Photo Uncrop

Abhinav Sharma
Columbia University
New York
as5414@columbia.edu

Aashish Kumar Misraa
Columbia University
New York
akm2215@columbia.edu

1. Introduction

The world around us is highly structured, diverse and the ability of humans to make sense of this structure, extrapolate even with some parts missing is really uncanny. In our project we try to explore whether a neural network will be able to do the same. We are building a black box that takes in cropped images as input and generates uncropped images. We choose 40px from each side of an image replaced with black pixels as our input. Context-Encoders [7] dealing with hole-filling and inpainting had holes with relatively small size compared to the amount of pixels our model is trying to predict. Our model generates contextually-relevant pixels and fills the black areas of the cropped image. We use a CNN based encoder to learn a dense encoding of the input images. These dense encoding is then passed to a decoder consisting of layers of transposed convolutions with learn-able filters which is used to decode the encoded features.

For our model to generate contextually relevant pixels it needs to understand the contents of an image and generate a possible hypothesis for the missing border. This is a multi-modal problem and there are multiple ways to fill this hole. Thus to separate this burden on our model we used a weighted reconstruction loss and adversarial loss. Reconstruction loss is just the L2 loss between network's output and the target image. This loss ensures that the generated pixels are structurally similar to the ones present in the current image, but results in a very blurry image. Incorporating the adversarial loss along with reconstruction loss in our model it is able to generate very sharp natural images. The adversarial loss is derived from GANs [3] where the existing network is treated as the generator and a separate discriminator is trained using alternating SGD.

2. Related Work

Photo uncropping can be approached from a wide spectrum of techniques in computer vision. Methods proposed in [10] take the classical approach where photos similar to the input image are learned to retrieve from the dataset. They are then blended/stitched together [8] to give a reasonable uncropped image. Other classical inpainting techniques like [6] or texture synthesis approaches [2, 1] since the missing portion is too large they cannot handle the border filling/tasks quite well using the classical non-semantic methods. In the world of graphics, scene completion [4] is used to fill large holes. But this technique finds it really hard to fill arbitrary holes and generally used to fill gaps when a complete object is removed. Also our model which uses learned metrics is superior to other classical technique like [5], which used nearest-neighbor computation to complete fill gaps.

The first problem is that these methods are not able to produce good results because they are not able to capture a good semantic understanding of the image. Deep neural networks, particularly Convolutional Neural Networks(CNNs) have proven to be excellent in image understanding tasks in the recent years. Therefore, having complex learned features using CNNs is essential to having a good semantic understanding of the image which will help in producing good uncropped output. The second problem is producing the output itself from the learned representations, which can be solved by using generative models. Generative models for images, especially works using Generative Adversarial Networks (GANs) [3] such as [9] have provided very encouraging results. Both feature extraction and image generation approaches described above have been culminated into the idea of context encoders for image inpainting task [7] and we take inspiration from this.

3. Dataset

We used a scene-centric dataset called Places [11] by MIT, with 205 scene categories and 2.5 millions of images with a category label. We used 10 outdoor scene classes (Butte, corn-field, desert, desert-road, farm, field-road, hayfield, mountain-path, pasture, sky). We chose only outdoor scenes because it would help us get more consistent results due to contextual similarity in outdoor images. We have

