Image Inpainting using Block-wise Procedural Training with Annealed Adversarial Counterpart

Anonymous ECCV submission

Paper ID 1283

Abstract. Recent advances in deep generative models have shown promising potential in image inpanting, which refers to the task of predicting missing pixel values of an incomplete image using the known context. However, existing methods can be slow or generate unsatisfying results with easily detectable flaws. In addition, there is often perceivable discontinuity near the holes and require further post-processing to blend the results. We present a new approach to address the difficulty of training a very deep generative model to synthesize high-quality photo-realistic inpainting. Our model uses conditional generative adversarial networks (conditional GANs) as the backbone, and we introduce a novel block-wise procedural training scheme to stabilize the training while we increase the network depth. We also propose a new strategy called adversarial loss annealing to reduce the artifacts. We further describe several losses specifically designed for inpainting and show their effectiveness. Extensive experiments and user-study show that our approach outperforms existing methods in several tasks such as inpainting, face completion and image harmonization. Finally, we show our framework can be easily used as a tool for interactive guided inpainting, demonstrating its practical value to solve common real-world challenges.

Keywords: deep generative model, image inpainting, image harmonization, image composition.

1 Introduction

Image inpainting is the task to fill in the missing part of an image with visually plausible contents. It is one of the most common operations of image editing [1] and low-level computer visions [2, 3]. The goal of image inpainting is to create semantically plausible contents with realistic texture details. The inpainting can be consistent with the original image or different but coherent with the known context. Other than restoring and fixing damaged images, inpainting can also be used to remove unwanted objects, or in the case of guided inpainting, be used to composite with another guide image. In the latter scenario, we often need inpainting to fill in the gaps and remove discontinuities between target region of interest on the guide image and the source context. Image harmonization is also required to adjust the appearance of the guide image such that it is compatible with the source, making the final composition appear natural.

Traditional image inpainting methods mostly develop texture synthesis techniques to address the problem of hole-filling [4, 2, 5–8]. In [6], Barnes et al.

(a) Input (b) GLI [10] (c) Ours (c) Input (d) DH [9] (e) Ours

Fig. 1. An example of inpainting (top) and harmonzation (bottom) comparing with state-of-the-art methods. Zoom in for best viewing quality.

proposes the Patch-Match algorithm which efficiently searches the most similar patch to reconstruct the missing regions. Wilczkowiak et al. [8] takes further steps and detects desirable search regions to find better match patches. However, these methods only exploit the low-level signal of the known contexts to hallucinate missing regions and fall short of understanding and predicting high-level semantics. Furthermore, it is often inadequate to capture the global structure of images by simply extending texture from surrounding regions. Another line of work in inpainting aims to fill in holes with contents from another guide image by using composition and harmonization [3, 9]. The guide is often retrieved from a large database based on similarity with the source image and is then combined with the source. Although these methods are able to propagate high-frequency details from the guide image, they often lead to inconsistent regions and gaps which are easily detectable with human eyes.

More recently, deep neural networks have shown excellent performance in various image completion tasks, such as texture synthesis and image completion. For inpainting, adversarial training becomes the de facto strategy to generate sharp details and natural looking results [11–14, 10]. Pathak et al. [11] first proposes to train an encoder-decoder model for inpainting using both reconstruction loss and adversarial loss. In [12], Yeh et al. uses a pre-trained model to find the most similar encoding of a corrupted image and uses the found encoding to synthesize what is missing. In [14], Yang et al. proposes a multi-scale approach and optimizes the hole contents such that neural feature extracted from a pre-trained CNN matches with features of the surrounding context. The optimization scheme improves the inpainting quality and resolution at the cost of computational efficiency. In [10], Iizuka et al. proposes a deep generative model trained with global and local adversarial losses, and can achieve good inpainting performance for mid-size images and holes. However, it requires extensive training (two months as described in the paper), and the results often contain excessive artifacts and noise patterns. Another limitation of [14] and [10] is that they are unable to handle perceptual discontinuity, making it necessary to resort to post-processing (e.g. Poisson blending).

