CS230

Old Photo Restoration: Pix2Pix vs Partial Convolutions

Nivedita Rahurkar
Cisco Inc.
Stanford University
rahurkar@stanford.edu

Daria Reshetova
Department of Electrical Engineering
Stanford University
resh@stanford.edu

Abstract

Old photo restoration is necessary since many prints of photo taken decades ago with film camera are no longer in good condition. Moreover, doing the restoration manually can be very tedious and requires expertise in some image processing software like Photoshop, Affinity photo, Gimp etc. Therefore we propose a novel deep learning based approach where the user only provides the damaged image to the network and the network outputs the restored image. We achieve this using two distinct networks, one that uses a cGAN and the other that uses a VGG-16 based U-Net but with convolutions replaced by partial convolutions [7]. We also simulate the damaged input images using masks generated by randomly applying lines, circles and ellipses of varying sizes and number at distinct parts of the image, since the available training data for damaged image and its restored counterpart was limited. Finally, we compare the two results produced by both the networks.

1 Introduction

Many people nowadays have archives of printed photos, taken 50 or more years ago, some of which have white dots, stripes, creases and are even ripped apart. These pictures can be restored using image processing software, ex. Photoshop, but that requires a considerable amount of time, effort and special skill. We tried to build a DNN to automate the task.

More specifically, the input to our algorithm is a damaged grayscale image. We then use a neural network to fill in the damaged parts. We compared the following 2 network types:

1. A cGAN [3]. Both the generator (U-Net) and the discriminator have access to the input image (a damaged image in our case) and are trained in an adversarial manner.
2. A PConv: It is a U-net based architecture with modified convolutional layers. It uses mask-aware convolutions(partial convolutions) instead of the standard convolutions to account for the fact that the initial image has to stay unchanged outside of the mask.

The performance of the networks is measured visually as the two approaches have different loss function types (e.g. the cGAN loss involves the generator loss, while a partial convolutions network doesn't have a discriminator).

2 Related work

To the best of our knowledge, the problem of restoring damaged photographs is usually solved using software like Photoshop. We could not find any previous work on automating the task.

CS230: Deep Learning, Winter 2018, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

Nevertheless, image-to-image translation tasks have recently been looked at from the perspective of neural networks. Unconditioned GANs have been successfully applied to the tasks of image inpainting [9], style transfer [5] and image superresolution [4]. The closeness of the result to the input image (the conditioning) was primarily attained by adding the distance between the desired input and the output of the generator (e.g. L2 regression) to the loss.

Conditional GANs deffer from the unconditioned ones in the fact that they are not application specific. They have been applied to colorization, scene change [3] and digits generation [8].

Our problem is most similar to image inpainting, which was successfully done by a pix2pix [3] architecture in case of rectangular holes and partial convolutions [7] for the irregular ones.

3 Dataset and Features

Our initial plan was to train both the networks on real damaged images. However we could only find 208 damaged images having corresponding restored images available when we searched online. Therefore we decided to train our network on simulated damaged images. We simulated the damage by applying a grayscale mask on top of images from the Coco dataset[6]. We rescaled all the images to have a size of 256×256.

Mask Generation The damage that we saw was mostly due to creases, folds or image sticking to the photo album. This led to missing pixel values which appear as white patches in the image. Due to the limitation of damaged images dataset we went with the assumption that most damaged pixels are white (more than 0.9 * max intensity). We randomly generated masks containing circles, lines and ellipses of varying sizes and number. Figure 1 shows the sample masks.

