Semantic Image Inpainting with Perceptual and Contextual Losses

Raymond Yeh[*] **Chen Chen**[*] **Teck Yian Lim,**
Mark Hasegawa-Johnson **Minh N. Do**
Dept. of Electrical and Computer Engineering
University of Illinois at Urbana-Champaign
{yeh17, cchen156, tlim11, jhasegaw, minhdo}@illinois.edu

Abstract

In this paper, we propose a novel method for image inpainting based on a Deep Convolutional Generative Adversarial Network (DCGAN). We define a loss function consisting of two parts: (1) a contextual loss that preserves similarity between the input corrupted image and the recovered image, and (2) a perceptual loss that ensures a perceptually realistic output image. Given a corrupted image with missing values, we use back-propagation on this loss to map the corrupted image to a smaller latent space. The mapped vector is then passed through the generative model to predict the missing content. The proposed framework is evaluated on the CelebA and SVHN datasets for two challenging inpainting tasks with random 80% corruption and large blocky corruption. Experiments show that our method can successfully predict semantic information in the missing region and achieve pixel-level photorealism, which is impossible by almost all existing methods.

1 Introduction

The goal of inpainting is to reconstruct the missing or damaged portions of an image. It has numerous applications such as restoration of damaged paintings or image editing [3]. Inpainting for general images is a challenging problem as it is ill-posed. Sufficient prior information is required in the reconstruction to achieve a meaningful and visually believable result.

Existing methods are often based on either local or non-local information to recover the image. Local methods rely on the prior information that exist in the input image. For example, the holes in texture images can be filled by finding the nearest patches from the same image [6]. Total variation approaches take into account of the smoothness of a natural image, which enables small holes and spurious noises to be removed [22]. Certain subsets of images could also contain special properties, such as being planar [12] or having a low rank structure [11]; reconstruction results can be greatly improved by taking into consideration these prior knowledge. These local methods can be efficient and achieve high quality results when the above assumptions are satisfied. While they are good at seamlessly filling holes with local pixels/patches, they are unable to predict the semantic information in the missing region, especially when the desired content is not contained in the corrupted image. For example, when the center of a human face image is missing, we should expect to recover the nose or eyes in the region instead of smooth skin, which is a major challenge by these local methods.

In order to solve the more general inpainting problem, non-local methods use external training images to predict the missing pixel values. To fill an image hole, Hays and Efros proposed to cut and paste a semantically similar patch from a huge database [10]. To inpaint a scene, images of the same scenes are retrieved from the Internet [24]. The result is obtained by registering and blending the retrieved images. When the scene is not contained in the training set or can be hardly retrieved, it

[*]Authors contributed equally to this work

is difficult to fill parts of the scene by cutting and pasting. Unlike the previous hand-crafted matching and editing, sparse coding is proposed to recover the images from a learned dictionary [18]. Recently, inpainting by convolutional neural networks has shown promising results[19, 25].

Instead of filling the holes with the background texture, we are interested in the more general and more difficult task called semantic inpainting [19]. It aims to predict the detailed contents in a large region based on the context of surrounding pixels. This task is easy for human beings while extremely difficult for the computer, when the example is not included in the training set. Previous methods such as sparse coding are based on a linear representation, which is often not powerful enough to produce smooth non-linear distortion of a basis vector as necessary to match uncorrupted image pixels. The closest work to ours is the context encoders by Pathak et al. [19]. Given a mask, a network is trained to predict the content in the mask. The recovered image is semantically correct, but still looks blurry and unrealistic compared with the natural images.

We consider semantic inpainting as a constrained image generation problem. The generated content needs to be well aligned with surrounding pixels and semantically realistic based on the context. Therefore, this is much more difficult than unconstrained image generation. We propose a new framework to fill holes in images using back-propagation on a pretrained image generative model, such as the Deep Convolutional Generative Adversarial Network (DCGAN) [21]. With the corrupted image, we find the "closest" corresponding vector in the smaller latent space, which is then used to reconstruct the whole image. To find the "closest" vector in the latent space we apply back-propagation on the designed loss function. We define a contextual loss to limit deviation between the recovered image and the corrupted image. The perceptual loss penalizes recovered images that are perceptually unrealistic. The proposed framework can be used to fill arbitrary holes in the image without retraining the network. We evaluate our method on the CelebA and SVHN datasets with large portions of block and random missing pixels. The results demonstrate that our method can generate much more realistic images and significantly higher quality images than the existing methods on these challenging tasks.

