

Semantic Image Inpainting with Deep Generative Models

Raymond A. Yeh*, Chen Chen*, Teck Yian Lim,
Alexander G. Schwing, Mark Hasegawa-Johnson, Minh N. Do
University of Illinois at Urbana-Champaign

{yeh17, cchen156, tlim11, aschwing, jhasegaw, minhdo}@illinois.edu

Abstract

Semantic image inpainting is a challenging task where large missing regions have to be filled based on the available visual data. Existing methods which extract information from only a single image generally produce unsatisfactory results due to the lack of high level context. In this paper, we propose a novel method for semantic image inpainting, which generates the missing content by conditioning on the available data. Given a trained generative model, we search for the closest encoding of the corrupted image in the latent image manifold using our context and prior losses. This encoding is then passed through the generative model to infer the missing content. In our method, inference is possible irrespective of how the missing content is structured, while the state-of-the-art learning based method requires specific information about the holes in the training phase. Experiments on three datasets show that our method successfully predicts information in large missing regions and achieves pixel-level photorealism, significantly outperforming the state-of-the-art methods.

1. Introduction

Semantic inpainting [30] refers to the task of inferring arbitrary large missing regions in images based on image semantics. Since prediction of high-level context is required, this task is significantly more difficult than classical inpainting or image completion which is often more concerned with correcting spurious data corruption or removing entire objects. Numerous applications such as restoration of damaged paintings or image editing [3] benefit from accurate semantic inpainting methods if large regions are missing. However, inpainting becomes increasingly more difficult if large regions are missing or if scenes are complex.

Classical inpainting methods are often based on either local or non-local information to recover the image. Most existing methods are designed for single image inpainting.

* Authors contributed equally.

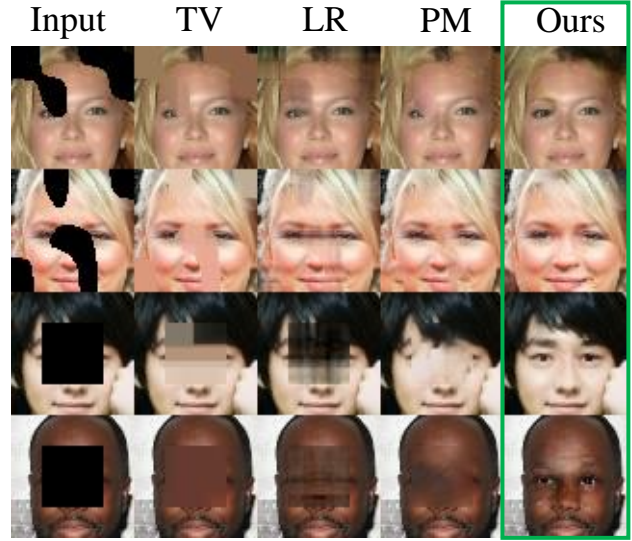


Figure 1. Semantic inpainting results by TV, LR, PM and our method. Holes are marked by black color.

Hence they are based on the information available in the input image, and exploit image priors to address the ill-posedness. For example, total variation (TV) based approaches [34, 1] take into account the smoothness property of natural images, which is useful to fill small missing regions or remove spurious noise. Holes in textured images can be filled by finding a similar texture from the same image [6]. Prior knowledge, such as statistics of patch offsets [11], planarity [13] or low rank (LR) [12] can greatly improve the result as well. PatchMatch (PM) [2] searches for similar patches in the available part of the image and quickly became one of the most successful inpainting methods due to its high quality and efficiency. However, all single image inpainting methods require appropriate information to be contained in the input image, *e.g.*, similar pixels, structures, or patches. This assumption is hard to satisfy, if the missing region is large and possibly of arbitrary shape. Consequently, in this case, these methods are unable to recover the missing information. Fig. 1 shows some challenging examples with large missing regions, where local

methods fail to recover the nose and eyes.

In order to address inpainting in the case of large missing regions, non-local methods try to predict the missing pixels using external data. Hays and Efros [10] proposed to cut and paste a semantically similar patch from a huge database. Internet based retrieval can be used to replace a target region of a scene [37]. Both methods require exact matching from the database or Internet, and fail easily when the test scene is significantly different from any database image. Unlike previous hand-crafted matching and editing, learning based methods have shown promising results [27, 38, 33, 22]. After an image dictionary or a neural network is learned, the training set is no longer required for inference. Oftentimes, these learning-based methods are designed for small holes or for removing small text in the image.

Instead of filling small holes in the image, we are interested in the more difficult task of semantic inpainting [30]. It aims to predict the detailed content of a large region based on the context of surrounding pixels. A seminal approach for semantic inpainting, and closest to our work is the Context Encoder (CE) by Pathak *et al.* [30]. Given a mask indicating missing regions, a neural network is trained to encode the context information and predict the unavailable content. However, the CE only takes advantage of the structure of holes during training but not during inference. Hence it results in blurry or unrealistic images especially when missing regions have arbitrary shapes.

In this paper, we propose a novel method for semantic image inpainting. We consider semantic inpainting as a constrained image generation problem and take advantage of the recent advances in generative modeling. After a deep generative model, *i.e.*, in our case an adversarial network [9, 32], is trained, we search for an encoding of the corrupted image that is “closest” to the image in the latent space. The encoding is then used to reconstruct the image using the generator. We define “closest” by a weighted context loss to condition on the corrupted image, and a prior loss to penalizes unrealistic images. Compared to the CE, one of the major advantages of our method is that it does not require the masks for training and can be applied for arbitrarily structured missing regions during inference. We evaluate our method on three datasets: CelebA [23], SVHN [29] and Stanford Cars [17], with different forms of missing regions. Results demonstrate that on challenging semantic inpainting tasks our method can obtain much more realistic images than the state of the art techniques.

2. Related Work

A large body of literature exists for image inpainting, and due to space limitations we are unable to discuss all of it in detail. Seminal work in that direction includes the aforementioned works and references therein. Since our method is based on generative models and deep neural nets, we will



Figure 2. Images generated by a VAE and a DCGAN. First row: samples from a VAE. Second row: samples from a DCGAN.

review the technically related learning based work in the following.

Generative Adversarial Networks (GANs) are a framework for training generative parametric models, and have been shown to produce high quality images [9, 4, 32]. This framework trains two networks, a generator, G , and a discriminator D . G maps a random vector \mathbf{z} , sampled from a prior distribution $p_{\mathbf{z}}$, to the image space while D maps an input image to a likelihood. The purpose of G is to generate realistic images, while D plays an adversarial role, discriminating between the image generated from G , and the real image sampled from the data distribution p_{data} .

The G and D networks are trained by optimizing the loss function:

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

where \mathbf{h} is the sample from the p_{data} distribution; \mathbf{z} is a random encoding on the latent space.

With some user interaction, GANs have been applied in interactive image editing [40]. However, GANs can not be directly applied to the inpainting task, because they produce an entirely unrelated image with high probability, unless constrained by the provided corrupted image.

Autoencoders and Variational Autoencoders (VAEs) [16] have become a popular approach to learning of complex distributions in an unsupervised setting. A variety of VAE flavors exist, *e.g.*, extensions to attribute-based image editing tasks [39]. Compared to GANs, VAEs tend to generate overly smooth images, which is not preferred for inpainting tasks. Fig. 2 shows some examples generated by a VAE and a Deep Convolutional GAN (DCGAN) [32]. Note that the DCGAN generates much sharper images. Jointly training VAEs with an adversarial loss prevents the smoothness [18], but may lead to artifacts.

