IMAGE INPAINTING USING GANS WITH PARTIAL CONVOLUTIONS

CS 230

Justin Lundgren, 06289145
julu1@stanford.edu

Wilhelm Bolin, 06318803
wbolin@stanford.edu

Nicolas Jersch, 06225324
njersch@stanford.edu

Stanford University

Abstract

We propose a Generative Adversarial Network (GAN) with partial convolutions for image inpainting. Using local and global discriminators, we show the importance of adversarial training to remove blur and produce crisp details. We visualize how skip connections in a symmetric U-Net-like architecture allow our model to learn increasingly finer details of an image. It is demonstrated that very good results can be achieved even with significantly smaller networks than state-of-the-art approaches such as [6].

Mask Input Output

Figure 1: Image Inpainting

1 Introduction

Pictures and images have become ubiquitous in our lives. Consequently, there is an increasing need for post-production and image editing. One important type of editing is to remove undesired parts of an image and replace them with suitable imagery that blends in with the rest of the image. This task is called *image inpainting*. Figure 1 shows an application of this with our specific mask.

Existing approaches to inpainting normally replace the masked holes with a substitute value, for example the mean value. However this approach often suffers from artifacts such as color differences or blur since the model output is also conditioned on these invalid placeholders. To circumvent this issue, we propose a Generative Adversarial Network (GAN) with partial convolutions [6] that only takes into account valid pixels. To the best of our knowledge, our approach is novel in that we are the first to train partial convolutions adversarially as part of a GAN.

2 Related Work

In recent years, computer vision has made significant progress on several tasks ranging from image classification and object detection to segmentation. This has been made possible due to the recent advancements in convolutional neural networks (CNNs). These advancements have further enabled progress in other related fields such as better understanding of image context and generation of realistic images using GANs [3].

The advancements in GANs have made content prediction and classification more powerful. These models have been used by many projects related to image inpainting as an approach to more accurately being able to predict the missing part of the image conditioned on the known image regions, as seen in [6, 3, 2, 7]. Pathak et al. [7] introduce a network structure built on an encoder-decoder architecture to predict the missing parts of an image using an adversarial loss. Their model performs poorly in generating low-level features and fine-detailed textures and tend to overfit to local features and produce a less realistic fill given the global features of the training data. To address this issue, Iizuka, Simo-Serra, and Ishikawa [4] introduce a GAN structure with a combination of local and global discriminators. This proved to produce globally and locally consistent results with realistic details.

More recently, a new type of convolutional neural network called U-Net was used by [9] in image inpainting for the first time. U-Net is a fully convolutional neural network initially developed for biomedical image segmentation with fewer training images while still yielding precise segmentations [8]. This structure proved to be useful in many domains and recently so in image inpainting [6, 9]. The U-Net consists of an AutoEnconder-CNN structure where additional skip connections are introduced to concatenate the features from each layer of the encoder and those of the corresponding layer of the decoder. These skip connections pass information over the central bottleneck layer, which helps to preserve some of the information outside the inpainted regions [9].

More recently, Liu et al. [6] have obtained very good results by using *partial convolutions* [6]. In contrast to standard convolutions, partial convolutional layers apply a binary mask before calculating the convolution. [6] find this approach to effectively reduce artifacts such as color discrepancies and blurry output.

3 Dataset

The selected dataset is Places 365 [1]. It consists of a large set of images of cities, buildings, parks

etc. Due to computational limitations, we use 30,000 images for training and 1,000 for testing. To increase training speed, we scale the images down to the size of $128 \times 128 \times 3$. Figure 2 shows a variety of examples from Places 365.

Figure 2: Examples from Places 365 [1].

We normalize the images in the training and test set by dividing all pixels with 255. After the normalization, a mask $\mathbf{M}$ containing two rectangular holes is applied to each training sample (see Figure 1). The mask is defined as a set of binary values $\mathbf{M} \in \{0, 1\}^{128 \times 128 \times 1}$ where a 0 signals that the corresponding pixels in the training sample is replaced by the average pixel value.

4 Methods

Our proposed model uses a DCGAN architecture with partial convolutions and global and local discriminators. To the best of our knowledge, training partial convolutions adversarially as part of a GAN is a novel approach which has not been attempted before. Figure 3 visualizes our GAN architecture. We discuss each different component below.