2 Related Work: Image Generation

Generative Adversarial Networks (GAN) generate meaningful image content through the competition between a generative model, which captures the data distribution, and a discriminative model, which penalizes flaws in the generative model. The generative model and discriminative model iteratively improve each other during the training.

After the networks are trained, one can input a random vector to generate an image with similar distribution to the training set. Encouraging results have been obtained by GAN [4, 9, 21]. For the inpainting task, GAN will not work because it will produce a completely unrelated image with high probability unless constrained by information about the corrupted image.

The context encoders [19] can be also viewed as a generative model. While the input of GAN [4, 9, 21] is often a random noise vector, the input of the context encoders is the uncorrupted pixels. A network is trained to predict the missing content. However, the predicted content tends to have very limited resolution compared the surrounding pixels. As a result, the recovered image looks unrealistic. Instead of training an end-to-end network to predict the image content, we use back-propagation on the pretrained GAN to find the reconstruction from the closest image representation.

There are some existing methods using back-propagation to generate and modify images. For example, back-propagation has been used on texture synthesis and style transfer [7, 8, 14]. Google's DeepDream uses back-propagation to create dreamlike images [1]. These works are used to generate desired image effects by designing loss functions. Additionally, back-propagation has also been used to visualize and understand the learned features in a trained network, by "inverting" the network through updating the gradient at the input layer[5, 15, 17, 23].

3 Method

In this section we will review the generative adversarial network (GAN) of Goodfellow *et al.* [9]. We will then introduce our method, which utilizes the image representation from GAN for the image inpainting task.

3.1 Generative Adversarial Networks

GAN is a framework for training generative parametric models, and has been shown to generate high quality natural images [4, 9, 21]. This framework trains two networks, a generative model, G, and a discriminative model D. G maps a random vector $\mathbf{z}$, sampled from a prior distribution $p_{\mathbf{Z}}$, to the image space. D maps an input image to a likelihood. The purpose of G is to generate realistic images, while D plays an adversarial role to discriminate between the image generated from G, and the image sampled from data distribution p_{data}.

The networks are trained by optimizing the loss function (1):

$$\min_G \max_D V(G, D) = \mathbb{E}_{\mathbf{h}\sim p_{data}(\mathbf{h})}[\log(D(\mathbf{h}))] + \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{Z}}(\mathbf{z})}[\log(1 - D(G(\mathbf{z}))] \tag{1}$$

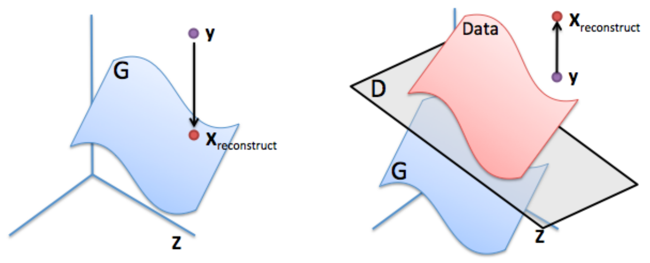
where $\mathbf{h}$ is the sample from the p_{data} distribution; $\mathbf{z}$ is randomly generated and lies in some latent space.

Simply stated, G is updated to fool D into misclassifying the generated sample, $G(\mathbf{z})$, while D tries not to be fooled. In this work, both G and D are deep convolutional neural networks and are trained with an alternating gradient descent algorithm in [9]. After convergence, D is able to reject images that are too fake, and G can produce high quality images similar to the training distribution from the latent space.

