missing regions in incomplete or broken images. We will attempt to reproduce some experimental results on a new image domain by tuning the models and compare the strengths and weaknesses of each.

1 Introduction

Image inpainting is a technique that seeks to reconstruct portions of an image based on the rest of the image, whose broad qualitative objective is to appear natural to the human eye. Applications can range from removing unwanted objects, text, or to fill in parts of an image that were lost or corrupted. Traditional approaches have mostly used either patches, which look for similar clumps of pixels, or diffusion, which gradually diffuse pixels around the area to be restored (1). However, as with many other areas, some of the most promising results have come hand in hand with advances in deep learning.

2 Literature Review

Convolutional neural networks are a staple of computer vision and are thus a natural approach to take for image inpainting. Convolutional filters are typically run over both the observed and predicted pixels of an image; Liu et al propose using partial convolutions that act on only the known pixels to prevent conditioning on assumed patterns (2). Generative adversarial networks are useful in that they try to make it hard to distinguish if an image has been inpainted, similar to our main qualitative objective previously mentioned. Demir and Unal use PatchGAN (3) coupled with a global GAN in an effort to learn both local and holistic features for inpainting [4]. CNNs and GANs together represent two of the most used classes of deep learning models in image inpainting today (1).

Recently, there have been diffusion inpainting techniques that can be used for even the most complex masks.(4) To control the generation process, we only modify the reverse diffusion iterations by sampling the unmasked regions using the given image information.(4) As this method does not change or condition the original DDPM network, the model produces high-quality and diverse output images for any inpainting task.

In what follows we will give an overview of a particular implementation of each model, but as our focus is on the application, tuning, and comparison of the models, we will not go beyond what is necessary for understanding these parts and we believe it is redundant to do so; we refer the reader to the full papers for full details.

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

3 GANs

The specific GAN model we are going to take a look at is EdgeConnect (5), which first attempts to learn what edges are missing from the image before inpainting based on those edges. There's a generator and discriminator for both parts of the model to ensure that a visually consistent image emerges as the product. The generators each use dilated convolutional layers and downsample twice before upsampling back to the original image size. The discriminators adopt PatchGAN (3), which determines if overlapping image patches are real or not.

The input to the edge generator G_1 consists of the grayscale image $\tilde{I}_{gray}$ along with an edge map C_{gt} and image mask M as a precondition:

$$C_{pred} = G_1(\tilde{I}_{gray}, C_{gt}, M)$$

The loss of the generator and discriminator D_1 consists of an adversarial and feature mapping component:

$$min_{G_1} max_{D_1} \mathcal{L}(G_1) = min_{G_1}(\lambda_{adv,1} max_{D_1}(\lambda_{adv,1}) + \lambda_{FM}\mathcal{L}(FM))$$

For our experiment, we used the code from their public repo (6) and tried to test EdgeConnect to see if it could generalize to drawings rather than the photo-realistic images it was trained on. We took the pre-trained weights from the model trained on the Places2 (7) dataset and further trained it on pictures of stegosauruses from the Caltech 101 (8) dataset. Each picture was randomly paired with an irregular mask from the dataset used by Liu et al in (9). We trained on 50 pictures in batch sizes of 5, with 5 being used as a validation set and the final 4 as a test set. It has been observed that a small amount of additional training can be enough to orient a pre-trained neural network towards a new task, so we wanted to test that hypothesis. Because of the small number of update steps, we increased the learning rate by a factor of 100 from the default 0.0001 in the code provided by the authors to 0.01.

The λs in the loss equation are regularization parameters. For our experiment, we chose to scale up $\lambda_{adv,1}$ by a factor of 10, so that we emphasize the model producing edges similar to that of other stegosaurus drawings. We also slightly decreased the edge detection threshold from 0.5 to 0.4 as some of the edges in the drawings appeared finer than in real life photos.

Training took 13.6 minutes on the Python 3 Google Compute Engine backend CPU after which we were able to produce a visualization of the edges generated for the test images and corresponding masks. From the visualization 1, it appears as if the network is having a hard time dealing with the many lines present in each of the drawings, as the central column shows an overabundance of edges that obscure the larger image. This may be due to the small size of the training set and shorter training time. While the authors fail to provide any concrete time frames when discussing the model's training time, they do mention in the acknowledgements that they had access to Nvidia's Titan V GPU. Based on that and the training time of other image networks (10), we assume it was longer than it would have been feasible for us to reproduce. We were unable to run the second GAN on our dataset nor reproduce the results of the images in the paper, as the code in the public repo would not run on the examples they had provided.