The Context Encoder (CE) [30] can be also viewed as an autoencoder conditioned on the corrupted images.

It produces impressive reconstruction results when the structure of holes is fixed during both training and inference, *e.g.*, fixed in the center, but is less effective for arbitrarily structured regions.

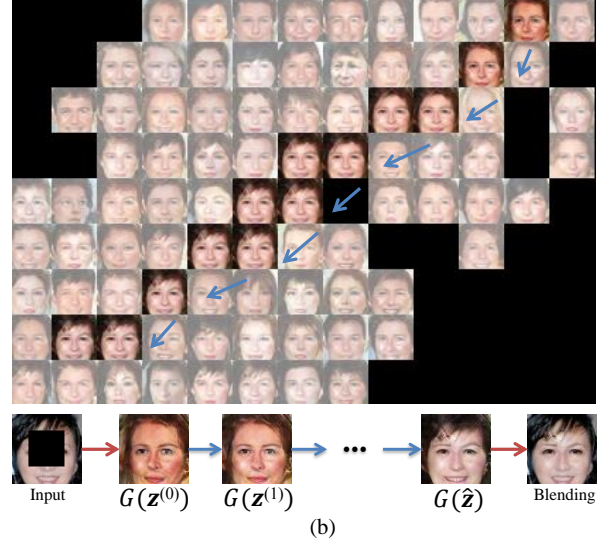
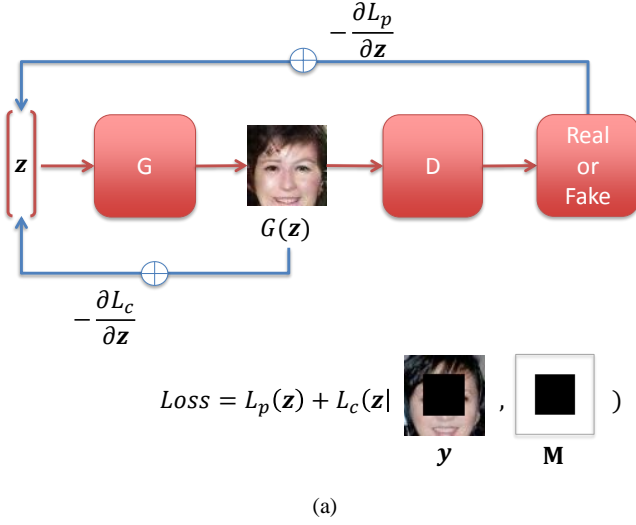


Figure 3. The proposed framework for inpainting. (a) Given a GAN model trained on real images, we iteratively update \mathbf{z} to find the closest mapping on the latent image manifold, based on the designed loss functions. (b) Manifold traversing when iteratively updating \mathbf{z} using back-propagation. $\mathbf{z}^{(0)}$ is random initialized; $\mathbf{z}^{(k)}$ denotes the result in k -th iteration; and $\hat{\mathbf{z}}$ denotes the final solution.

Back-propagation to the input data is employed in our approach to find the encoding which is close to the provided but corrupted image. In earlier work, back-propagation to augment data has been used for texture synthesis and style transfer [8, 7, 20]. Google’s DeepDream uses back-propagation to create dreamlike images [28]. 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 [26, 5, 35, 21]. Similar to our method, all these back-propagation based methods require specifically designed loss functions for the particular tasks.

3. Semantic Inpainting by Constrained Image Generation

To fill large missing regions in images, our method for image inpainting utilizes the generator G and the discriminator D , both of which are trained with uncorrupted data. After training, the generator G is able to take a point \mathbf{z} drawn from $p_{\mathbf{z}}$ and generate an image mimicking samples from p_{data} . We hypothesize 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 encoding manifold, \mathbf{z} . Therefore, we aim to recover the encoding $\hat{\mathbf{z}}$ “closest” to the corrupted image while being constrained to the manifold, as illustrated in Fig. 3; we visualize the latent manifold, using t-SNE [25] on the 2-dimensional space, and the intermediate results in the optimization steps of finding $\hat{\mathbf{z}}$. After $\hat{\mathbf{z}}$ is obtained, we can generate the missing content by using the trained generative model G .

More specifically, we formulate the process of finding $\hat{\mathbf{z}}$ as an optimization problem. Let \mathbf{y} be the corrupted image and \mathbf{M} be the binary mask with size equal to the image, to indicate the missing parts. An example of \mathbf{y} and \mathbf{M} is shown in Fig. 3 (a).

Using this notation we define the “closest” encoding $\hat{\mathbf{z}}$ via:

$$\hat{\mathbf{z}} = \arg \min_{\mathbf{z}} \{ \mathcal{L}_c(\mathbf{z}|\mathbf{y}, \mathbf{M}) + \mathcal{L}_p(\mathbf{z}) \}, \quad (2)$$

where \mathcal{L}_c denotes the context loss, which constrains the generated image given the input corrupted image \mathbf{y} and the hole mask \mathbf{M} ; \mathcal{L}_p denotes the prior loss, which penalizes unrealistic images. The details of the proposed loss function will be discussed in the following sections.

Besides the proposed method, one may also consider using D to update \mathbf{y} by maximizing $D(\mathbf{y})$, similar to back-propagation in DeepDream [28] or neural style transfer [8]. However, the corrupted data \mathbf{y} is neither drawn from a real image distribution nor the generated image distribution. Therefore, maximizing $D(\mathbf{y})$ may lead to a solution that is far away from the latent image manifold, which may hence lead to results with poor quality.

3.1. Importance Weighted Context Loss

To fill large missing regions, our method takes advantage of the remaining available data. We designed the context loss to capture such information. A convenient choice for the context loss is simply the ℓ_2 norm between the generated sample $G(\mathbf{z})$ and the uncorrupted portion of the input image \mathbf{y} . However, such a loss treats each pixel equally, which is not desired. Consider the case where the center

block is missing: a large portion of the loss will be from pixel locations that are far away from the hole, such as the background behind the face. Therefore, in order to find the correct encoding, we should pay significantly more attention to the missing region that is close to the hole.

To achieve this goal, we propose a context loss with the hypothesis that the importance of an uncorrupted pixel is positively correlated with the number of corrupted pixels surrounding it. A pixel that is very far away from any holes plays very little role in the inpainting process. We capture this intuition with the importance weighting term, \mathbf{W} ,

$$\mathbf{W}_i = \begin{cases} \sum_{j \in N(i)} \frac{(1 - \mathbf{M}_j)}{|N(i)|} & \text{if } \mathbf{M}_i \neq 0 \\ 0 & \text{if } \mathbf{M}_i = 0 \end{cases}, \quad (3)$$

where i is the pixel index, \mathbf{W}_i denotes the importance weight at pixel location i , $N(i)$ refers to the set of neighbors of pixel i in a local window, and $|N(i)|$ denotes the cardinality of $N(i)$. We use a window size of 7 in all experiments.