3.2 Inpainting with Pretrained GAN

Our method for image inpainting utilizes the G and D networks from GAN, pretrained with uncorrupted data, to reconstruct images. After training, G is able to embed the images from p_{data} onto some non-linear manifold of $\mathbf{z}$. We speculate that if G is efficient in its representation then an image that is not from p_{data} (e.g. corrupted data) should not lie on the learned manifold. To do reconstruction, we hope to recover the "closest" image on the manifold to the corrupted image, as illustrated in figure 1 (a).

Denote the corrupted image as $\mathbf{y}$. Naively, one might consider using D to update $\mathbf{y}$ by maximizing $D(\mathbf{y})$, like the back-propagation in DeepDream. However, the corrupted data is not drawn from those distributions. Therefore, $\mathbf{y}$ is neither on the p_{data} manifold, nor on the G manifold. Therefore, maximizing $D(\mathbf{y})$ does not lead the desired reconstruction. A possible scenario is illustrated in Fig. 1 (b), where the image converges to neither of the manifolds. We consider using both D and G for reconstruction. To quantify the "closest" mapping from $\mathbf{y}$ to the reconstruction, we define a function consisting of contextual loss $\mathcal{L}_{contextual}$ and a perceptual loss $\mathcal{L}_{perceptual}$.



(a) (b)

Figure 1: (a) Illustration of the non-linear manifold and the task of inpainting, (b) Illustration of a failure example by maximizing $D(\mathbf{y})$. The image does not necessarily converge to any point on the desired manifold.

3.2.1 Contextual Loss

We need to incorporate the information from the uncorrupted portion of the given image. The contextual loss is used to measure the context similarity between the reconstructed image and the uncorrupted portion, which is defined as

$$\mathcal{L}_{contextual}(\mathbf{z}) = \|\mathbf{M} \odot G(\mathbf{z}) - \mathbf{M} \odot \mathbf{y}\|_1 \tag{2}$$

where $\mathbf{M}$ denotes the binary mask of the uncorruption and $\odot$ denotes the element-wise product operation. The corrupted portion, i.e., $(1 - \mathbf{M}) \odot \mathbf{y}$ is not used in the loss. The choice of ℓ_1-norm is empirical. From our experiments, images recovered with ℓ_1-norm loss tend to be sharper and with higher quality compared to ones reconstructed with ℓ_2-norm.

3.2.2 Perceptual Loss

The perceptual loss encourages the reconstructed image to be similar to the samples drawn from the training set.

This is achieved by updating $\mathbf{z}$ to fool D. As a result, D will predict $G(\mathbf{z})$ to be from the data with a high probability. We use the same loss for fooling D as in GAN:

$$\mathcal{L}_{perceptual}(\mathbf{z}) = \log(1 - D(G(\mathbf{z}))) \tag{3}$$

Without $L_{perceptual}$, the mapping from $\mathbf{x}$ to $\mathbf{z}$ can converge to a perceptually implausible result. Some reconstructed images tend to be unrealistic. We illustrate this by showing the unstable examples where we optimized with and without $\mathcal{L}_{perceptual}$ in Fig. 2.

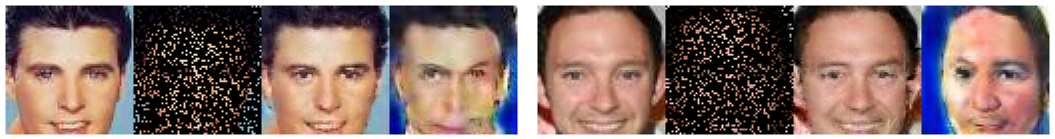


Figure 2: Inpainting with different losses. For each example, **Column 1:** Original images from the dataset. **Column 2:** Corrputed image. **Column 3:** Inpainting by our method with contextual and perceptual losses. **Column 4:** Recovered image without perceptual loss.