4.1 Partial Convolutional Layers

Existing approaches normally replace the masked holes with a placeholder value like the image mean. This often results in artifacts such as color inconsistencies or blurry output since convolutions are also applied on these invalid pixels. To address this issue, we use partial convolutional layers instead of typical ones as suggested by Liu et al. [6]. In contrast to standard convolutions, partial convolutional layers apply a binary mask before calculating the convolution. Each of these layers comprises two separate steps: (1) a masked convolution step and (2) a mask-update operation. In the convolution step, the partial convolution p_c at every sliding window is defined as

$$p_c = \begin{cases} \mathbf{W}^\top (\mathbf{X} \odot \mathbf{M}) c + \mathbf{b} & \text{if } \mathbf{1}^\top \mathbf{M} \mathbf{1} > 0, \\ 0 & \text{else} \end{cases}$$

where $\mathbf{X}$ is the pixel value of the current sliding window, $\mathbf{M}$ the corresponding binary mask, and $\mathbf{W}$ and $\mathbf{b}$ the filter weights. $c = w \cdot h / (\mathbf{1}^\top \mathbf{M} \mathbf{1})$ is a scaling factor, where w and h correspond to the width and height of the mask $\mathbf{M}$. In the second step, the mask is updated for the next layer where each position m in the mask is expressed as:

$$m = \begin{cases} 1 & \text{if } \mathbf{1}^\top \mathbf{M} \mathbf{1} > 0, \\ 0 & \text{else} \end{cases}$$

With sufficient applications of partial convolutional layers, the mask will eventually reach a state where it only contains ones and the partial convolution operates just like a normal convolution.

4.2 Generator

Our generator takes as input the masked image $\mathbf{I}_{in}$ and produces an inpainted image $\mathbf{I}_{out}$ of the same size as $\mathbf{I}_{in}$. The architecture follows a U-Net structure similar to Isola et al. [5] with encoder and decoder parts. In the encoder stage, partial convolutional layers successively map the image to a more compact latent feature representation to learn the semantics of the image. The decoder stage then uses this representation to reconstruct a realistic version of the full inpainted image. Following Isola et al. [5], we connect each encoding layer through a skip connection to the corresponding decoding layer. Each layer is followed by a leaky ReLU activation with $\alpha = 0.2$ except for a sigmoid activation in the last layer. We apply batch normalization after each layer to speed up training. Table 1 summarizes our generator architecture.

Layer	K	S	F	Activation
PConv	5	2	64	ReLu
PConv	5	2	128	ReLu
PConv	3	2	256	ReLu
PConv	3	2	256	ReLu
PConv	3	2	256	ReLu
PConv	3	1	256	Leaky ReLu
PConv	3	1	256	Leaky ReLu
PConv	3	1	128	Leaky ReLu
PConv	3	1	64	Leaky ReLu
PConv	3	1	3	Leaky ReLu
PConv	1	1	3	Sigmoid

Table 1: Generator structure, $K :=$ kernel size, $S :=$ stride, $F :=$ number of filters.

Figure 3: Network Architecture

4.3 Discriminators

Based on [4], we train both global and local discriminators to distinguish real from inpainted images. The global discriminator D_g takes into account the entire output image and produces a binary value indicating a real or inpainted image.

In addition, we use two local discriminators to assess different parts of the image separately. Specifically we use one local discriminator D_l that operates on the left third $\mathbf{I}_{out,l}$ of $\mathbf{I}_{out}$ containing the first of the two inpainted regions (see mask in Figure 1), and one local discriminator D_r operating on the remaining two thirds $\mathbf{I}_{out,r}$ containing the second inpainted region.

We finally combine the three discriminators into a single binary score $D(D_g, D_l, D_r)$ using a fully connected layer with a single unit and a sigmoid activation. Table 2 summarizes our discriminator architecture.

Layer	K	S	F	Activation
Conv	5	2	32	ReLu
Conv	5	2	64	ReLu
Conv	5	2	64	ReLu
Conv	5	2	128	ReLu
Conv*	5	2	128	ReLu
FC	-	-	-	ReLu

* 5^{th} layer only exists for global.

Table 2: Global and local discriminator structure, K, S and F are defined as in Table1.