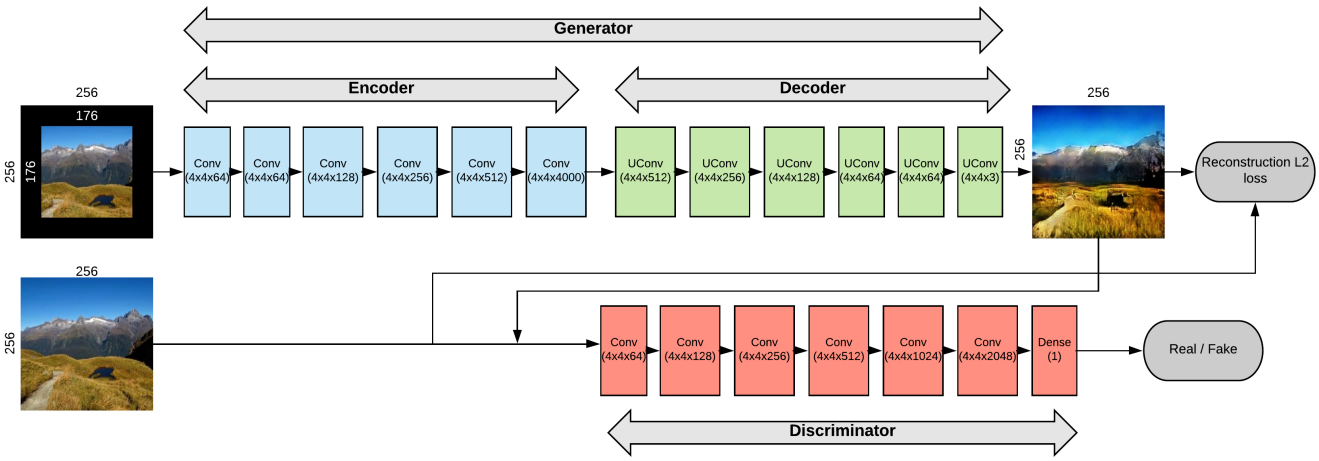

Figure 1. Architecture of the complete model containing the encoder-decoder (generator) and discriminator. Each of the convolution and de-convolution filters are represented by (height x width x num-filters). The numbers along the images denote the size in pixels. We use a fixed crop width of 40 pixels for all input images.

around 50,000 training and 5,000 test images in total, all of size 256x256.

4. Methods

We describe the methods we used in detail below. The overall idea of the deep network and joint loss is inspired from [7] where they use it for solving the inpainting problem, i.e., fill out holes in images.

4.1. Encoder-decoder pipeline

At the heart of our model is a simple encoder-decoder network. The detailed architecture of the model is shown in Fig 1. The encoder consists of a series of Convolutional layers and the decoder consists of a series of De-Convolutional layers. We use batch-normalization after each layer in both the encoder and decoder. For activation, we use leaky-ReLU for encoder and ReLU for decoder. The encoder takes an input image with the border pixels blacked out and produces a latent feature representation of that image. The decoder takes this latent representation as input and generates a full size image.

4.2. GAN pipeline

Generative Adversarial Networks (GANs) [3] have seen significant success in image generation [9]. Therefore, we use an adversarial model to get more realistic predictions. Specifically, we use the encoder-decoder network to double up as the generator, G for the GAN and use a separate discriminator network D. During training, G and D plays a two-player game where G tries to fool the network with generated images and D has to tell whether the image its provided is real or fake. The exact composition of D is shown in Fig 1.

4.3. Loss function

4.3.1 Reconstruction Loss

We train the encoder-decoder network by regressing to the ground-truth of the input image. Specifically, we use reconstruction loss which is simply the normalized L2 distance between the generated output and the ground truth. This loss helps us get a rough outline of the image but often fails to generate high frequency details. This is because L2 loss tends to predict the mean of the distribution as it helps to minimize the mean pixel error. We experiment with two variants of this loss which is described below:

1. **Masked reconstruction loss**: Here we use a binary mask M which is of the same shape as the input image but with 1s in the blacked out pixels and 0s elsewhere. We multiply this mask to the difference image so that the reconstruction loss is computed only over the blacked out pixels. The formula is given below ($\hat{y}$ is the decoder output, y is the ground-truth, $*$ is element wise product):

$$L_{rec} = ||M * (\hat{y} - y)||_2^2$$

   Since we are only interested in generating values for the blacked out pixels in the border, we can use this mask to focus on those pixels and ignore the rest.

2. **Unmasked reconstruction loss**: Here we compute the reconstruction loss for all pixels in the output image and ignore the mask used before. The network therefore generates the full image in this case including that for the input patch. The formula is given below:

$$L_{rec} = ||\hat{y} - y||_2^2$$

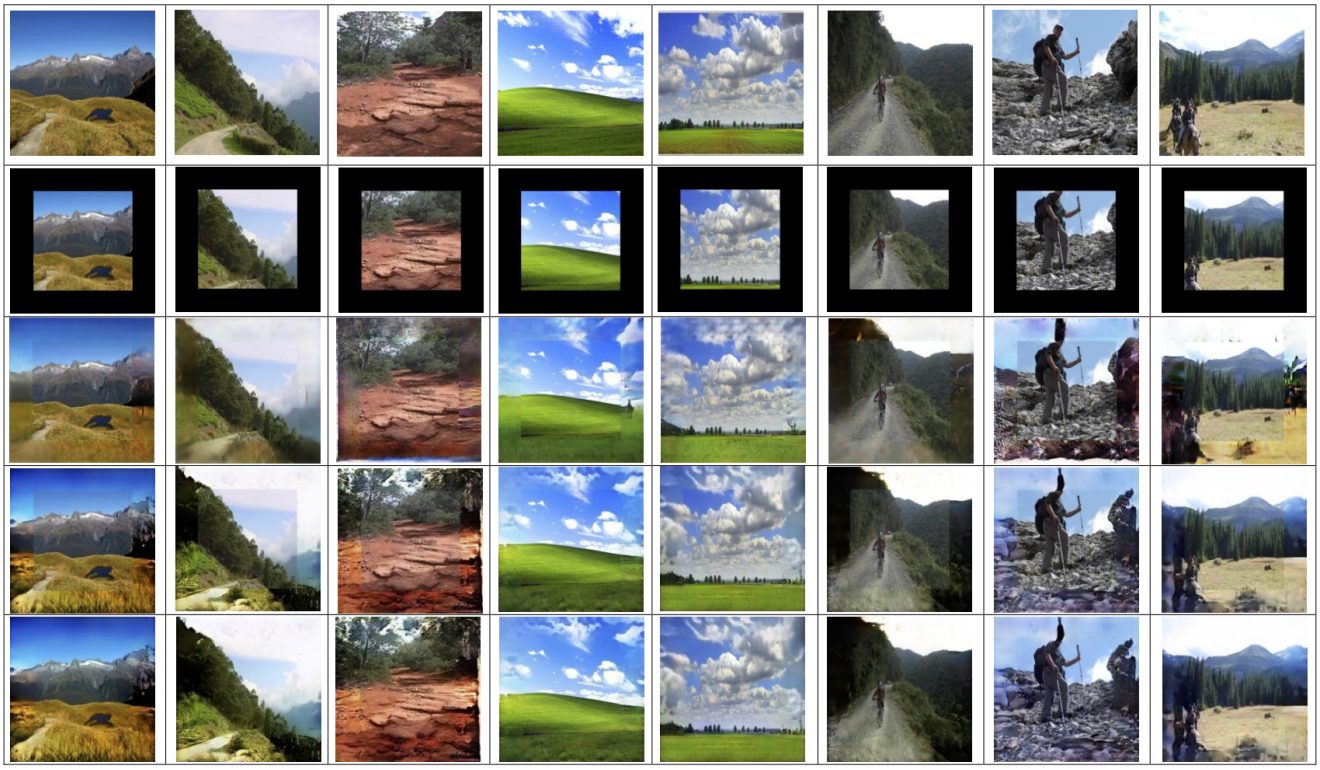

Figure 2. Test results from the model. Each row contains the following starting from the first row: Ground-truth, cropped input, overlayed output from model using mask loss, overlayed output from model using unmask loss, post-processed output from model using unmask loss. All the images (other than the fourth one from the left which is grabbed from the internet) is from the test set. Last two columns depict instances of inaccurate output.

4.3.2 Adversarial Loss

As mentioned in section 4.2, we model the encoder-decoder network as the generator G for the GAN and train a separate discriminator D. The adversarial loss used here is given below:

$$L_{adv} = \max_{D} E_{y \in Y}[\log(D(y)) + \log(1 - D(\hat{y})]$$

Here D is the discriminator, y is the ground-truth (real) image and $\hat{y}$ is the generator (decoder in our case) output. The generator and discriminator are jointly optimized using alternating SGD.

4.3.3 Joint Loss

We combine the reconstruction and adversarial loss as follows:

$$L = \lambda_{rec} L_{rec} + \lambda_{adv} L_{adv}$$

We set $\lambda_{rec} = 0.9$ and $\lambda_{adv} = 0.1$. This joint loss encourages the network to have an average reconstruction (from L_{rec}) while also produce high detail artifacts in it (from L_{adv}). We use the joint loss function for all our experiments.

4.4. Implementation details

4.4.1 Training

We train the model for 50 epochs on the training set using ADAM optimizer for both the generator and discriminator. The pipeline is implemented using Tensorflow, the code for which is available here [https://github.com/abhinavs95/photo-uncrop]. We train two separate instances of the network, one with the masked reconstruction loss and other with the unmasked loss as described in section 4.3.1.

4.4.2 Testing

To test the network on unseen images, we pick images from the held-out test set, black out the border pixels (of width 40) and input it to the generator. We then overlay the input image on top of the generated output in such a way that the blacked out border pixels is replaced by the generated border. We call this the overlayed output.