3.2.3 Inpainting

With the defined perceptual and contextual losses, the corrupted image can be mapped to the closest $\mathbf{z}$ in the latent representation space. $\mathbf{z}$ is updated using back-propagation with the total loss:

$$\hat{\mathbf{z}} = \arg\min_{\mathbf{z}} (\mathcal{L}_{contextual}(\mathbf{z}) + \lambda \mathcal{L}_{perceptual}(\mathbf{z})) \tag{4}$$

where λ is a weighting parameter. In practice, λ has to be relatively small to constrain the recovered image with the input pixels. After finding $\hat{\mathbf{z}}$, the inpainting can be obtained by:

$$\mathbf{x}_{reconstructed} = \mathbf{M} \odot \mathbf{y} + (1 - \mathbf{M}) \odot G(\hat{\mathbf{z}}) \tag{5}$$

In many cases, we found the predicted pixels may not exactly preserve the same intensities of the surrounding pixels, although the content is correct and well aligned. Poisson blending [20] is used to reconstruct our final results:

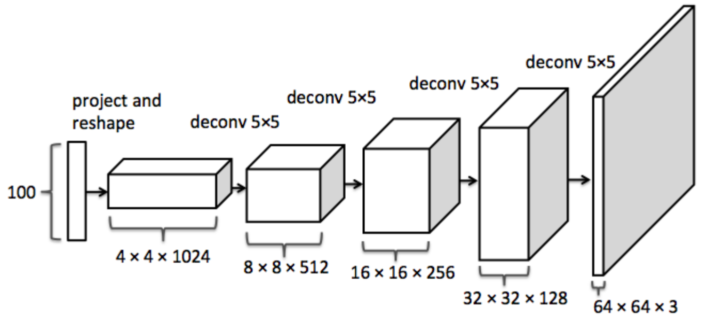
$$\begin{aligned} \mathbf{x}_{reconstructed} = \mathbf{M} \odot \mathbf{y} + \arg\min_{\mathbf{x}} \Bigg\{ & \sum_{\mathbf{M}_i=0, j \in N_i \& \mathbf{M}_j=0} [(\mathbf{x}_i - \mathbf{x}_j) - (G(\hat{\mathbf{z}})_i - G(\hat{\mathbf{z}})_j)]^2 \\ & + \sum_{\mathbf{M}_i=0, j \in N_i \& \mathbf{M}_j=1} [(\mathbf{x}_i - \mathbf{y}_j) - (G(\hat{\mathbf{z}})_i - G(\hat{\mathbf{z}})_j)]^2 \Bigg\} \end{aligned} \tag{6}$$

where i, j denote the i^{th} and j^{th} element; N_i denotes the neighboring of pixel i. The minimization problem contains two quadratic terms, which can be solved in closed-form.

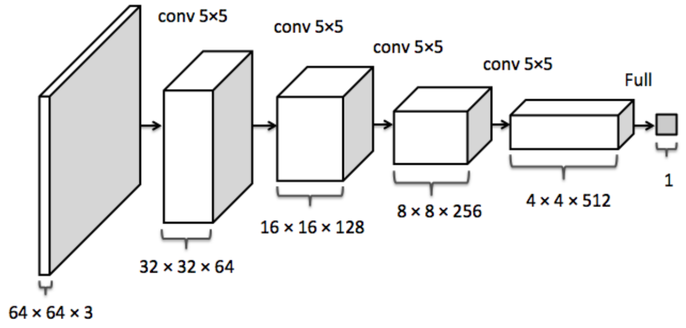
4 Experiments

4.1 Implementation Details

We used the DCGAN model architecture from Radford *et al.* [21] in this work, but our framework is orthogonal to the specific GAN model that one can choose. The generative model, G, is structured as follows: The input is a 100 dimensional random noise vector, drawn from uniform distribution between $[-1, 1]$, followed by an 8192 fully connected layer reshaped into dimensions of $4 \times 4 \times 512$. Each of the following layers are deconvolutional layers, where the numbers of channels are halved, and image dimension doubles from the previous layers. The output layer is of dimension $64 \times 64 \times 3$, the size of the image space.



(a)



(b)

Figure 3: (a) Generator architecture. (b) Discriminator architecture.