Empirically, we also found the ℓ_1 -norm to perform slightly better than the ℓ_2 -norm in our framework. Taking it all together, we define the contextual loss to be a weighted ℓ_1 -norm difference between the recovered image and the uncorrupted portion, defined as follows,

$$\mathcal{L}_c(\mathbf{z}|\mathbf{y}, \mathbf{M}) = \|\mathbf{W} \odot (G(\mathbf{z}) - \mathbf{y})\|_1. \quad (4)$$

Here, \odot denotes the element-wise multiplication.

3.2. Prior Loss

The prior loss refers to a class of penalties based on high-level image feature representations instead of pixel-wise differences. In this work, the prior loss encourages the recovered image to be similar to the samples drawn from the training set. Our prior loss is different from the one defined in [14] which uses features from pre-trained neural networks.

Our prior loss penalizes unrealistic images. Recall that in GANs, the discriminator, D , is trained to differentiate generated images from real images. Therefore, we choose the prior loss to be identical to the GAN loss for training the discriminator D , *i.e.*,

$$\mathcal{L}_p(\mathbf{z}) = \lambda \log(1 - D(G(\mathbf{z}))). \quad (5)$$

Here, λ is a parameter to balance between the two losses. \mathbf{z} is updated to fool D and make the corresponding generated image more realistic. Without \mathcal{L}_p , the mapping from \mathbf{y} to \mathbf{z} may converge to a perceptually implausible result. We illustrate this by showing the unstable examples where we optimized with and without \mathcal{L}_p in Fig. 4.

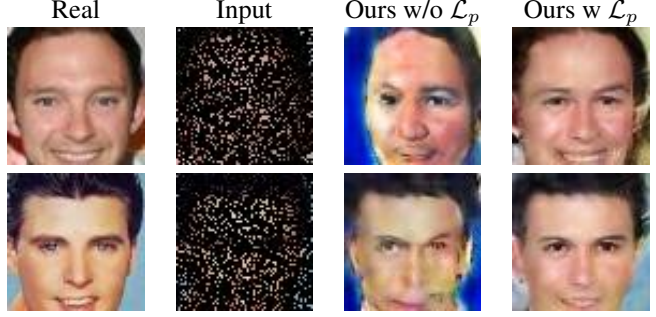


Figure 4. Inpainting with and without the prior loss.

3.3. Inpainting

With the defined prior and context losses at hand, the corrupted image can be mapped to the closest \mathbf{z} in the latent representation space, which we denote $\hat{\mathbf{z}}$. \mathbf{z} is randomly initialized and updated using back-propagation on the total loss given in Eq. (2). Fig. 3 (b) shows for one example that \mathbf{z} is approaching the desired solution on the latent image manifold.

After generating $G(\hat{\mathbf{z}})$, the inpainting result can be easily obtained by overlaying the uncorrupted pixels from the input. However, we found that the predicted pixels may not exactly preserve the same intensities of the surrounding pixels, although the content is correct and well aligned. Poisson blending [31] is used to reconstruct our final results. The key idea is to keep the gradients of $G(\hat{\mathbf{z}})$ to preserve image details while shifting the color to match the color in the input image \mathbf{y} . Our final solution, $\hat{\mathbf{x}}$, can be obtained by:

$$\begin{aligned} \hat{\mathbf{x}} &= \arg \min_{\mathbf{x}} \|\nabla \mathbf{x} - \nabla G(\hat{\mathbf{z}})\|_2^2, \\ \text{s.t. } \mathbf{x}_i &= \mathbf{y}_i \text{ for } \mathbf{M}_i = 1, \end{aligned} \quad (6)$$

where ∇ is the gradient operator. The minimization problem contains a quadratic term, which has a unique solution [31]. Fig. 5 shows two examples where we can find visible seams without blending.

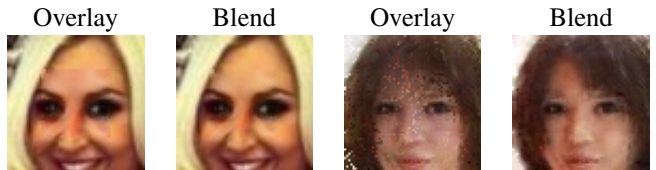


Figure 5. Inpainting with and without blending.

3.4. Implementation Details

In general, our contribution is orthogonal to specific GAN architectures and our method can take advantage of any generative model G . We used the DCGAN model architecture from Radford *et al.* [32] in the experiments. The

generative model, G , takes a random 100 dimensional vector drawn from a uniform distribution between $[-1, 1]$ and generates a $64 \times 64 \times 3$ image. The discriminator model, D , is structured essentially in reverse order. 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 half, and the number of channels is double the size of the previous layer, and the output layer is a two class softmax.

For training the DCGAN model, we follow the training procedure in [32] and use Adam [15] for optimization. We choose $\lambda = 0.003$ in all our experiments. We also perform data augmentation of random horizontal flipping on the training images. In the inpainting stage, we need to find \hat{z} in the latent space using back-propagation. We use Adam for optimization and restrict z to $[-1, 1]$ in each iteration, which we observe to produce more stable results. We terminate the back-propagation after 1500 iterations. We use the identical setting for all testing datasets and masks.

4. Experiments

In the following section we evaluate results qualitatively and quantitatively, more comparisons are provided in the supplementary material.

4.1. Datasets and Masks

We evaluate our method on three dataset: the CelebFaces Attributes Dataset (CelebA) [23], the Street View House Numbers (SVHN) [29] and the Stanford Cars Dataset [17].

The CelebA contains 202,599 face images with coarse alignment [23]. We remove approximately 2000 images from the dataset for testing. The images are cropped at the center to 64×64 , which contain faces with various view-points and expressions.

The SVHN dataset contains a total of 99,289 RGB images of cropped house numbers. The images are resized to 64×64 to fit the DCGAN model architecture. We used the provided training and testing split. The numbers in the images are not aligned and have different backgrounds.

The Stanford Cars dataset contains 16,185 images of 196 classes of cars. Similar as the CelebA dataset, we do not use any attributes or labels for both training and testing. The cars are cropped based on the provided bounding boxes and resized to 64×64 . As before, we use the provided training and test set partition.

We test four different shapes of masks: 1) central block masks; 2) random pattern masks [30] in Fig. 1, with approximately 25% missing; 3) 80% missing complete random masks; 4) half missing masks (randomly horizontal or vertical).

4.2. Visual Comparisons

Comparisons with TV and LR inpainting. We compare our method with local inpainting methods. As we already

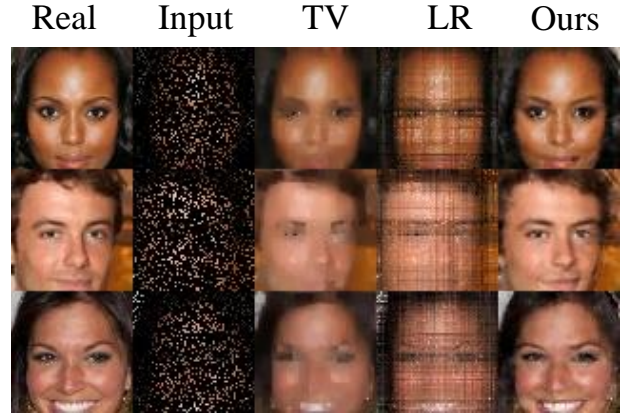


Figure 6. Comparisons with local inpainting methods TV and LR inpainting on examples with random 80% missing.