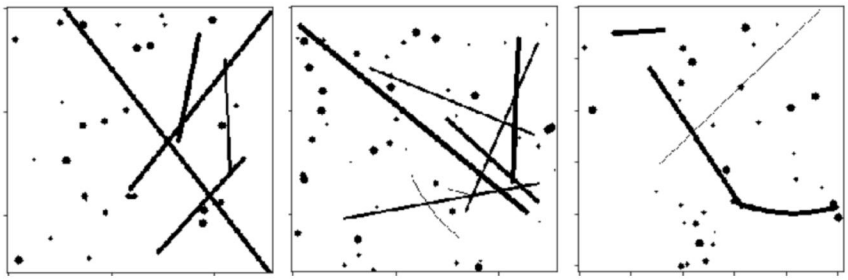


Figure 1: Simulated Mask Generation

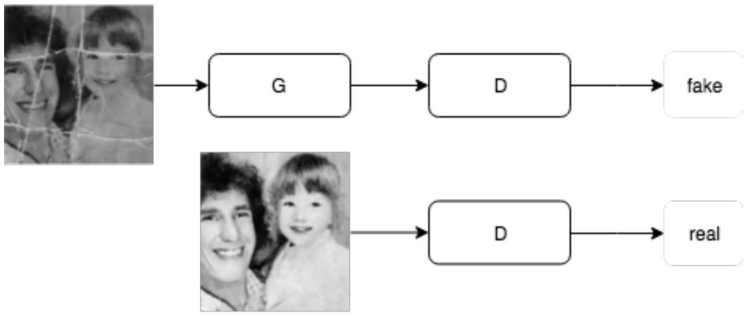
We tested on the real damaged images. To eliminate extra improvements done by photoshop artists and make the dataset more uniform, we created masks of damages and then fed masked photoshopped images to the network. The masks were created by first aligning the images using structural similarity index, feature extraction [10] and homography transform in opencv [1]. Then we used the difference of the Fourier transforms of the 2 images to get the noise mask.

4 Methods

As previously mentioned, we used 2 types of architectures: cGAN and partial convolutions. The reason for choosing them is that the partial convolutions most closely correspond to the task, while conditional GANs with a patch discriminator should perform well on the small amounts of data as a patch discriminator has relatively few training parameters.

4.1 Conditional GAN

The conditional GAN consists of a generator G and the discriminator D. The generator is trained to produce images from the same distribution as the output images conditioned on the input images (i.e. indistinguishable from them), while the discriminator is trained to distinguish the output of the generator and the desired output images. The desired behaviour of the discriminator is shown on the following diagram, while generator is trying to break that behaviour.



Loss We used a cGAN loss combined with L1(MAE) loss on the generator to encourage less blurring: if x is the damaged image and y is the restored image:

$$\mathcal{L}(G, D) = \mathbb{E}_y[\log(D(y))] + \mathbb{E}_{(x)}[\log(1 - D(G(x))] + \lambda\mathbb{E}[\|y - G(x)\|_1]$$

The objective is a minimax problem:

$$G^* = \underset{G}{\operatorname{argmin}} \max_{D} \mathcal{L}(G, D)$$

Architecture We used U-Net architecture (encoder-decoder convolutional NN with skip-connections between corresponding encoder and decoder layers), shown at figure 4.2 The filter sizes for the encoder were $[64, 128, 256, 512, 512, 512, 512, 512]$ – the middle layers have less parameters than the actual pix2pix framework as the larger network gave an out of memory error when training. The generator network had a total of 70,407,617 trainable parameters> For the discriminator we used a patch framework, i.e. the decision is made for each patch of the generator output separately. We tried 128×128 and 64×64 patch sizes. The architecture of the discriminator is the same as the one of the encoder with filter sizes doubling and image size shrinking by a factor of 2 each layer.

4.2 Partial Convolutions (Pconv)

In this approach we treat damaged image restoration as an inpainting problem. Pconv is a UNet-based architecture where convolution layers are replaced with partial convolution layers [7]. We can see the model architecture below.