4.4 Loss Function and Training Procedure

We use five types of loss functions:

- $\mathcal{L}^{+}_{MSE} = \frac{1}{n_I} \left\| (\mathbf{I}_{out} - \mathbf{I}_{in}) \odot (\mathbf{1} - \mathbf{M}) \right\|^2$
- $\mathcal{L}^{-}_{MSE} = \frac{1}{n_I} \left\| (\mathbf{I}_{out} - \mathbf{I}_{in}) \odot \mathbf{M} \right\|^2$
- $\mathcal{L}_{MSE} = \mathcal{L}^{+}_{MSE} + \frac{1}{6}\mathcal{L}^{-}_{MSE}$
- $\mathcal{L}_D = -\log D(\mathbf{I}_{out}) - \log\left(1 - D(G(\mathbf{I}_{in}))\right)$
- $\mathcal{L}_G = \mathcal{L}_{MSE} - \frac{1}{2000} \log D(G(\mathbf{I}_{in}))$

where n_I denotes the number of elements in $\mathbf{I}_{in}$. $\mathcal{L}^{+}_{MSE}$ corresponds to the mean squared error (MSE) in the inpainted regions, while $\mathcal{L}^{-}_{MSE}$ corresponds to the MSE in the remaining parts. $\mathcal{L}_D$ is the adversarial loss function for the discriminators and $\mathcal{L}_G$ the generator loss.

Following [4], we train our model in three phases: In the first stage, we train only the generator network on the combined MSE loss $\mathcal{L}_{MSE}$ for $n_1 = 25{,}000$ iterations. In the second stage, we train only our discriminators for $n_2 = 6{,}000$ iterations on $\mathcal{L}_D$. In the third stage, we simultaneously train the generator and discriminators using $\mathcal{L}_D$ and $\mathcal{L}_G$ for $n_3 = 94{,}000$ iterations. The network weights are updated by an Adam optimizer with mini-batch gradient descent using a batch size of 16. Training takes approximately 14 hours on an NVIDIA K80 GPU.

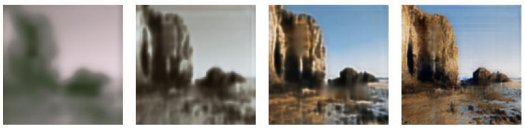


Figure 5: Generator output $G(\mathbf{I}_{in})$ after n number of iterations

5 Results

Figure 5 shows how our network increasingly learns the semantics of an image during the training procedure. At the beginning of training, our model begins to capture low-frequency features like color and brightness. In the later stages of training, the model is able to accurately

Input (a) Full model (b) MSE only (c) No LD (d) No PC Ground truth

Figure 4: (a) Full model with global and local discriminators and partial convolutions. (b) Full model but trained only on the MSE loss $\mathcal{L}_{MSE}$. (c) Model without local discriminators. (d) Full model but replacing all partial with typical convolutional layers.

reproduce higher-frequency features and generates crisp details in the inpainted regions.

As discussed by Yu et al. [10] and Liu et al. [6], there is no clear candidate for a quantitative metric to evaluate inpainting results since many different plausible solutions exist. Currently, the canonical approach is to rely on qualitative comparisons to evaluate the reasonableness of the inpainted content.

Figure 4 shows the inpainted results produced by different specifications of our model. Our model effectively learns the semantics of the image and produces plausible content for the inpainted regions. Previous work has often relied on post-processing to reduce color discrepancies and inconsistencies along the edges of the inpainted regions. For example, Iizuka, Simo-Serra, and Ishikawa [4] use fast marching and Poisson image blending. In contrast, we find our approach does not benefit from similar post-processing.

5.1 Adversarial Loss

Figure 6 reveals the importance of adversarial training to produce crisp details in the inpainted regions. When trained only on the MSE loss $\mathcal{L}_{MSE}$, our model still generates imagery that incorporates smoothly with the surrounding parts. However, the inpainted content becomes significantly blurrier and fails to reproduce the fine details of the image.

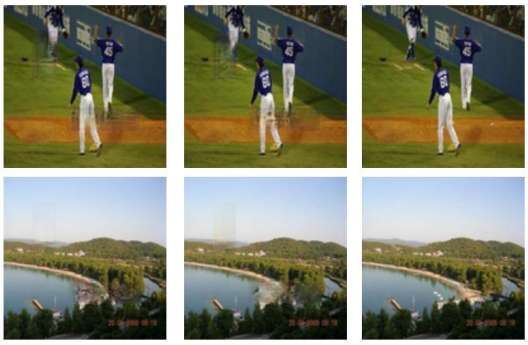
(a) Adversarial (b) MSE only

Figure 6: (a) Full model with adversarial loss. (b) Same model but trained only on the MSE loss $\mathcal{L}_{MSE}$.