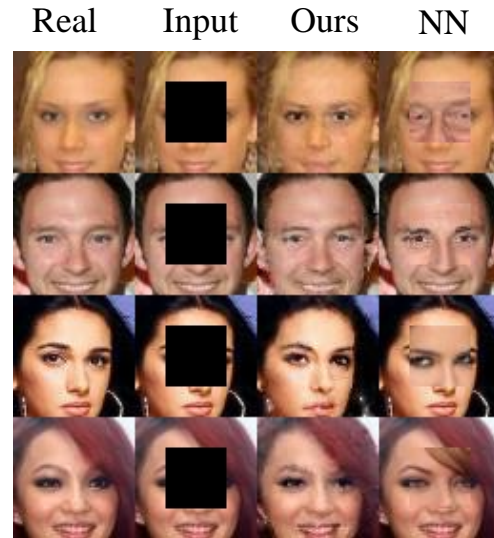


Figure 7. Comparisons with nearest patch retrieval.

showed in Fig. 1, local methods generally fail for large missing regions. We compare our method with TV inpainting [1] and LR inpainting [24, 12] on images with small random holes. The test images and results are shown in Fig. 6. Due to a large number of missing points, TV and LR based methods cannot recover enough image details, resulting in very blurry and noisy images. PM [2] cannot be applied to this case due to insufficient available patches.

Comparisons with NN inpainting. Next we compare our method with nearest neighbor (NN) filling from the training dataset, which is a key component in retrieval based methods [10, 37]. Examples are shown in Fig. 7, where the misalignment of skin texture, eyebrows, eyes and hair can be clearly observed by using the nearest patches in Euclidean distance. Although people can use different features for retrieval, the inherit misalignment problem cannot be easily solved [30]. Instead, our results are obtained automatically without any registration.

Comparisons with CE. In the remainder, we compare our

Table 1. The PSNR values (dB) on the test sets. Left/right results are by CE[30]/ours.

Masks/Dataset	CelebA	SVHN	Cars
Center	21.3 /19.4	22.3 /19.0	14.1 /13.5
pattern	19.2 /17.4	22.3 /19.8	14.0/ 14.1
random	20.6/ 22.8	24.1/ 33.0	16.1/ 18.9
half	15.5 /13.7	19.1 /14.6	12.6 /11.1

result with those obtained from the CE [30], the state-of-the-art method for semantic inpainting. It is important to note that the masks is required to train the CE. For a fair comparison, we use all the test masks in the training phase for the CE. However, there are infinite shapes and missing ratios for the inpainting task. To achieve satisfactory results one may need to re-train the CE. In contrast, our method can be applied to arbitrary masks without re-training the network, which is according to our opinion a huge advantage when considering inpainting applications.

Figs. 8 and 9 show the results on the CelebA dataset with four types of masks. Despite some small artifacts, the CE performs best with central masks. This is due to the fact that the hole is always fixed during both training and testing in this case, and the CE can easily learn to fill the hole from the context. However, random missing data, is much more difficult for the CE to learn. In addition, the CE does not use the mask for inference but pre-fill the hole with the mean color. It may mistakenly treat some uncorrupted pixels with similar color as unknown. We could observe that the CE has more artifacts and blurry results when the hole is at random positions. In many cases, our results are as realistic as the real images. Results on SVHN and car datasets are shown in Figs. 10 and 11, and our method generally produces visually more appealing results than the CE since the images are sharper and contain fewer artifacts.

4.3. Quantitative Comparisons

It is important to note that semantic inpainting is not trying to reconstruct the ground-truth image. The goal is to fill the hole with realistic content. Even the ground-truth image is one of many possibilities. However, readers may be interested in quantitative results, often reported by classical inpainting approaches. Following previous work, we compare the PSNR values of our results and those by the CE. The real images from the dataset are used as groundtruth reference. Table 1 provides the results on the three datasets. The CE has higher PSNR values in most cases except for the random masks, as they are trained to minimize the mean square error. Similar results are obtained using SSIM [36] instead of PSNR. These results conflict with the aforementioned visual comparisons, where our results generally yield to better perceptual quality.

We investigate this claim by carefully investigating the errors of the results. Fig. 12 shows the results of one exam-



Figure 8. Comparisons with CE on the CelebA dataset.

ple and the corresponding error images. Judging from the figure, our result looks artifact-free and very realistic, while the result obtained from the CE has visible artifacts in the reconstructed region. However, the PSNR value of CE is 1.73dB higher than ours. The error image shows that our result has large errors in hair area, because we generate a hairstyle which is different from the real image. This indi-



Figure 9. Comparisons with CE on the CelebA dataset.



Figure 10. Comparisons with CE on the SVHN dataset.

cates that quantitative result do not represent well the real performance of different methods when the ground-truth is not unique. Similar observations can be found in recent super-resolution works [14, 19], where better visual results corresponds to lower PSNR values.

For random holes, both methods achieve much higher PSNR, even with 80% missing pixels. In this case, our

method outperforms the CE. This is because uncorrupted pixels are spread across the entire image, and the flexibility of the reconstruction is strongly restricted; therefore PSNR is more meaningful in this setting which is more similar to the one considered in classical inpainting works.



Figure 11. Comparisons with CE on the car dataset.

4.4. Discussion

While the results are promising, the limitation of our method is also obvious. Indeed, its prediction performance strongly relies on the generative model and the training procedure. Some failure examples are shown in Fig. 13, where our method cannot find the correct \hat{z} in the latent image

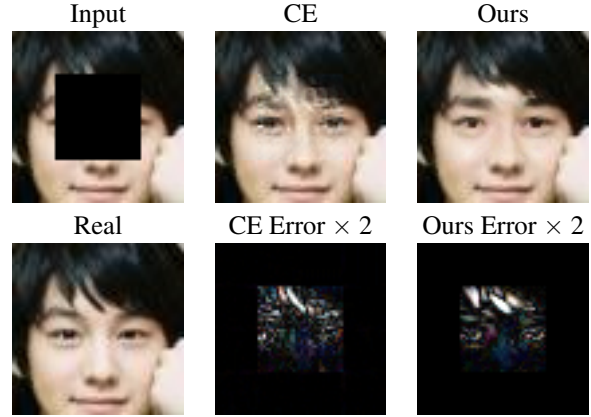


Figure 12. The error images for one example. The PSNR for context encoder and ours are 24.71 dB and 22.98 dB, respectively. The errors are amplified for display purpose.

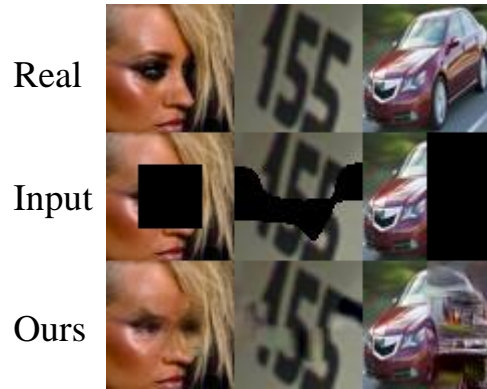


Figure 13. Some failure examples.

manifold. The current GAN model in this paper works well for relatively simple structures like faces, but is too small to represent complex scenes in the world. Conveniently, stronger generative models, improve our method in a straight-forward way.