On the other hand, the discriminating model, D, is structured essentially in reverse. The input layer is an image of dimension $64 \times 64 \times 3$, followed by a series of convolution layers where the image dimension is halved, and the number of channels is doubled from the previous layer, and the output layer is a two class softmax. The architectures of D and G are shown in Fig. 3.

For training the DCGAN model, we follow the training procedure in [21] and use Adam [13] for optimization. We tune the learning-rate and Adam's β_1 term. For image inpainting, we again use Adam for optimization and tune the hyperparameters on a validation set; the hyperparameters include learning-rate, Adam's β_1 term, λ ratio of the loss function, and the number of iterations. We use the hyperparameters that produces the best qualitative result on the validation set for testing.

4.2 CelebA Dataset

The CelebFaces Attributes Dataset (CelebA) contains $202,599$ face images with coarse alignment [26]. For testing, we remove 2000 images from the dataset before training. We crop the center 64×64 of the image for both training and testing, which contains face.

We evaluate our method on the CelebA dataset, with two types of corruptions: 80% of pixels randomly deleted, or a large missing block in the central. Both tasks are very challenging. For the former task, 80% of pixels have be to recovered from the very limited given information. For the latter one, the recovered region must contain semantically correct content, i.e., eyes, nose, and eyebrows on human faces. And more importantly, all the content should be aligned perfectly to the surrounding face features. Our results are shown in Fig. 4, which demonstrate that our method can successfully predict the missing content with high quality.