Given that the model essentially tries to mimic how artists often work, where lines are drawn first that determine the boundaries of colors, we expect some peformance boost shown by some of the impressive results given in the paper. However, the longer time needed to train two GANs concurrently on a new image domain may not be worth it, even when using pre-trained weights as we demonstrated here. Additionally, it may be difficult to predict where edges lie in masks with larger holes, in which case another method such as diffusion that gradually inpaints the hole may be better suited.

4 Diffusion

A fast semi-automatic method proposed by Oliviera et al (11) involved iteratively convolving the region to be inpainted with a diffusion kernel (for example, with a Gaussian kernel), progressing from the boundary of the region inwards each iteration. However, this process can leave undesirable artifacts in the image, such as blurring across edges in the area to be inpainted. Oliviera et al proposed

a solution to this by introducing diffusion barriers which may be placed manually on the image, which can help preserve edges by stopping the above process when it hits one of the pixels of the barrier.

The first diffusion method we tried was the Fast Inpainting method described in the paper by Oliviera et al (11). In this method, we let Ω be the region to be inpainted, and $\partial\Omega$ be a one pixel thick boundary of said region. The proposed method first clears the colour information in Ω, then repeatedly convolves the region with a diffusion kernel. This process continues until either the difference in change after convolution is insignificant, or a certain number of iterations is complete. For this experiment, we manually added masks to images individually, then ran the algorithm individually to see its effect. For our parameters, we set our maximum iterations as 500, and our threshold boundary as 0.1. We determined that an this provided a balance of completion of filling in our pictures while being realistic with how much the algorithm can accomplish. As our threshold boundary increased past our chosen values, there was an increase in blurring. An example of standard results is shown in figure 2.

It requires the inpainted region to be locally small, else it cannot perform correctly. It also has a hard time preserving both structural and textural information if the image is complex enough. This results in a blurriness over the inpainted region. One of the major downsides is that the inpainting region, or Ω, must be locally small. As Oliveira suggests, smaller regions allow for simpler models to be used to approximate the results. This allows for faster run times for a more limited scope. However, larger regions will result in errors.

In addition, as we are using convolution of gaussian kernels, blurring will naturally occur. Thus if an image is visually dense with a lot of textural information, then the algorithm fails to reproduce it. The algorithm also does not keep track of structural information such as edges. Thus edges connected to the inpainted region will not be recreated to it's full accuracy.

Many current techniques train for a particular arrangement of masks, which restricts their generalization capacities to obscure cover types. Furthermore, training with pixel-wise and perceptual losses frequently results in basic textural augmentations towards the missing parts instead of semantically significant creation. The second diffusion paper we examine is RePaint: A Denoising Diffusion Probabilistic Model (DDPM) based inpainting approach that is suitable for even extreme masks. (4)

In the course of training, DDPM approaches establish a diffusion process that alters an image x0 to a white Gaussian noise xT that follows a normal distribution (0, 1) over T time periods. Each step of the progression is specified by:

$q\left(x_t \mid x_{t-1}\right)=\mathcal{N}\left(x_t ; \sqrt{1-\beta_t} x_{t-1}, \beta_t \mathbf{I}\right)$

The DDPM is trained to undo this. This reversal is represented by a neural network that anticipates the values of (xt, t) and (xt, t) of a Gaussian distribution:

$p_\theta\left(x_{t-1} \mid x_t\right)=\mathcal{N}\left(x_{t-1} ; \mu_\theta\left(x_t, t\right), \Sigma_\theta\left(x_t, t\right)\right)$

As extended by Ho et al. (12), this loss can be further decomposed as,

$$\mathbb{E}_q[\underbrace{D_{\mathrm{KL}}\left(q\left(\mathbf{x}_T \mid \mathbf{x}_0\right) \| p\left(\mathbf{x}_T\right)\right)}_{L_T}+\sum_{t>1} \underbrace{D_{\mathrm{KL}}\left(q\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0\right) \| p_\theta\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t\right)\right)}_{L_{t-1}} \underbrace{-\log p_\theta\left(\mathbf{x}_0 \mid \mathbf{x}_1\right)}_{L_0}]$$

It has been demonstrated by Ho et al. (12) that the most effective way to parametrize the model is to forecast the total noise ϵ_0 that is added to the current intermediate image x_t. This leads to a closed form expression of the objective since $\left(\mathbf{x}_{l-1} \mid \mathbf{x}_t, \mathbf{x}_0\right)$ is also Gaussian (12). As a result, the term L_{t-1} trains the network (2) to execute one reverse diffusion step, allowing for the aforementioned parametrization of the predicted mean $\mu_\theta\left(x_t, t\right)$.

$$\mu_\theta\left(x_t, t\right)=\frac{1}{\sqrt{\alpha_t}}\left(x_t-\frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}} \epsilon_\theta\left(x_t, t\right)\right)$$