In practice, we found that directly training a very deep generative network to synthesize high-frequency details is challenging due to optimization difficulty and unstable adversarial training. As a result, as the network becomes deeper, the inpainting quality may worsen. To overcome this difficulty, we come up with a new training scheme that guides and stabilizes the training of a very

deep generative model, which, combining with carefully designed training losses, significantly reduces artifacts and improves the quality of results. The main strategy, referred to as **B**lock-wise **P**rocedural **T**raining (**BPT**), progressively increases the number of residual blocks and the depth of the network. With BPT, we first train a cGAN-based **Generator Head** until converging, followed by adding more residual blocks one at a time, which refines and improves the results. This enables us to train a network deeper than [10] and generates more realistic-looking details. We also observe that to reduce the noise level, it is essential to steadily reduce the weight of adversarial loss given to the generator. We refer this training scheme as **A**dversarial **L**oss **A**nnealing (**ALA**). Finally, we also propose two new losses specifically designed for inpainting: the **P**atch **P**erceptual **L**oss (**PPL**) and **M**ulti-**S**cale **P**atch **A**dversarial **L**oss (**MSPAL**). Experiments show that these losses work better than traditional losses used for inpainting, such as ℓ_2 and the more general adversarial loss.

To evaluate the proposed approach, we conduct extensive experiments on a variety of datasets. Shown by qualitative and quantitative results, also by the user-study, our approach generates high-quality inpainting results and outperforms other state-of-the-art methods. We also show that our framework, although being designed for inpainting, can be directly applied on general image translation tasks such as image harmonization and composition, and easily achieve results superior to other methods (Fig. 1). This enables us to train a multi-task model and use it for interactive guided inpainting, which is a very common and useful image editing scenario. We will describe it in detail in Sec. 3.4.

In summary, in this paper we present:

1. A novel and effective framework that generates high-quality results for several image editing tasks like inpainting and harmonization. We also provide a thorough analysis and ablation study about the components in our framework.
2. Extensive qualitative and quantitative evaluations on a variety of datasets, such as scenes, objects, and faces. We also conduct a user-study to make fair and rigorous comparisons with other state-of-the-art methods.
3. Results of interactive guided inpainting as a novel and useful application of our approach.

2 Our Method

Our method consists of training a generator head as initialization and optimizing additional residual blocks as refinement. The generator head is trained for inpainting and is based on conditional GAN framework. Vanilla cGAN for image inpainting consists of a Generator G and a Discriminator D. The generator learns to predict the hole contents and restore the complete image, while the discriminator learns to distinguish real images from generated images. Using the original image as ground truth, the model is trained in a self-supervised manner via the following minimax objective:

$$\min_G \max_D E_{(s,x)}[\log D(s,x)] + E_s[\log(1 - D(s, G(s)))]. \quad (1)$$

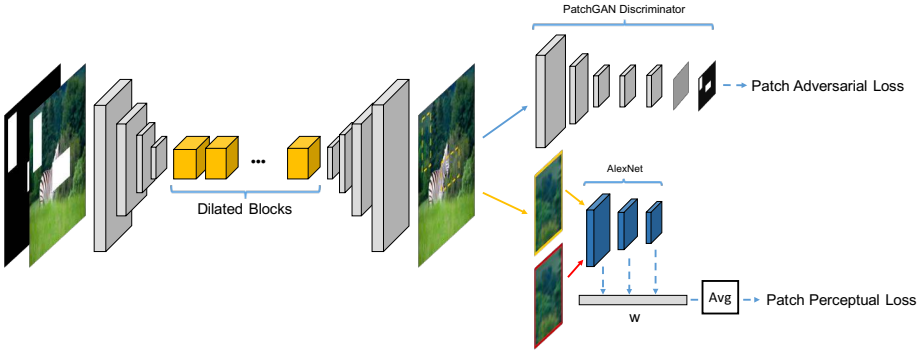


Fig. 2. Generator head and the training losses. We only illustrate one scale of Patch Adversarial Loss and Patch Perceptual Loss. Note that to compute the Patch Adversarial Loss, we need to use the mask to find out which patch overlaps with the hole.

Here s and x are the incomplete image as input and the original image as target. $G(s)$ is the output of the generator which could be the hole content or the complete image. In the case when $G(s)$ restores the complete image, only the contents inside the hole are kept to combine with the known regions of s.