5. Conclusion

In this paper, we proposed a novel method for semantic inpainting. Compared to existing methods based on local image priors or patches, the proposed method learns the representation of training data, and can therefore predict meaningful content for corrupted images. Compared to CE, our method often obtains images with sharper edges which look much more realistic. Experimental results demonstrated its superior performance on challenging image inpainting examples.

Acknowledgments: This work is supported in part by IBM-ILLINOIS Center for Cognitive Computing Systems Research (C3SR) - a research collaboration as part of the IBM Cognitive Horizons Network. This work is supported by NVIDIA Corporation with the donation of a GPU.

References

- [1] M. V. Afonso, J. M. Bioucas-Dias, and M. A. Figueiredo. An augmented lagrangian approach to the constrained optimization formulation of imaging inverse problems. *IEEE TIP*, 2011.
- [2] C. Barnes, E. Shechtman, A. Finkelstein, and D. Goldman. PatchMatch: a randomized correspondence algorithm for structural image editing. *ACM TOG*, 2009.
- [3] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester. Image inpainting. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, 2000.
- [4] E. L. Denton, S. Chintala, R. Fergus, et al. Deep generative image models using a Laplacian pyramid of adversarial networks. In *NIPS*, 2015.
- [5] A. Dosovitskiy and T. Brox. Inverting visual representations with convolutional networks. *arXiv preprint arXiv:1506.02753*, 2015.
- [6] A. A. Efros and T. K. Leung. Texture synthesis by non-parametric sampling. In *ICCV*, 1999.
- [7] L. Gatys, A. S. Ecker, and M. Bethge. Texture synthesis using convolutional neural networks. In *NIPS*, 2015.
- [8] L. A. Gatys, A. S. Ecker, and M. Bethge. Image style transfer using convolutional neural networks. In *CVPR*, 2016.
- [9] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *NIPS*, 2014.
- [10] J. Hays and A. A. Efros. Scene completion using millions of photographs. *ACM TOG*, 2007.
- [11] K. He and J. Sun. Statistics of patch offsets for image completion. In *ECCV*, 2012.
- [12] Y. Hu, D. Zhang, J. Ye, X. Li, and X. He. Fast and accurate matrix completion via truncated nuclear norm regularization. *IEEE PAMI*, 2013.
- [13] J.-B. Huang, S. B. Kang, N. Ahuja, and J. Kopf. Image completion using planar structure guidance. *ACM TOG*, 2014.
- [14] J. Johnson, A. Alahi, and L. Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *ECCV*, 2016.
- [15] D. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.
- [16] D. Kingma and M. Welling. Auto-encoding variational bayes. In *ICLR*, 2014.
- [17] J. Krause, M. Stark, J. Deng, and L. Fei-Fei. 3D object representations for fine-grained categorization. In *ICCV Workshops*, 2013.
- [18] A. B. L. Larsen, S. K. S nderby, and O. Winther. Autoencoding beyond pixels using a learned similarity metric. In *ICML*, 2016.
- [19] C. Ledig, L. Theis, F. Husz r, J. Caballero, A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi. Photo-realistic single image super-resolution using a generative adversarial network. *arXiv preprint arXiv:1609.04802*, 2016.
- [20] C. Li and M. Wand. Combining Markov random fields and convolutional neural networks for image synthesis. In *CVPR*, 2016.
- [21] A. Linden et al. Inversion of multilayer nets. In *IJCNN*, 1989.
- [22] S. Liu, J. Pan, and M.-H. Yang. Learning recursive filters for low-level vision via a hybrid neural network. In *ECCV*, 2016.
- [23] Z. Liu, P. Luo, X. Wang, and X. Tang. Deep learning face attributes in the wild. In *ICCV*, 2015.
- [24] C. Lu, J. Tang, S. Yan, and Z. Lin. Generalized nonconvex nonsmooth low-rank minimization. In *CVPR*, 2014.
- [25] L. v. d. Maaten and G. Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 2008.
- [26] A. Mahendran and A. Vedaldi. Understanding deep image representations by inverting them. In *CVPR*, 2015.
- [27] J. Mairal, M. Elad, and G. Sapiro. Sparse representation for color image restoration. *IEEE TIP*, 2008.
- [28] A. Mordvintsev, C. Olah, and M. Tyka. Inceptionism: Going deeper into neural networks. *Google Research Blog. Retrieved June*, 20, 2015.
- [29] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshops*, 2011.
- [30] D. Pathak, P. Kr henb hl, J. Donahue, T. Darrell, and A. Efros. Context encoders: Feature learning by inpainting. 2016.
- [31] P. P rez, M. Gangnet, and A. Blake. Poisson image editing. In *ACM TOG*, 2003.
- [32] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.
- [33] J. S. Ren, L. Xu, Q. Yan, and W. Sun. Shepard convolutional neural networks. In *NIPS*, 2015.
- [34] J. Shen and T. F. Chan. Mathematical models for local non-texture inpaintings. *SIAM Journal on Applied Mathematics*, 2002.
- [35] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.
- [36] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE TIP*, 2004.
- [37] O. Whyte, J. Sivic, and A. Zisserman. Get out of my picture! internet-based inpainting. In *BMVC*, 2009.
- [38] J. Xie, L. Xu, and E. Chen. Image denoising and inpainting with deep neural networks. In *NIPS*, 2012.
- [39] X. Yan, J. Yang, K. Sohn, and H. Lee. Attribute2Image: Conditional image generation from visual attributes. *arXiv preprint arXiv:1512.00570*, 2015.
- [40] J.-Y. Zhu, P. Kr henb hl, E. Shechtman, and A. A. Efros. Generative visual manipulation on the natural image manifold. In *ECCV*, 2016.

6. Supplementary Material



Figure 14. Additional results on the celebA dataset.