In RePaint, since the forward process is determined by a Markov Chain of additional Gaussian noise, we can take a sample of the intermediate image x_t at any stage in time.(4) This permits us to take a sample of the known regions $m \odot x_t$ at any moment t. Therefore, we obtain the following expression for one backward step in our approach,

$x_{t-1}^{known} \sim \mathcal{N}\left(\sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t)\,\mathbf{I}\right)\, x_{t-1}^{unkown} \sim \mathcal{N}\left(\mu_\theta\left(x_t, t\right), \Sigma_\theta\left(x_t, t\right)\right) x_{t-1} = m \odot x_{t-1}^{known} + (1-m) \odot x_{t-1}^{unknown}$

Hence, the known pixels in the given image $m \odot x_0$ are used to sample x_{t-1}^{known}, while the model is used to sample $x_{t-1}^{unknown}$ based on the prior iteration x_t. (4)

We used libraries from dalle-pytorch(13) and their pre-trained models, then further trained it with Python 3 Google Compute Engine backend CPU on pictures of stegosauruses from the Caltech 101 (8) dataset. Each picture was paired with randomly generated masks. However, the models ran into issues when analysed on new datasets and were not analysed. We thus examine some simpler randomly generated images and their masks. We use images of size 256, and resize our training sets to match this. We used T = timesteps of 1000 and conducted 10 replications with a jump size of 10. To condition the generation process, we only modify the inverse diffusion iterations by sampling the unmasked regions using the provided image data. (4) We validate our strategy for both faces and general-purpose image inpainting using standard and extreme masks. (4)

5 Conclusion.

We examine various models for image inpainting. The specific GAN model we review us EdgeConnect (5), which first attempts to learn what edges are missing from the image before inpainting based on those edges. Most GAN-based Image inpainting methods are prone to deterministic transformations due to the lack of control during the image synthesis. However, similar to super-resolution methods that leverage the Style-GAN latent space, it is to limited specific scenarios like faces.

We analyze straightforward inpainting approaches such as fast inpainting; a technique of inpaint that is not only speedy but simple to implement. Nevertheless, it necessitates the inpainted area to be locally small, else it cannot function properly. We also evaluate a mask-independent approach that broadens the selection of masks for free-form inpainting. Nevertheless, RePaint is dependent on an unconditional pretrained DDPM. Thus, the algorithm may be biased towards the dataset on which it was trained.

References

[1] X. Zuoa, Y. Yangb, and Z. Haoc, "Performance comparison of irregular face inpainting via deep learning," *Academic Journal of Computing & Information Science*, vol. 5, no. 3, pp. 70–77, 2022.

[2] G. Liu, F. A. Reda, K. J. Shih, T.-C. Wang, A. Tao, and B. Catanzaro, "Image inpainting for irregular holes using partial convolutions," in *Proceedings of the European conference on computer vision (ECCV)*, pp. 85–100, 2018.

[3] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1125–1134, 2017.

[4] A. Lugmayr, M. Danelljan, A. Romero, F. Yu, R. Timofte, and L. V. Gool, "Repaint: Inpainting using denoising diffusion probabilistic models," *ArXiv*, vol. abs/2201.09865, 2022.

[5] K. Nazeri, E. Ng, T. Joseph, F. Z. Qureshi, and M. Ebrahimi, "Edgeconnect: Generative image inpainting with adversarial edge learning," *arXiv preprint arXiv:1901.00212*, 2019.

[6] "Github - knazeri/edge-connect," in *https://github.com/knazeri/edge-connect*. commit 1c0b0ac7b77f43f6e9222c98309e29c4320e757c.

[7] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba, "Places: A 10 million image database for scene recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[8] F.-F. Li, M. Andreeto, M. Ranzato, and P. Perona, "Caltech 101," Apr 2022.

[9] G. Liu, F. A. Reda, K. J. Shih, T.-C. Wang, A. Tao, and B. Catanzaro, "Nvidia irregular mask dataset," in *https://nv-adlr.github.io/publication/partialconv-inpainting*, 2018.

[10] S. Iizuka, E. Simo-Serra, and H. Ishikawa, "Globally and locally consistent image completion," *ACM Transactions on Graphics (ToG)*, vol. 36, no. 4, pp. 1–14, 2017.

[11] M. M. Oliveira, B. Bowen, R. McKenna, and Y. Chang, "Fast digital image inpainting," in *Proceedings of the IASTED International Conference on Visualization, Imaging and Image Processing (VIIP 2001), Marbella, Spain, September 3-5, 2001* (M. H. Hamza, ed.), pp. 261–266, ACTA Press, 2001.

[12] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *CoRR*, vol. abs/2006.11239, 2020.

