Image Inpainting Using a Generative Adverserial Network

M. Adeel Hassan
University of Massachusetts Amherst
muhammadadee@umass.edu

1. Introduction

1.1. Problem Statement

This project implements a solution to the problem of image inpainting based on the paper, "Globally and Locally Consistent Image Completion" by Iizuka et al. [4]. Image inpainting is the task of filling in missing regions, or 'holes', in images in such a way that the completed image looks realistic. It has major applications in photo-editing (for example, for removing objects in images).

The paper introduces the novel idea of using not one but two discriminators for adversarial training, with one discriminator focusing on the quality of the completed image as a whole and the other on a smaller local region containing the completed portion of the image.

The paper authors train the model on a dataset of 8 million images, on 4 GPUs, for 2 months, and evaluate the quality of the model output by surveying humans. These methods cannot be reproduced in this project due to obvious limitations, so compromises are made.

For this project, the model is trained on the CIFAR10 dataset, consisting of 50,000 32x32 images. The total training time is on the order of a few hours. And the results are evaluated quantitatively on per pixel loss for the completed regions in the images, and qualitatively on the author's subjective judgement.

2. Related work

Inpainting is a well researched task in Image Processing, and there were a number of established statistical solutions prior to the past few years when deep learning-based solutions became popular.

Some of the old methods include:

- Diffusion-based image synthesis [2], in which features from the regions around the hole are propagated inwards.
- Patch based approaches, where the missing region is filled in with patches sampled from other parts of the image. PatchMatch [1], is an algorithm for speeding up patch matching used in these approaches.
- Approaches based on graph cuts and energy optimization [5], [6].

The first major work to apply deep learning to image inpainting was by Xie et al. [9] in 2012. Then in 2016, [8] applied adversarial training to good effect. A number of different works, includeding the paper this project is based on, using deep learning based approaches were proposed in 2017, such as [11], [10], and [3].

3. Data

The dataset used for this project is the CIFAR10 dataset. It comprises 50,000 32x32 images. The images are divided into 10 classes for classification tasks, but that is not relevant for this project.

A train-val-test split of 90%-5%-5% is used. The data is augmented during training by including random horizontal flips of images.

4. Approach

4.1. Architecture

The project uses the same architecture as the paper, shown in Figure 1.

4.1.1 Generator

The Generator is fully convolutional and uses an autoencoder-like architecture, in which the image is sampled down to 1/4th its original size using convolutions with stride 2, and then scaled up again using deconvolutions to the output size (which is equal to the input size). In the middle, dilated convolutions are applied as a way of increasing each neuron's receptive field.

4.1.2 Discriminator

The Discriminator is made of two different networks.

The *Global Discriminator* takes in the entire image as input and applies successive convolutions of stride 2 and has a linear layer of size 1024 as the last layer.