2.1 The Generator Head

Existing works have investigated different architectures of G, most notably the encoder-decoder style of [11] and the FCN style of [10]. [10] shows that FCN generates higher-quality and less blurred results comparing with encoder-decoder, largely because the network used is much deeper and is fully convolutional. In contrast, the encoder-decoder in [11] uses an intermediate bottleneck, fully-connected layer, which results in size reduction and information loss.

Similar to [10], our generator head is based on FCN and leverage the many properties of convolutional neural networks, such as translation invariance and parameter sharing. However, a major limitation of FCN is the constraint of the receptive field size. Given the convolution layers are locally connected, pixels far away from the hole carry no influence on the hole. We propose several modifications to alleviate this drawback. First, we build our network with three components: a down-sampling front end to reduce the size, followed by multiple residual blocks [15], and an up-sampling back end to restore the full dimension. Using down-sampling increases the receptive field of the residual blocks and makes it easier to learn the transformation at a smaller size. Second, we stack multiple residual blocks to further enlarge the view at later layers. Finally, we adopt the dilated convolution [16] in all residual blocks, with the dilation factor set to 2. Dilated convolutions use spaced kernels, making it compute each output value with a wider view of input without increasing the number of parameters and computational burden. From experiments, we observe that increasing the size of the receptive field, especially by using dilated convolution, is critical for enhanced inpainting quality. In contrast, other image translation tasks such as

super-resolution, denoising, etc., usually rely more on local statistics rather than global context.

The detail of our architecture is as follows: the down-sampling front end consists of three convolutional layers, each with a stride of 2. The intermediate part contains 9 residual blocks, and the up-sampling back-end consists of three transposed convolution, also with a stride of 2. Each convolutional layer is followed by batch normalization (BN) and ReLu activation, except for the last layer which outputs the image. Similar architecture without dilated convolution has been used in [17] for image synthesis. Comparing with [10], our receptive field is much larger, as we adopted more down-sampling and dilated layers. This makes us generate better results with less artifacts. As ablation study, we compare different types of layers and architectures in Sec. 3.3.

2.2 The Training Losses

Different losses have been used to train an inpainting network. These losses can be cast into two categories. The first category which can be referred to as *similarity loss*, is used to measure the similarity between the output and the original image. The second category which we refer to as the *realism loss*, is used to measure how realistic-looking is the output image. We summarize the losses used in different inpainting methods in Table 1.

Method	Similarity Loss	Realism Loss
Context Encoder (CE) [11]	ℓ_2	Global Adversarial Loss
Global Local Inpainting (GLI) [10]	ℓ_2	Global and Local Adversarial Loss
Our Approach	Patch Perceptual Loss (PPL)	Improved Multi-Scale Adversarial Loss

Table 1. Comparison of training losses used in different methods.

Patch Perceptual Loss As shown in Table 1, using ℓ_2 as reconstruction loss to measure the difference between the output and the original image has been the default choice of previous inpainting methods. However, it is known that ℓ_2 loss does not correspond well to human perception of visual similarity (Zhang et al. [18]). This is because using ℓ_2 assumes each output pixel is conditionally independent of all others, which is not the case. An example is that blurring an image leads to small changes in terms of ℓ_2 distance but causes significant perceptual difference. Recent research suggests that a better metric for perceptual similarity is the internal activations of deep convolutional networks, usually trained on a high-level image classification task. Such loss is called "perceptual loss", and is used in various tasks such as neural style transfer [19], image super-resolution [20], and conditional image synthesis [21, 22].