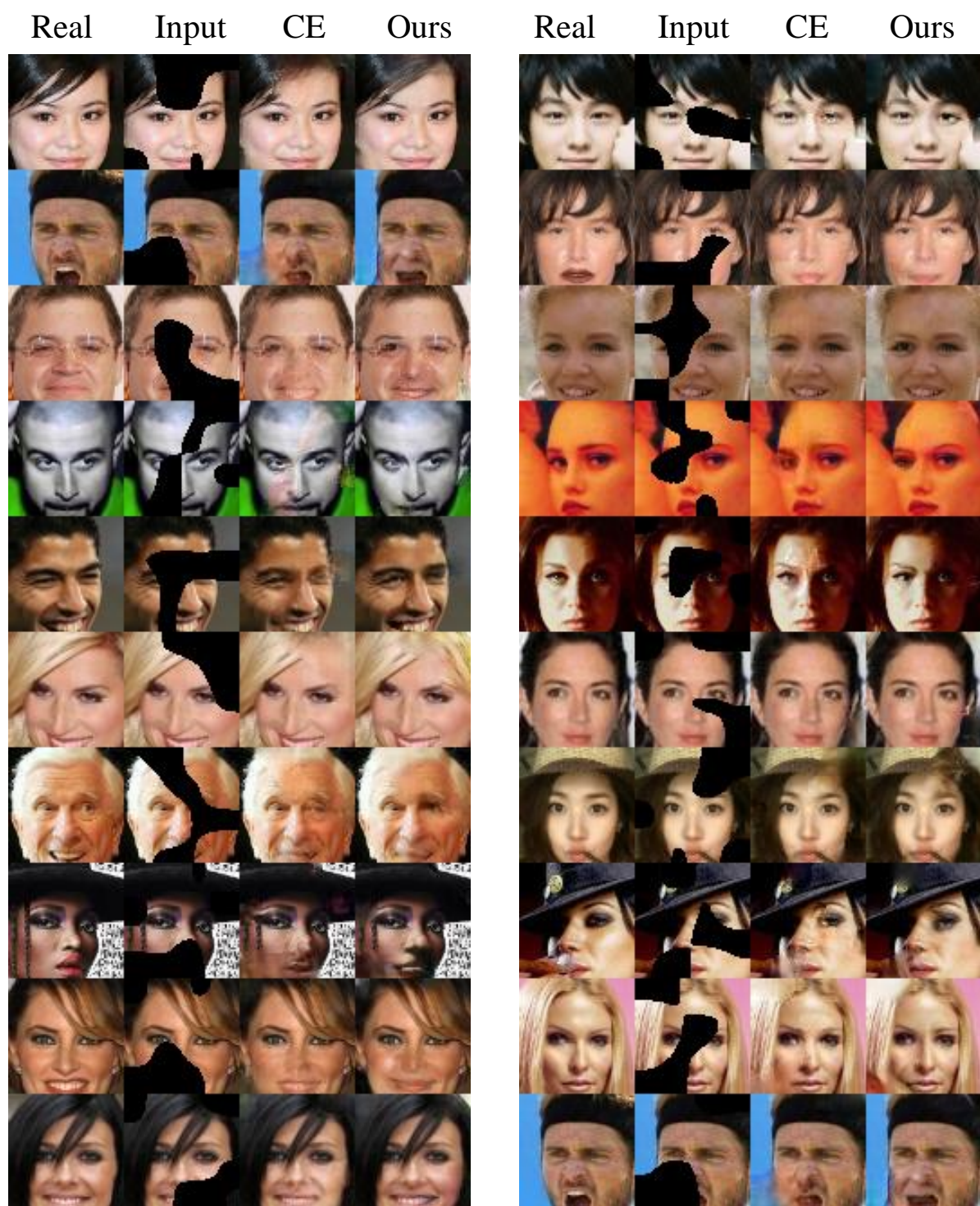


Figure 15. Additional results on the celebA dataset.

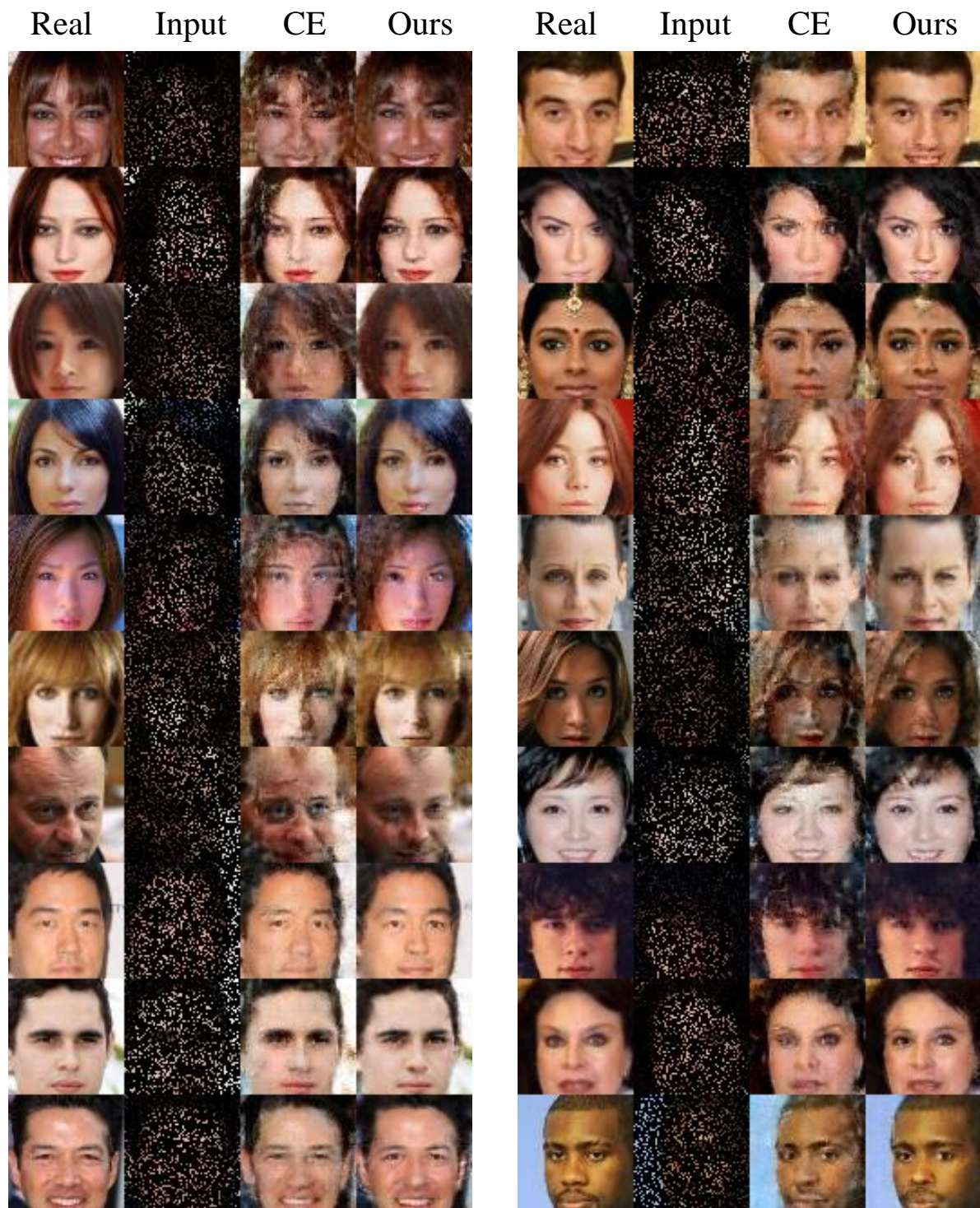


Figure 16. Additional results on the celebA dataset.