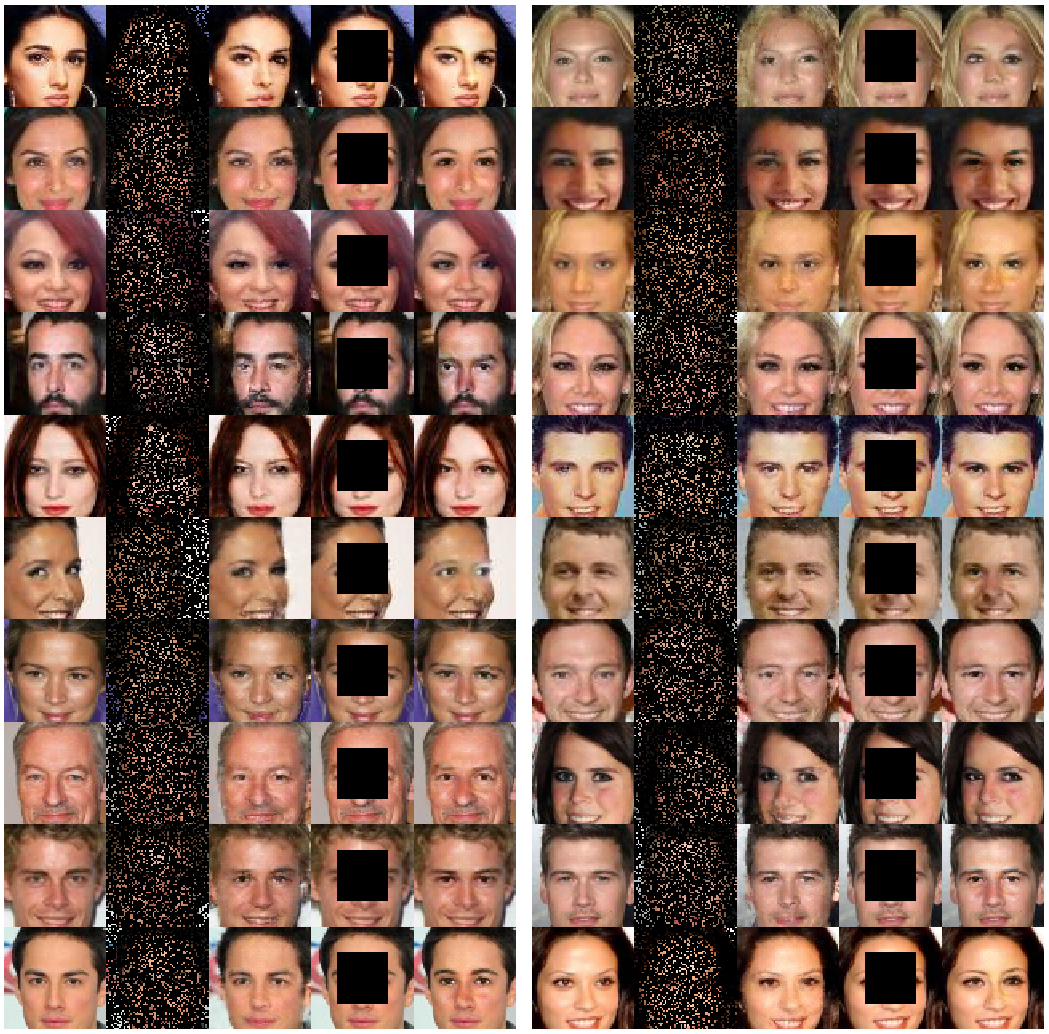


Figure 4: For each example, **Column 1:** Original images from the dataset. **Column 2:** Images with 80% random missing pixels. **Column 3:** Inpainting of column 2 by our method. **Column 4:** Image with large central region missing. **Column 5**: Inpainting of column 4 by our method.

We compare our method with local methods using Total Variation (TV) regularization [2] and low rank minimization [11, 16]. The results of random 80% missing pixels are shown in Fig. 5. Due to the high missing ratio, existing methods cannot recover enough image details, resulting in very blurry and noisy images. For the block hole-filling task, they directly fail to recover the image, as the missing content cannot be found in the input image. These results are not shown here as one can easily imagine the failures.

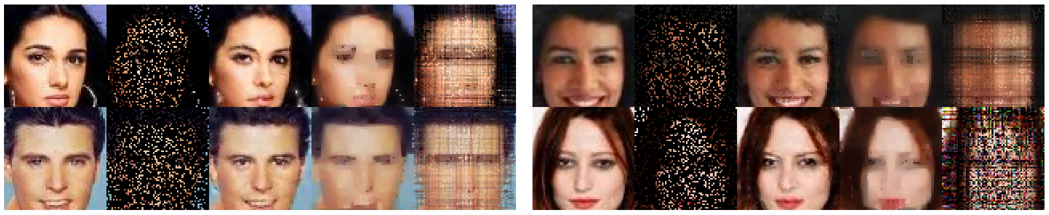


Figure 5: Comparisons with local inpainting methods on examples with random 80% missing. For each example, **Column 1:** Original images from the dataset. **Column 2:** The corrupted image. **Column 3:** Inpainting by our method. **Column 4:** Inpainting by TV minimization. **Column 5**: Inpainting by low rank minimization.

For the block hole filling task, we compare the results by nearest neighbor filling, which is widely used in the cut-and-paste framework [10, 24]. Unlike editing whole objects [10], the problem here is more difficult. For example, when the central region is missing, part of the hair, nose, eyes may still remain in the image. Therefore, cut-and-paste based method may easily fail due to the misalignment. Examples are shown in Fig. 6, where the misalignment of skin texture, eyebrows, eyes and hair can be clearly observed by using the nearest patches in Euclidean distance. Such misalignment is often because the nearest patch is from a different subject, with a different imaging angle, which can not be removed easily by registration. Instead, our results are obtained automatically without any registration.

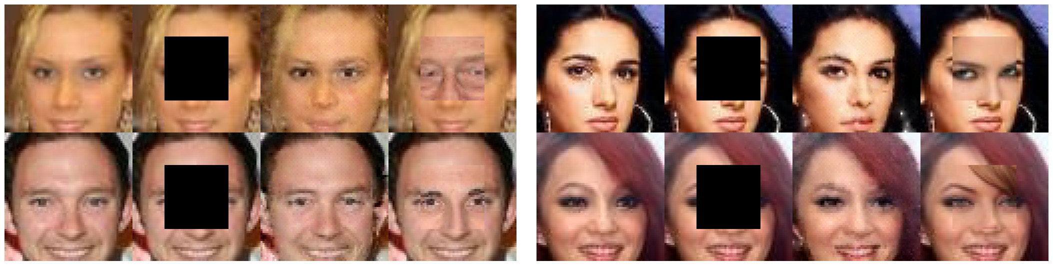


Figure 6: Central hole filling. For each example, **Column 1:** Original images from the dataset. **Column 2:** The corrupted image. **Column 3:** Inpainting by our method. **Column 4:** Inpainting by the nearest patch in the dataset.