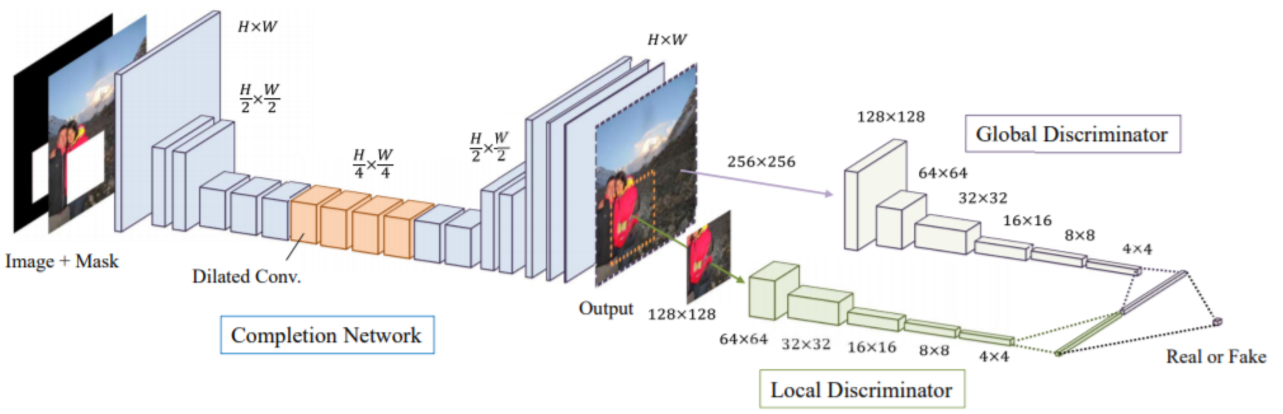

Figure 1. The GAN architecture

The *Local Discriminator* is basically the same as the Global one, except that it takes as input a smaller region that contains and is centered around the missing region that the generator has filled. This region is always a fixed size, which, in this project is 16x16 pixels.

4.2. Masks

The masks used for representing holes are arrays of zeros with ones in the positions that comprise the whole. For training, the masks were randomly generated. They were rectangular in shape with uniformly randomly (and independently) sampled heights and widths in the range 5 to 12 pixels.

5. Training

5.1. Loss

The paper uses the traditional Cross Entropy GAN loss for the generator and discriminators, with the addition that the generator loss also contains a term for the mean squared loss with the ground truth pixels for the completed region.

$$L(x, M_c) = \| M_c \odot (C(x, M_c) - x) \|^2$$

Where M_c are the random masks, x the input, and $C(x, M_c)$ the completed image. This additional term and the discriminator-based generator loss are combined using a weighted sum, controlled by the hyperparameter α.

$$\min_C \max_D \mathbb{E}[\ L(x, M_c) + \alpha \log D(x, M_d) \\ + \alpha \log(1 - D(C(x, M_c), M_c))\ ],$$

This project replaces the cross-entropy loss for the generator and discriminators with squared loss, since it has been shown to lead to better and more stable gradient propagation [7].

5.2. Other details

For each batch, random masks were generated. These were applied to the input images and concatenated with them as an additional color channel before passing them onto the generator.

To train the discriminator to learn to recognize real images, images without any holes but with random masks attached were passed to it.

The training was broken up into 3 phases:

- Generator-only training using just the squared error loss for the completed region
- Discriminator-only training with the Generator weights frozen
- Combined training with the full GAN + MSE loss for the generator, and the GAN loss for the discriminator.

A batch size of 1024 was used for training. The training was run for about a couple of dozen epochs (a few hours). Adam was used as the optimizer. The value of α was set to 0.0005.

The training was done on Google Colab, which has a Tesla T4 as a GPU with 15 GB of memory. All the code was written in Python using PyTorch and TorchVision.

6. Evaluation

The model achieved a per-pixel squared loss of 0.0155.

Some output images are shown in Figures 2 and 3. Figure 2 images are from the test set. Figure 3 images are from the validation set.

It can be seen that the model is starting to get the colors right (even when there are multiple colors in the region, as in the second row). It is not yet able to produce convincing textures. It would likely take a lot more training time than was feasible for this project.