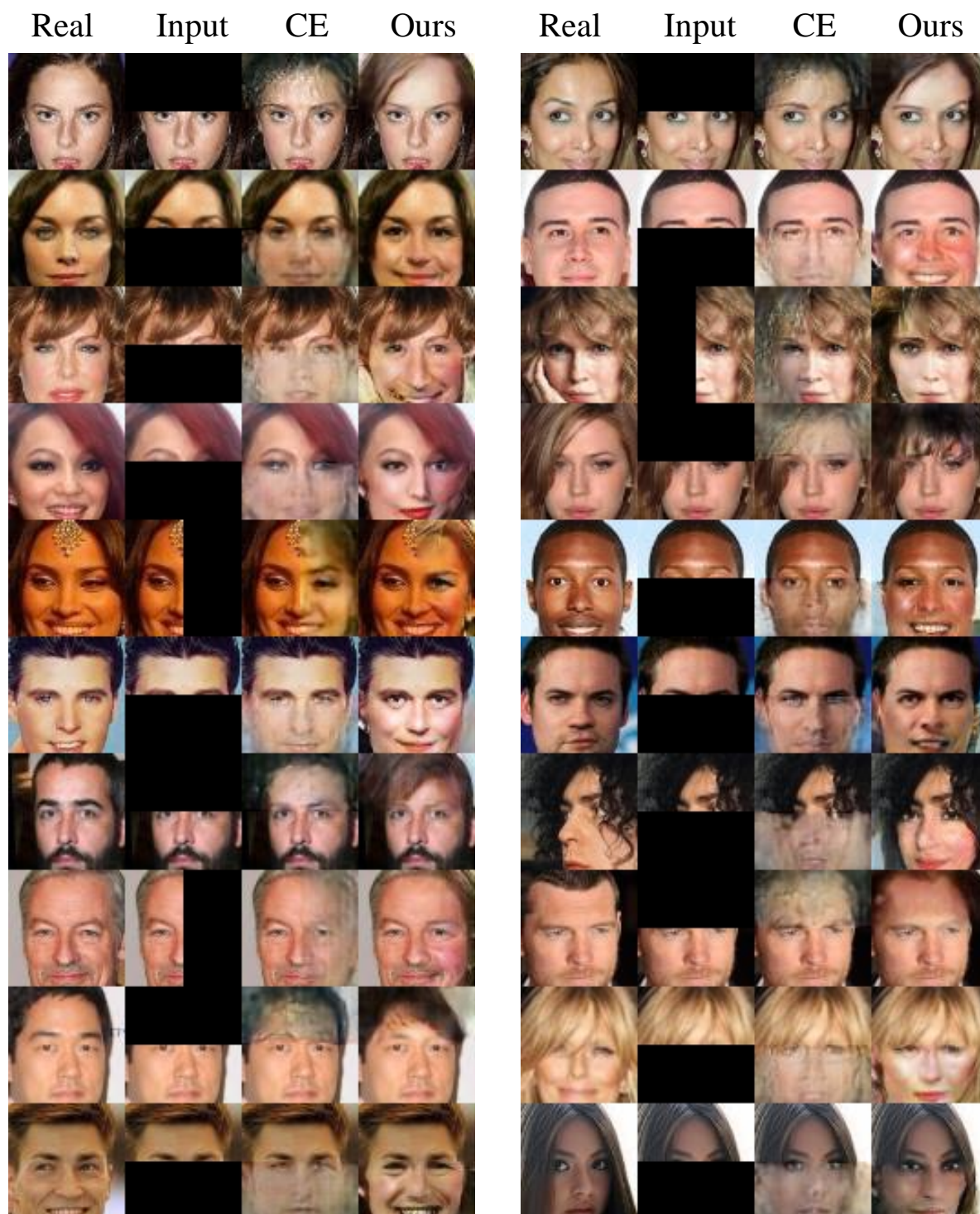


Figure 17. Additional results on the celebA dataset.



Figure 18. Additional results on the SVHN dataset.



Figure 19. Additional results on the SVHN dataset.

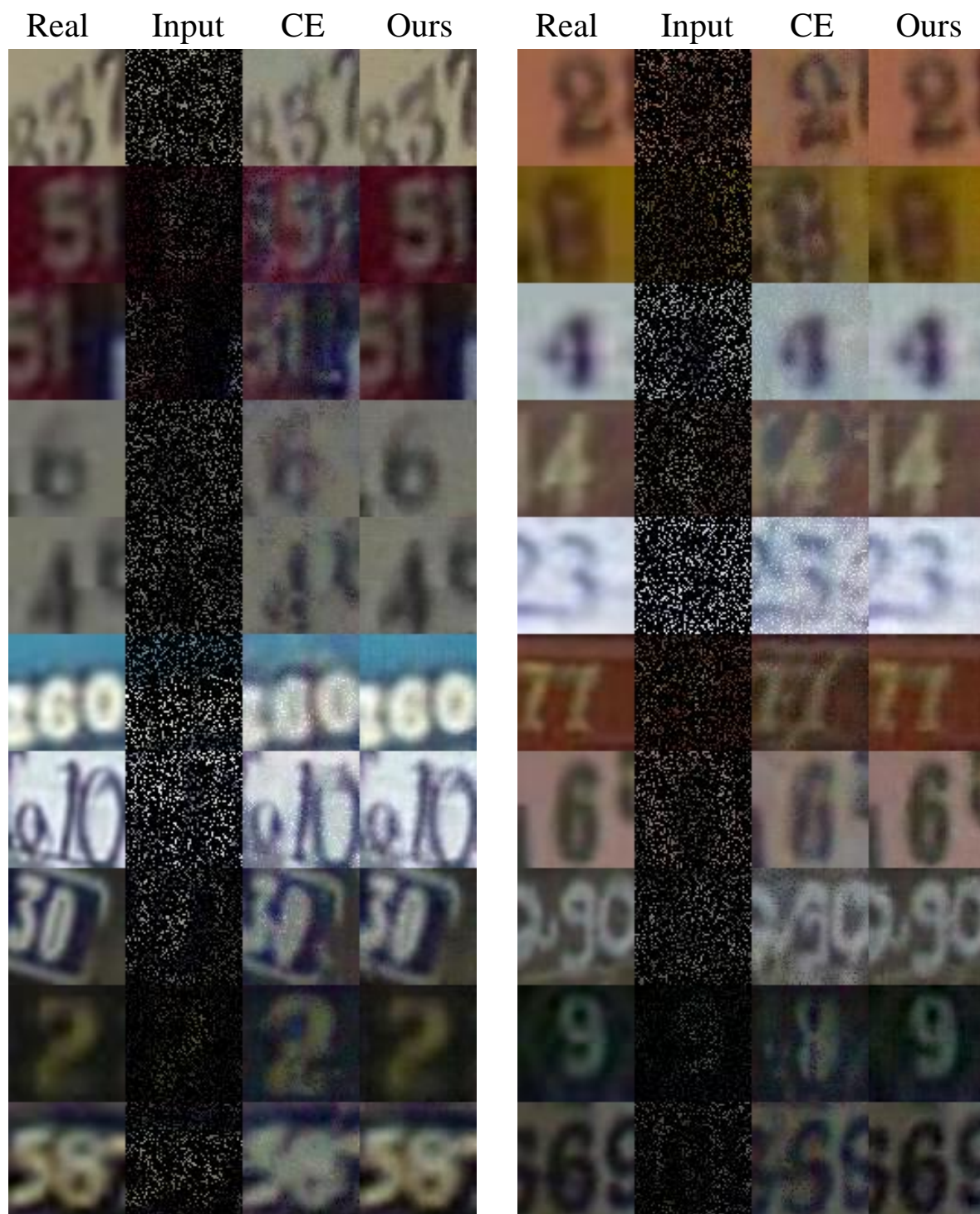


Figure 20. Additional results on the SVHN dataset.



Figure 21. Additional results on the SVHN dataset.