Based on this observation, we propose a new "patch perceptual loss" as the substitute of the ℓ_2 losses. Traditional perceptual loss typically uses VGG-Net and computes the ℓ_2 distance of the activations on a few feature layers. Recently, [18] trained a specific perceptual network based on AlexNet to measure the perceptual differences between two image patches, making it an ideal candidate for our task. The perceptual network computes the activations and sums

up the ℓ_2 distances across all feature layers, each scaled by a learned weight. Furthermore, taking both the local view and the global view into account, we compute PPL at two scales. Local PPL considers the local hole patch, while the global PPL slightly zooms out to cover a larger contextual area. More formally, our PPL is defined as:

$$\sum_{p=1,2} PPL_k(G(s)_p, x_p) = \sum_{k=1,2} \sum_{l} \frac{1}{H_l W_l} \sum_{h,w} \parallel w_l^T \odot (\hat{F}(x_p)^l_{hw} - \hat{F}(G(s)_p)^k_{hw}) \parallel_2^2 \quad (2)$$

Here p refers to the hole patch. $\hat{F}$ is the AlexNet and l is the feature layer. w_l is the layer-wise learned weight. Ablation study in Sec. 3.3 shows that our patch perceptual loss gives better inpainting quality than traditional ℓ_2.

Multi-Scale Patch Adversarial Loss Adversarial losses are given by trained discriminators to discern whether an image is real or fake. The global adversarial loss of [11] takes the entire image as input and outputs a single real/fake prediction which does not consider local realism of holes. The additional local adversarial loss of [10] adds another discriminator specifically for the hole, but it requires the hole to be fixed in shape and size during training to fit the local discriminator. To consider both the global view and the local view, and to be able to use multiple holes of arbitrary shape, we propose to use PatchGAN discriminators [23] at three scales of image resolutions. The discriminator at each scale is identical and only the input is a differently scaled version of the *entire image*. Each discriminator is a fully convolutional PatchGAN and outputs a vector of real/fake predictions and each value corresponds to a local image patch. In this way, the discriminators are trained to classify global and local patches across the image at multiple scales, and it enables us to use multiple holes of arbitrary shapes since now the input is the entire image rather than the hole itself. However, since the output image is the composition of synthesized holes and known background, directly using PatchGAN is problematic because it does not differentiate between the hole patches (*fake*) and the background patches (*real*). To address this issue, when computing the PatchGAN loss, only the patches that overlap with the holes are labeled as fake. More formally, our Patch-wise Adversarial Loss is defined as:

$$\min_G \max_{D_1,D_2,D_3} \sum_{k=1,2,3} L_{GAN}(G, D_k) \quad (3)$$

$$= \sum_{k=1,2,3} E_{(s_k,x_k)}[\log(s_k, x_k)] + E_{s_k}[\log(Q_k - D_k(s_k, G(s_k)))]. \quad (4)$$

Here k refers to the image scale, and Q_k is a patch-wise real/fake vector, based on whether the patch overlaps with the holes. Using multiple GAN discriminators at the same or different image scale has been proposed in unconditional GANs [24] and conditional GANs [17]. Here we extend the design to take into account the holes, which is critical for obtaining semantically and locally coherent image completion results.

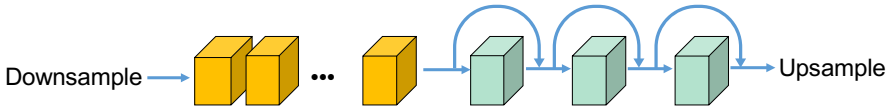


Fig. 3. Illustration of block-wise procedural training. The yellow residual blocks refer to the generator head which is trained first. The green residual blocks are progressively added one at a time. We also draw the skip connections between the already trained residual blocks and the up-sampling back end.

To summarize, our full objective combines both losses is therefore defined as:

$$\min_{G}((\max_{D_1,D_2,D_3} \lambda_{Adv} \sum_{k=1,2,3} L_{GAN}(G,D_k)) + \lambda_{PPL} \sum_{k=1,2} PPL_k(G(s)_p, x_p)). \quad (5)$$

We use λ_{Adv} and λ_{PPL} to control the importance of the two terms. In our experiment, we set initial $\lambda_{Adv} = 1$ and $\lambda_{PPL} = 10$ as used in [11, 17]. As training progresses, we gradually decrease the weight of λ_{Adv} based on adversarial loss annealing (Sec. 2.3).

Fig. 2 illustrates the architecture of our generator head and the training losses.

2.3 Blockwise Procedural Training with Adversarial Loss Annealing

Our experiments show that using the described generator head and training losses already generate results of descent quality. An intuitive effort to improve the results would be to stack more intermediate residual blocks to further expand the receptive view and increase the expressiveness of the model. However, we found that directly stacking more residual blocks makes it more difficult to stabilize the training. As the parameter space becomes much larger, it is also more challenging to find local optimum. In the end, the inpainting quality deteriorates as the model depth increases.