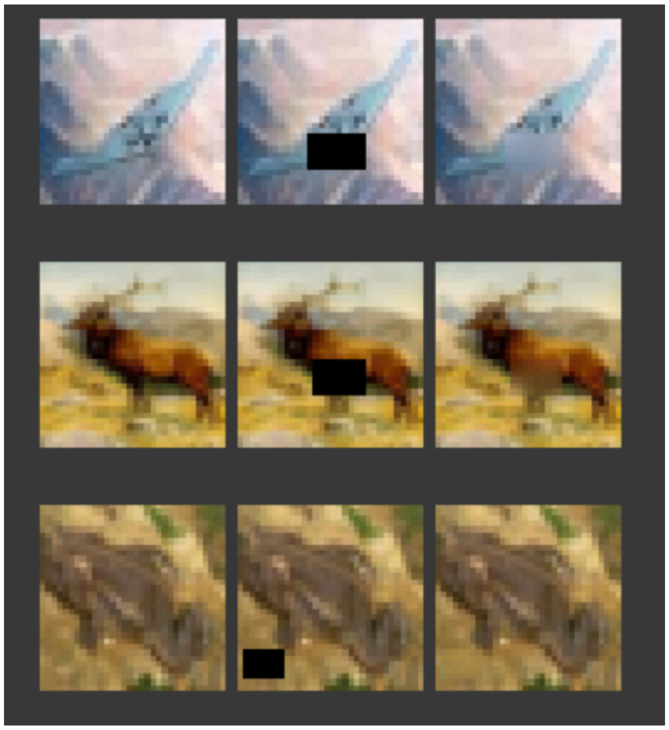
Figure 2. Results

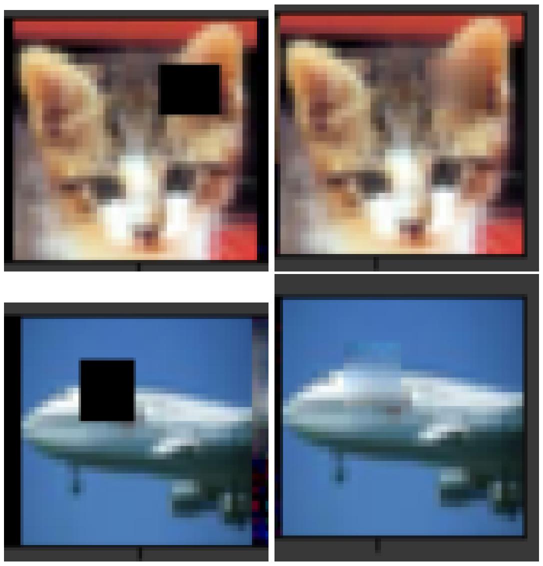
Figure 3. More results

References

[1] Connelly Barnes, Eli Shechtman, Adam Finkelstein, and Dan B Goldman. Patchmatch: A randomized correspondence algorithm for structural image editing. *ACM Trans. Graph.*, 28(3):24:1–24:11, July 2009.

[2] Marcelo Bertalmio, Guillermo Sapiro, Vincent Caselles, and Coloma Ballester. Image inpainting. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '00, pages 417–424, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[3] Nian Cai, Zhenghang Su, Zhineng Lin, Han Wang, Zhijing Yang, and Bingo Wing-Kuen Ling. Blind inpainting using the fully convolutional neural network. *The Visual Computer*, 33(2):249–261, 2017.

[4] Satoshi Iizuka, Edgar Simo-Serra, and Hiroshi Ishikawa. Globally and locally consistent image completion. *ACM Transactions on Graphics (ToG)*, 36(4):107, 2017.

[5] Vivek Kwatra, Arno Schödl, Irfan Essa, Greg Turk, and Aaron Bobick. Graphcut textures: Image and video synthesis using graph cuts. *ACM Trans. Graph.*, 22(3):277–286, July 2003.

[6] Yunqiang Liu and Vicent Caselles. Exemplar-based image inpainting using multiscale graph cuts. *IEEE transactions on image processing*, 22(5):1699–1711, 2013.

[7] Xudong Mao, Qing Li, Haoran Xie, Raymond YK Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2794–2802, 2017.

[8] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A. Efros. Context encoders: Feature learning by inpainting. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[9] Junyuan Xie, Linli Xu, and Enhong Chen. Image denoising and inpainting with deep neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 341–349. Curran Associates, Inc., 2012.

[10] Chao Yang, Xin Lu, Zhe Lin, Eli Shechtman, Oliver Wang, and Hao Li. High-resolution image inpainting using multi-scale neural patch synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6721–6729, 2017.

[11] Raymond A Yeh, Chen Chen, Teck Yian Lim, Alexander G Schwing, Mark Hasegawa-Johnson, and Minh N Do. Semantic image inpainting with deep generative models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5485–5493, 2017.