5.2 Local Discriminators

Figure 7 compares the results produced by models with and without local discriminators. Although the differences are relatively small and highly dependent on the image at hand, we gen-

erally find that using local discriminators reduces color discrepancies and produces crisper details.



(a) Full model (b) No LD GT

Figure 7: (a) Full model with global and local discriminators. (b) Model without local discriminators.

5.3 Partial Convolutions

We also compare the performance of a network with partial convolutions to a standard convolutional neural network (CNN). We find using standard convolutional layers has an ambiguous effect: It tends to mitigate the checkerboard artifacts like the ones observed in Figure 8 (a). However, in line with the work by [6], it also introduces slight blur and color discrepancies as can be seen in Figure 8 (b).

(a) With PC (b) Without PC

Figure 8: (a) Model with partial convolutions. (b) Model with typical convolutions.

6 Conclusion and Future Work

Our contribution is twofold: First, to the best of our knowledge we are the first to train a GAN with partial convolutions. Secondly, we show that good results can be obtained with a significantly smaller network than with the state-of-the-art approach by Liu et al. [6].

Future work may take two avenues: First, we would have liked our model to be able to inpaint arbitrary shapes. However, due to computational constraints, training our model on a large number of random masks proved infeasible. Secondly, since partial convolutions successively fill the masked parts, the size of the inpainted region is limited by the depth of the network. Future work could therefore try to train a deeper network to inpaint even larger regions.

7 Contributions

All group members have actively participated throughout the whole project. More specifically, Nico and Justin worked on implementing the pre-processing stages of our pipeline. Wilhelm implemented prediction and deployed our AWS instance. Nico implemented partial convolutions and the training stage. All members contributed equally to the final report and poster.

8 Acknowledgements

We would like to thank our project TA Farzan Farnia for his constant support throughout the quarter. We would also like to thank Kian Katanforoosh and Prof. Andrew Ng for their guidance in the lectures.

9 Code

The code for our project is available at `https://github.com/njersch/inpainting-GAN`.

References

[1] Zhou B. et al. *Places: A 10 million Image Database for Scene Recognition*. EEE Transactions on Pattern Analysis and Machine Intelligence. 2017. URL: `http://places2.csail.mit.edu/index.html`.

[2] C. Burlin, Y. Le Calonnec, and L. Duperier. "Deep Image Inpainting". In: *Stanford CS 231* (2018). URL: `http://cs231n.stanford.edu/reports/2017/pdfs/328.pdf`.

[3] U. Demir and G. Unal. "Patch-Based Image Inpainting with Generative Adversarial Networks". In: *Istanbul Techincal University* (2018). URL: `https://arxiv.org/pdf/1803.07422.pdf`.

[4] S. Iizuka, E. Simo-Serra, and H. Ishikawa. "Globally and Locally Consistent Image Completion". In: *ACM Trans. Graph. 36, 4, Article 107* (2017). URL: :%20http://dx.doi.org/10.1145/3072959.3073659.

[5] P. Isola et al. "Image-to-Image Translation with Conditional Adversarial Networks". In: *Berkley AI Research* (2018). URL: https://arxiv.org/pdf/1611.07004.pdf.

[6] G. Liu et al. "Image Inpainting for Irregular Holes Using Partial Convolutions". In: *Nvidia* (Dec. 2018). URL: https://arxiv.org/abs/1804.07723v2.

[7] D. Pathak et al. "Context Encoders: Feature Learning by Inpainting". In: *Berkeley University* (2016). URL: https://arxiv.org/pdf/1604.07379.pdf.

[8] O. Ronneberger, P. Fischer, and T. Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: *Sixth International Conference on Learning Representations* (2015). URL: https://arxiv.org/abs/1505.04597.

[9] Z. Yan et al. "Shift-Net: Image Inpainting via Deep Feature Rearrangemen". In: *Sixth International Conference on Learning Representations* (2018). URL: https://arxiv.org/pdf/1801.09392.pdf.

[10] J. Yu et al. "Generative Image Inpainting with Contextual Attention". In: *University of Illinois at Urbana-Champaign and Adobe Research* (2018). URL: https://arxiv.org/pdf/1801.07892.pdf.