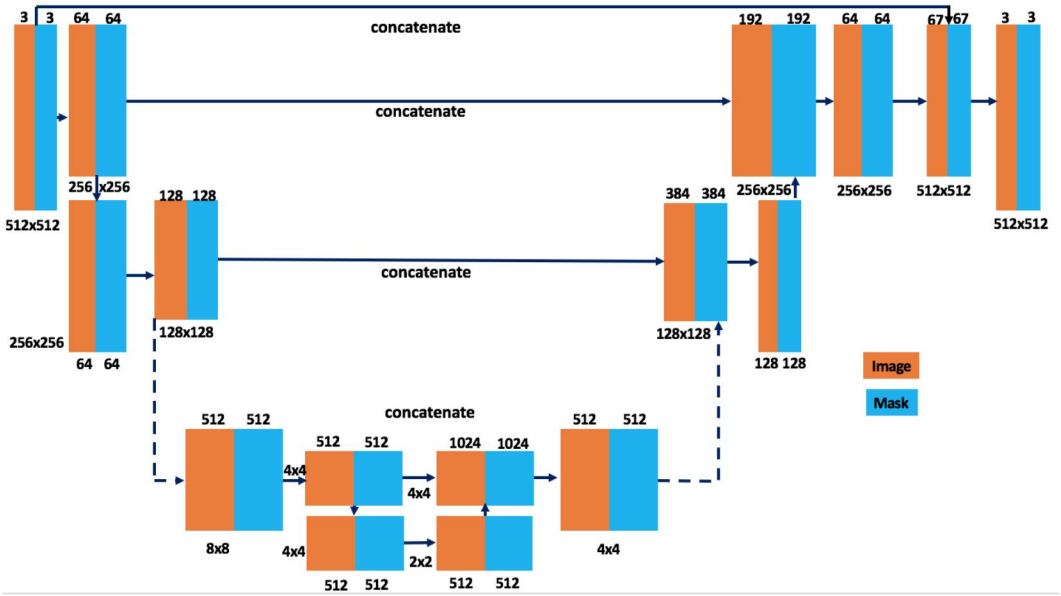


Figure 2: Model Architecture

Partial Convolution layer: A regular convolution layer is substituted with partial convolution followed by mask update step. For the convolutional filter W and corresponding bias b, X_{in} is the

feature pixel values for the current convolution and M is the corresponding binary mask. The output of the convolution layer is :

$$X_{out} = \begin{cases} W^T(X_{in} \odot M)\frac{1}{\sum_{ij} M_{ij}} + b & \text{if } \sum_{ij} M_{ij} > 0 \\ 0, & \text{otherwise} \end{cases} \qquad M_{out} = \begin{cases} 1, & \text{if } \sum_{ij} M_{ij} > 0 \\ 0, & \text{otherwise} \end{cases}$$

M_{out} is the updated mask after the convolution. After every partial convolution, if the convolution was able to condition its output on at least one valid input, then the mask is removed at that location.

5 Training

5.1 cGAN

We used [11] as our starting point and modified it to work with different patch sizes.

We decided the values for the parameters based on the learning curve and the images acquired during training. We chose $\lambda = 10^{-2}$ as this decreased the blurring, while further increasing λ lowered the performance of the discriminator. We used Adam's optimizer to to optimize the loss and a batchsize of 4, as larger batches gave an out-of-memory error.

During training of both networks, we increased the number and size of the generated damage, which corresponds to the peaks in the learning curve. The largest peak in training the 128×128 patch network corresponds to the noise that cannot be excluded, thus we decided not to increase the damage up to this level when training the 64×64 patch cGAN.

The training curves are presented at figure 3. The last peak corresponds to getting from the generated damage to damage masks obtained from the real damaged pictures as described in section 3.