4.3 SVHN Dataset

We validate the robustness of the proposed method on another dataset. The Street View House Numbers (SVHN) dataset contains a total of 99,289 RGB images of cropped house numbers. We used the training and testing partition as provided. We resize the images to 64×64 to fit our DCGAN model architecture. Fig. 7 shows the results. Although these numbers are not aligned, our method can predict realistic content in the missing region.

5 Conclusion and Discussion

In this paper, we have proposed a new method to predict missing content in an image based on the surrounding values. Compared with the existing methods based on local image priors or patches, the proposed method learns the distribution of training data, and can therefore predict meaningful contents unseen in the corrupted images. The recovered images by our method have sharp edges and look very realistic. Experimental results have demonstrated its superior performance on challenging image inpainting examples.

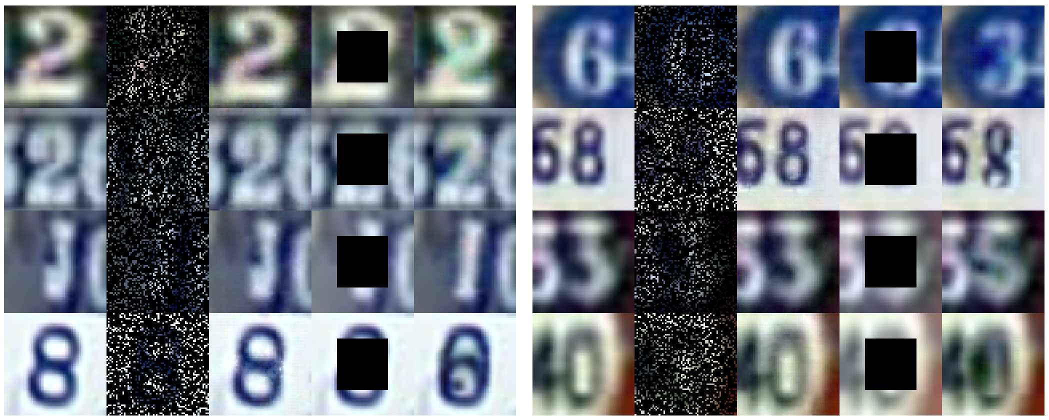


Figure 7: For each example, **Column 1:** Original images from the dataset. **Column 2:** Images with 80% random missing pixels. **Column 3:** Inpainting of column 2 by our method. **Column 4:** Image with large central region missing. **Column 5**: Inpainting of column 4 by our method.

While the results are promising, the limitation of our method is also obvious. It cannot solve the ambiguity caused by the corruption. For example, the number six can be recovered as three or the number three can be recovered as five in Fig. 7. Indeed, its prediction performance strongly relies on the generative model and the training procedure. When the training dataset cannot effectively represent the test image, it will fail to complete the correct content. Some failure examples are shown in Fig. 8, as the hat, half face and sun glasses appear less in the dataset. The current GAN model in this work is large enough for faces, but is too small to represent complex scenes. In future work, we will investigate more powerful generative models to complete complex scenes.

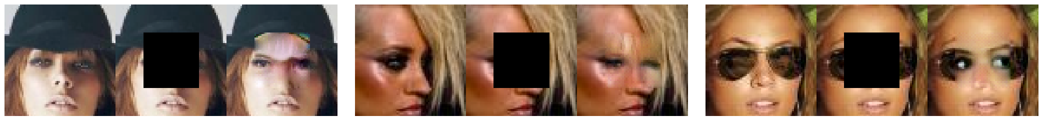


Figure 8: Some failure examples. For each example, **Column 1:** Original images from the dataset. **Column 2:** Images with central block missing. **Column 3:** Inpainting of column 2 by our method.