4.5. Post-processing

The overlayed output images have sharp and noticeable edges at locations where the input patch ends and the gen-

Output type	PSNR (in dB)	L2 loss (in %)	L1 loss (in %)
Cropped input	8.11	16.52	25.64
Raw output	16.60	2.53	11.70
Overlayed (Mask)	17.77	2.05	7.95
Overlayed (Unmask)	**18.63**	**1.60**	**6.78**
Post-processed	17.00	2.32	11.02

Table 1. Mean Peak Signal to Noise Ratio (PSNR), L1 and L2 loss for test images.

erated image begins. Also there are color and brightness differences in most cases between these two parts which reduce the realism of the resulting image. To alleviate this we use seamless cloning, a technique presented in [8]. Using this technique, the source input patch is seamlessly blended onto the destination raw output from the model. It removes the sharp edges and also matches the color and brightness of the source image to that of the destination image. This technique can only be applied to output from unmasked loss model as it requires the full generated image. For the model with masked loss, the outputs have empty region in the center and therefore cannot be used effectively for seamless cloning.

5. Results

5.1. Visual Analysis

Evaluating the quality of synthesized images is well known to be a difficult task, as conventional quantitative metrics might struggle to capture visual realism. Therefore we use visual analysis of the results as our primary evaluation method. Some result samples from our test set is shown in Fig 2. The model is able to encapsulate good contextual information from the input image as seen from the output of the first six columns. Particularly its able to complete roads in the images in columns one, two and six, produce high quality textures for the rocks in column three and generate realistic cloud patterns in columns four and five. The post-processing further improves the realism of the output. The last two columns denote examples of inaccurate output. The network has not seen a lot of images containing people or animals during training which is why that result is expected.

5.2. Quantitative Evaluation

We also evaluate our results with some quantitative metrics which can quickly give us a reasonable estimate of the performance of the model. The metrics are defined below:

1. **Mean L1 and L2 loss:** The L1 (mean absolute error) and L2 (mean squared error) loss is computed between the outputs and the ground-truth for all test images. The errors are then averaged for all the test images and converted to percentage. The values are in table 1.

2. **Peak Signal to Noise Ratio (PSNR):** The peak signal to noise ratio between the predicted image and the ground-truth is also used as an evaluation metric. First, the mean squared error between the two images is computed which is used to calculate the PSNR as follows: $PSNR = 10\log_{10}(R^2/MSE)$. Here, R is the maximum fluctuation in the input image data type. The values are shown in table 1.

As expected, for all the output types the metrics are significantly better than those of the cropped input. Also the metrics are better for all types of output than that of the raw generated output leaving out the cropped input. Interestingly, the metrics are consistently better for the overlayed output for the network with unmasked reconstruction loss network compared to that of its masked counterpart. This is probably because when generating the full image using the unmasked loss, the network is able to retain context from the input patch better than when it is just generating the border. Also the metrics for the post-processed output slightly worse than its overlayed counterpart. This is expected as the seamless cloning performed during post-processing changes the pixel intensities of the source image so that it matches the destination image and make the result more believable. But in doing so the result deviates from the ground-truth which directly impacts the metrics.

6. Work division

Abhinav: Model preparation and training, generating test output, postprocessing, evaluation metrics, presentation and report preparation.
Aashish: Data preparation and preprocessing, class selection, evaluation metrics, presentation and report preparation.

References

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

[2] A. A. Efros and T. K. Leung. Texture synthesis by non-parametric sampling. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 2, pages 1033–1038 vol.2, Sept 1999.

[3] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014.

[4] 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.

[5] A. Oliva and A. Torralba. Building the gist of a scene: the role of global image features in recognition, 2006.

[6] S. Osher, M. Burger, D. Goldfarb, J. Xu, and W. Yin. An iterative regularization method for total variation-based image restoration. *Multiscale Modeling & Simulation*, 4(2):460–489, 2005.

[7] D. Pathak, P. Krähenbühl, J. Donahue, T. Darrell, and A. A. Efros. Context encoders: Feature learning by inpainting. *CoRR*, abs/1604.07379, 2016.

[8] P. Pérez, M. Gangnet, and A. Blake. Poisson image editing. *ACM Trans. Graph.*, 22(3):313–318, July 2003.

[9] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *CoRR*, abs/1511.06434, 2015.

[10] Q. Shan, B. Curless, Y. Furukawa, C. Hernandez, and S. Seitz. Photo uncrop. *ECCV*, 8694, 2014.

[11] 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.