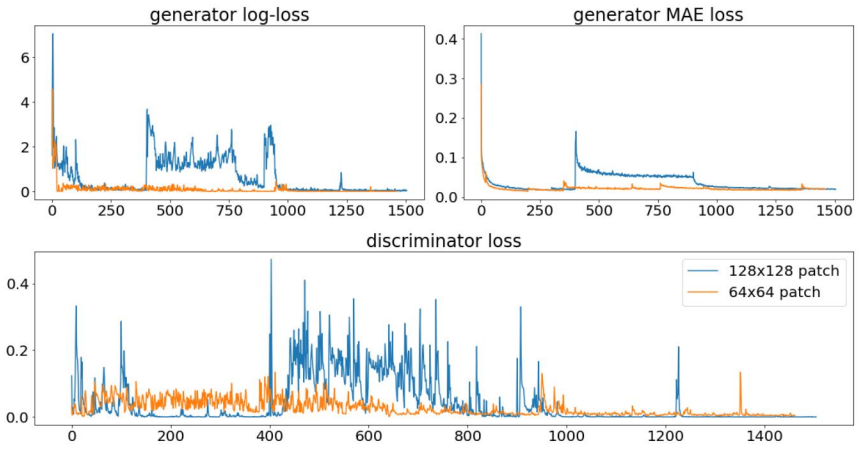


Figure 3: Losses in Pix2Pix network

5.2 PConv

To begin with, we started with the existing implementation [2] of the image inpainting problem in keras. We initialized our Partial convolutional network with VGG16 weights. We used a learning rate of $\lambda = 2 \times 10^{-4}$. For this network too we used Adam's optimizer and batch size = 4. We trained the network with number of epochs = 63, steps per epoch = 1000 and validation steps = 100. We used the same loss function as described in [7]. It consists of:
L1 losses for both masked and un-masked regions, Perceptual loss, Style loss on VGG-16 features both for predicted image and for computed image (non-hole pixel set to ground truth) and Total variation loss for a 1-pixel dilation of the hole region. The loss function is weighted as follows:

$$\mathcal{L}_{total} = \mathcal{L}_{valid} + 6\mathcal{L}_{hole} + 0.05\mathcal{L}_{perceptual} + 120(\mathcal{L}_{style_{out}} + \mathcal{L}_{style_{comp}}) + 0.1\mathcal{L}_{tv}$$

Midway through our training we had to increase the amount of damage in simulated images as the restoration was poor for heavily damaged images. Hence the sudden increase in the loss as seen in the Figure 4.

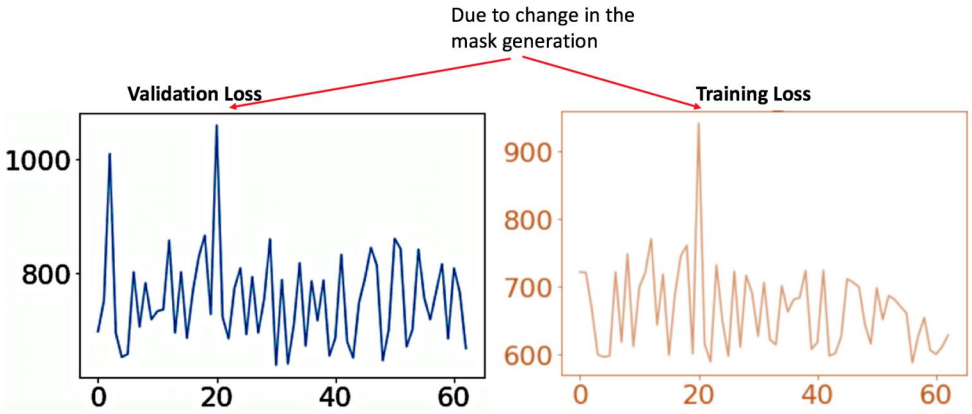


Figure 4: Validation and Training Loss in Partial Convolution as a function of epochs