References

[1] Inceptionism: Going deeper into neural networks. `http://googleresearch.blogspot.com/2015/06/inceptionism-going-deeper-into-neural.html`, 2016. Accessed: 2016-05-10.

[2] Manya V Afonso, José M Bioucas-Dias, and Mário AT Figueiredo. An augmented lagrangian approach to the constrained optimization formulation of imaging inverse problems. *IEEE Transactions on Image Processing*, 20(3):681–695, 2011.

[3] Marcelo Bertalmio, Guillermo Sapiro, Vincent Caselles, and Coloma Ballester. Image inpainting. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 417–424. ACM Press/Addison-Wesley Publishing Co., 2000.

[4] Emily L Denton, Soumith Chintala, Rob Fergus, et al. Deep generative image models using a laplacian pyramid of adversarial networks. In *Advances in Neural Information Processing Systems*, pages 1486–1494, 2015.

[5] Alexey Dosovitskiy and Thomas Brox. Inverting visual representations with convolutional networks. *arXiv preprint arXiv:1506.02753*, 2015.

[6] Alexei A Efros and Thomas K Leung. Texture synthesis by non-parametric sampling. In *IEEE International Conference on Computer Vision*, volume 2, pages 1033–1038, 1999.

[7] Leon Gatys, Alexander S Ecker, and Matthias Bethge. Texture synthesis using convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 262–270, 2015.

[8] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. A neural algorithm of artistic style. *arXiv preprint arXiv:1508.06576*, 2015.

[9] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014.

[10] James Hays and Alexei A Efros. Scene completion using millions of photographs. *ACM Transactions on Graphics*, 26(3):4, 2007.

[11] Yao Hu, Debing Zhang, Jieping Ye, Xuelong Li, and Xiaofei He. Fast and accurate matrix completion via truncated nuclear norm regularization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(9):2117–2130, 2013.

[12] Jia-Bin Huang, Sing Bing Kang, Narendra Ahuja, and Johannes Kopf. Image completion using planar structure guidance. *ACM Transactions on Graphics*, 33(4):129, 2014.

[13] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. 2015.

[14] Chuan Li and Michael Wand. Combining markov random fields and convolutional neural networks for image synthesis. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[15] Alexander Linden et al. Inversion of multilayer nets. In *International Joint Conference on Neural Networks*, pages 425–430, 1989.

[16] Canyi Lu, Jinhui Tang, Shuicheng Yan, and Zhouchen Lin. Generalized nonconvex nonsmooth low-rank minimization. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 4130–4137, 2014.

[17] Aravindh Mahendran and Andrea Vedaldi. Understanding deep image representations by inverting them. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 5188–5196, 2015.

[18] Julien Mairal, Michael Elad, and Guillermo Sapiro. Sparse representation for color image restoration. *IEEE Transactions on Image Processing*, 17(1):53–69, 2008.

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

[20] Patrick Pérez, Michel Gangnet, and Andrew Blake. Poisson image editing. In *ACM Transactions on Graphics*, volume 22, pages 313–318, 2003.

[21] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[22] Jianhong Shen and Tony F Chan. Mathematical models for local nontexture inpaintings. *SIAM Journal on Applied Mathematics*, 62(3):1019–1043, 2002.

[23] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

[24] Oliver Whyte, Josef Sivic, and Andrew Zisserman. Get out of my picture! internet-based inpainting. In *British Machine Vision Conference*, 2009.

[25] Junyuan Xie, Linli Xu, and Enhong Chen. Image denoising and inpainting with deep neural networks. In *Advances in Neural Information Processing Systems*, pages 341–349, 2012.

[26] Liu Ziwei, Luo Ping, Wang Xiaogang, , and Tang Xiaoou. Deep learning face attributes in the wild. In *International Conference on Computer Vision*, December 2015.