[13] "Github - lucidrains/dalle2-pytorch," in *https://github.com/lucidrains/DALLE2-pytorch/tree/1892f1ac1d63eff5cbe80a7a1f11ded2cc570665*.

6 Appendix

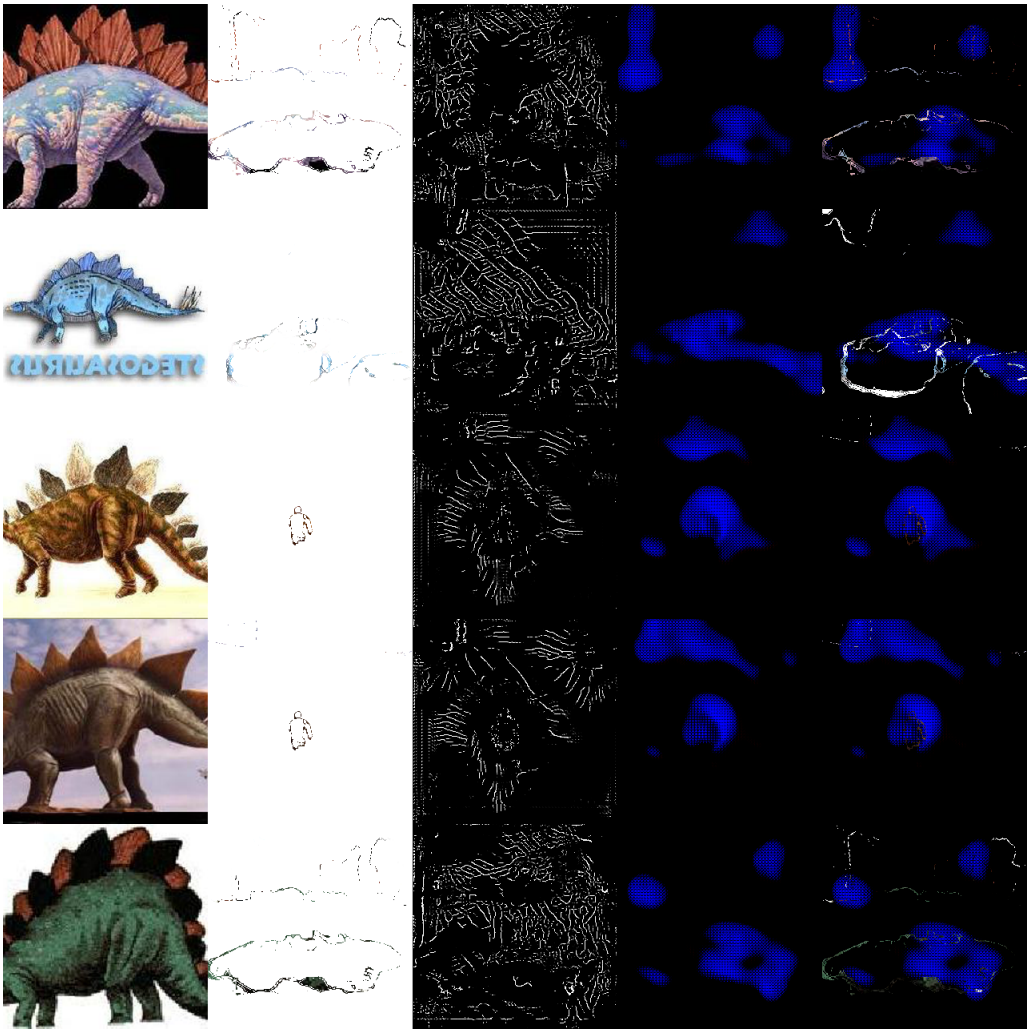
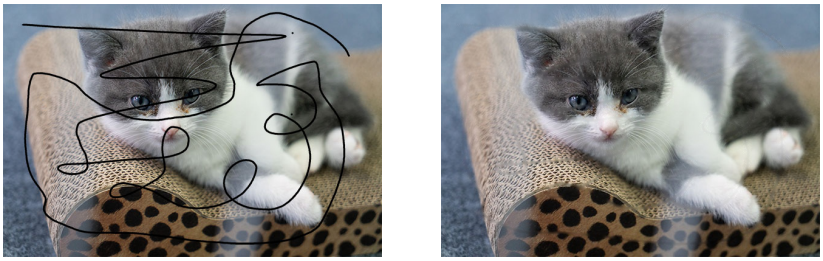


Figure 1: Stegosaurus edges and masks



(a) Damaged Image (b) Restored Result

Figure 2: Result of Fast Inpainting

Figure 3: Crease - Fast Inpainting

Figure 4: Various Mask Types- Fast Inpainting