6 Results

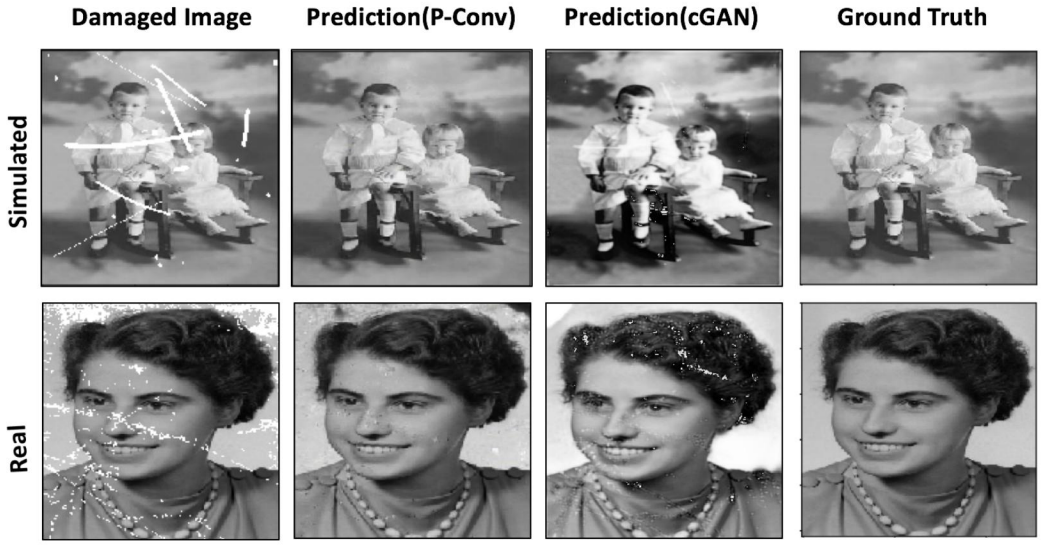


Figure 5: Sample Results

We observed that the cGAN performed better around the edges. However, it modified a few non damaged pixels, for eg. the hair in the second image in Figure 5. While the Pconv performed well on the overall damaged regions except on the corners.

7 Conclusion

The restoration results obtained on simulated damaged images are better than the real damaged images since both the images have different distributions. We observed that restoration fails near corners which we believe is due to : 1) Less information is available from neighborhood patches around corners as compared to interior patches. 2) In the training dataset we had limited images with damage around corners in the training set. All in all, both networks performed almost equally well.

8 Future Work

In future, we plan to resolve the issue of poor results at damaged corners by generating more masks with non-zero pixels around corners for training. We would also like to estimate the damaged regions in an image better by training it for the same. Finally we would like to color the image.

9 Contributions

Both authors contributed equally to the project. Nivedita was responsible for obtaining the Coco dataset and preprocessing it. She also performed random mask generation, damage generation and training of the Pconv network. Daria was responsible for collecting the damaged images dataset, preprocessing it and generating the damage masks. She trained of the cGAN networks. Both of us worked together on the project report, poster, research, planning and intermediate tasks.

Our code is on Github at : https://github.com/dashar/old-photo-restoration

References

[1] Gary Bradski and Adrian Kaehler. *Learning OpenCV: Computer vision with the OpenCV library*. " O'Reilly Media, Inc.", 2008.

[2] Mathias Gruber. Partial convolutions for image inpainting using keras. `https://github.com/MathiasGruber/PConv-Keras`, 2018.

[3] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. *arXiv preprint*, 2017.

[4] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew P Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *CVPR*, volume 2, page 4, 2017.

[5] Chuan Li and Michael Wand. Precomputed real-time texture synthesis with markovian generative adversarial networks. In *European Conference on Computer Vision*, pages 702–716. Springer, 2016.

[6] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan C., Lawrence Zitnick, and Piotr Dollar. Microsoft coco: Common objects in context. *arXiv:1405.0312v3 [cs.CV]*, 2015.

[7] Guilin Liu, Fitsum A Reda, Kevin J Shih, Ting-Chun Wang, Andrew Tao, and Bryan Catanzaro. Image inpainting for irregular holes using partial convolutions. *arXiv preprint arXiv:1804.07723*, 2018.

[8] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[9] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2536–2544, 2016.

[10] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *Computer Vision (ICCV), 2011 IEEE international conference on*, pages 2564–2571. IEEE, 2011.

[11] Fallcon William. Image-to-image translation with conditional adversarial networks (pix2pix) implementation in keras. `https://github.com/williamFalcon/pix2pix-keras`, 2017.