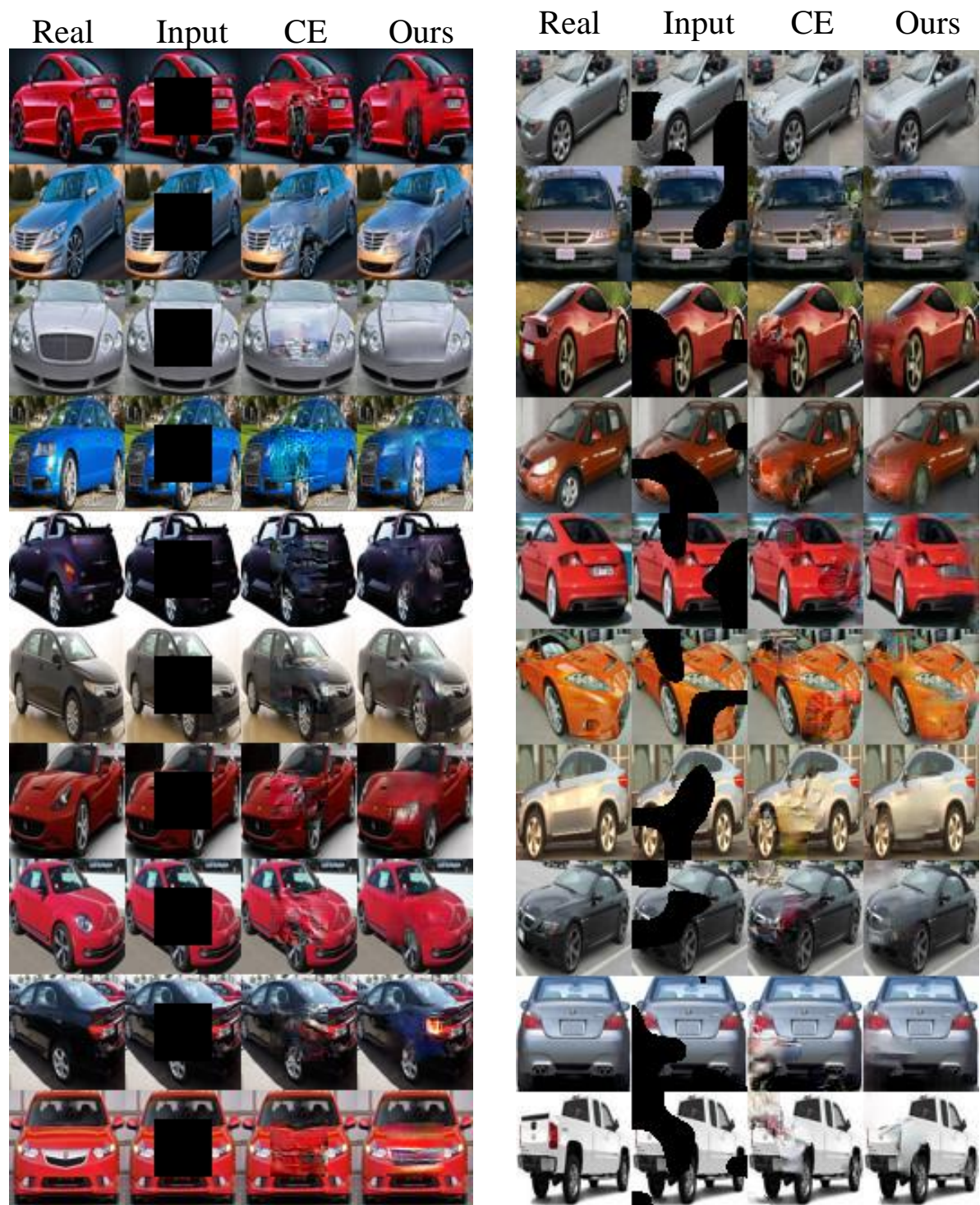


Figure 22. Additional results on the car dataset.

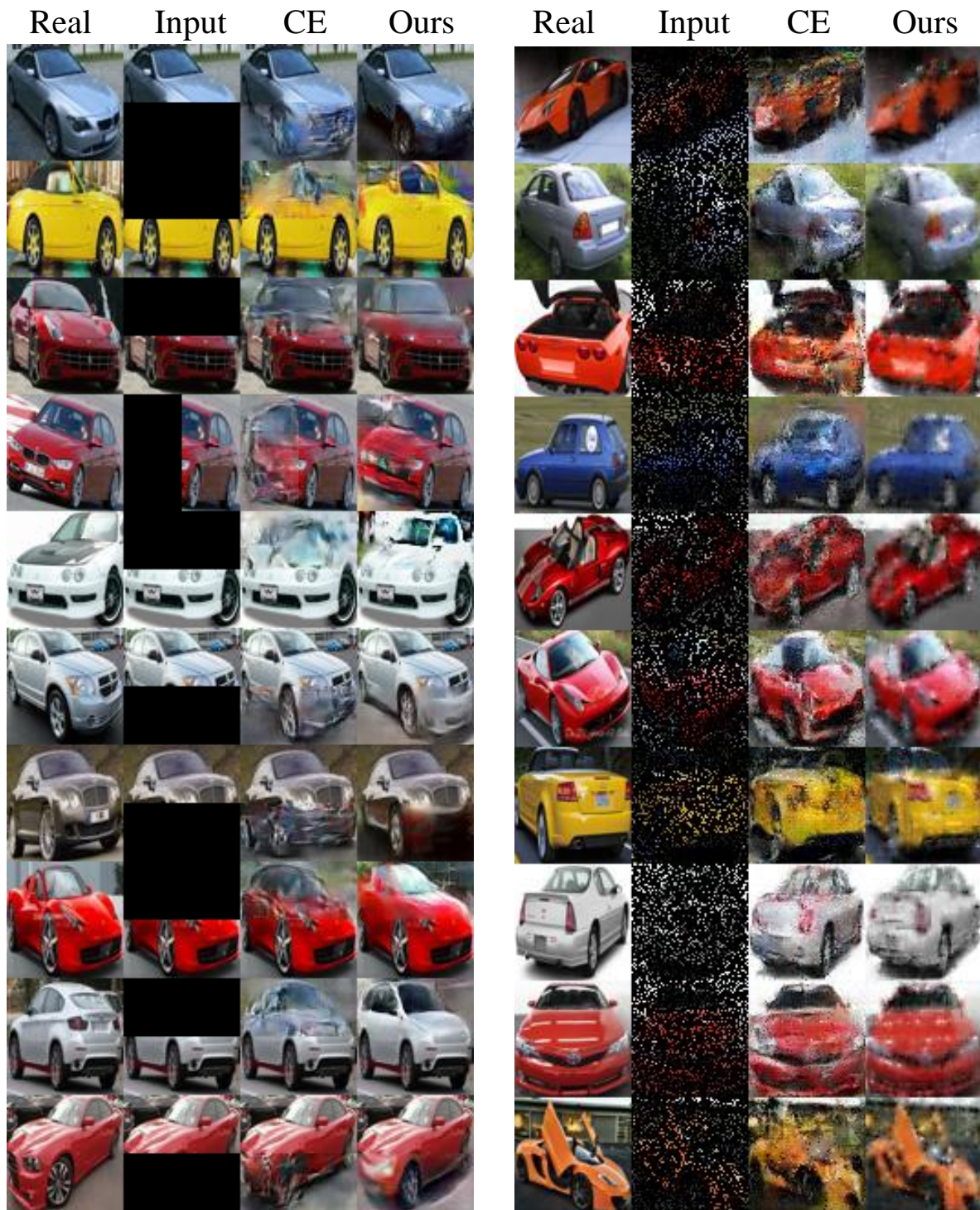


Figure 23. Additional results on the car dataset.