To address the challenge of training a deeper network, we propose to use procedural block-wise training to gradually increase the depth of the inpainting network. More specifically, we begin by training the generator head until it converges. Then, we add a new residual block after the already trained residual blocks, right before the back-end up-sampling layers. In order to smoothly introduce the new residual block without breaking the trained model, we add another skip path from the already trained residual blocks to the up-sampling layers. Initially, the weight of the skip path is set to 1, while the weight of the path containing the new block is set to 0. This essentially makes the initial network identical to the already trained network. We then slowly decrease the weight of the skip path and increase the weight of the new residual block as training progresses. In this way, the newly introduced residual block is trained to be a fine-tuning component, which adds another layer of fine details to the original results. This step are repeated multiple times, and each time we deepen the network by adding a new residual block. In our experiments, we found that the results improve significantly over generator head, after training with the first

residual block added. The output becomes stabilized after three residual blocks, and very little changes can be detected if more residual blocks are brought in. We illustrate the procedural training process in Fig. 8.

The block-wise procedural training has several benefits. First, it guides the training process of a very deep generator. Starting with the generator head and gradually fine-tuning with more residual blocks makes it easier to discover the mapping between the incomplete image and the complete image, even though the search space is huge given the diversity of natural images and the randomness of holes. Another benefit is training efficiency, as we found decoupling the training of the generator head and the fine-tuning of additional residual blocks requires significantly less training time comparing with training the network all at once. Similar idea of progressive training has been used in Progressive GAN [25]. However the task and the model are different in our case.

Adversarial Loss Annealing During training, the generator adversarial loss updates the generator weight if the discriminator successfully detects the generated image as fake:

$$\sum_{k=1,2,3} E_s[\log(\bar{Q} - D_k(s_k, G(s_k)))]. \tag{6}$$

Note that here $\bar{Q}$ reverses Q of 2.2 as this is the loss to the generator. We observe that the generator adversarial loss becomes dominant over PPL as training progresses, as the discriminator becomes increasingly good at detecting fake images during training. This is less of a problem for image synthesis tasks. However, for the inpainting task which requires the output to be faithful to the original image, the outcome is that the generator deliberately adds noise patterns to confuse the discriminators, leading to artifacts or incorrect textures. Based on this observation, we propose to use adversarial loss annealing, which decreases the weight of the adversarial loss for the generator at the time of adding new residual blocks. More formally, let the initial weight of the generator adversarial loss be λ^0_{adv}, and the weight of the generator adversarial loss be λ^i_{adv} after adding the i_{th} residual block. We found that simply decay the weight linearly by setting $\lambda^i_{G_{adv}} = 10^{-i}\lambda^0_{G_{adv}}$ gives results significantly better than using constant weight. In Sec. 4, we analyze the effect of ALA in more detail.

Finally, we summarize the discussed training schemes in Alg. 1.

Algorithm 1 Training the Inpainting Network

1: Set batch size to 8 and basic learning rate to $lr_0 \leftarrow 0.0002$.
2: Set $\lambda_{PPL} \leftarrow 10$ and $\lambda^0_{adv} \leftarrow 1$.
3: Train the Generator Head G_0 using MSPAL and PPL (2.2) for 150,000 iterations.
4: **for** i=1 to 3 **do**
5: Add the skip path and the residual block r_i.
6: Set $lr_i \leftarrow 10^{-i} lr_0$ and $\lambda^i_{G_{adv}} \leftarrow 10^{-i}\lambda^0_{G_{adv}}$.
7: Train the generator G_3 with the added r_i for 1,500 iterations.
8: **return** G_3

3 Results

In this section, we first describe our datasets and experiment setting (Sec. 3.1). We then provide quantitative and qualitative comparisons with several methods, and also report a subjective human perceptual test based on user-study (Sec. 3.2). In Sec. 3.3, we conduct several ablation study about the design choice of the framework. Finally, we show how our method can be applied to image harmonization and interactive guided inpainting (Sec. 3.4).

3.1 Experiment Setup

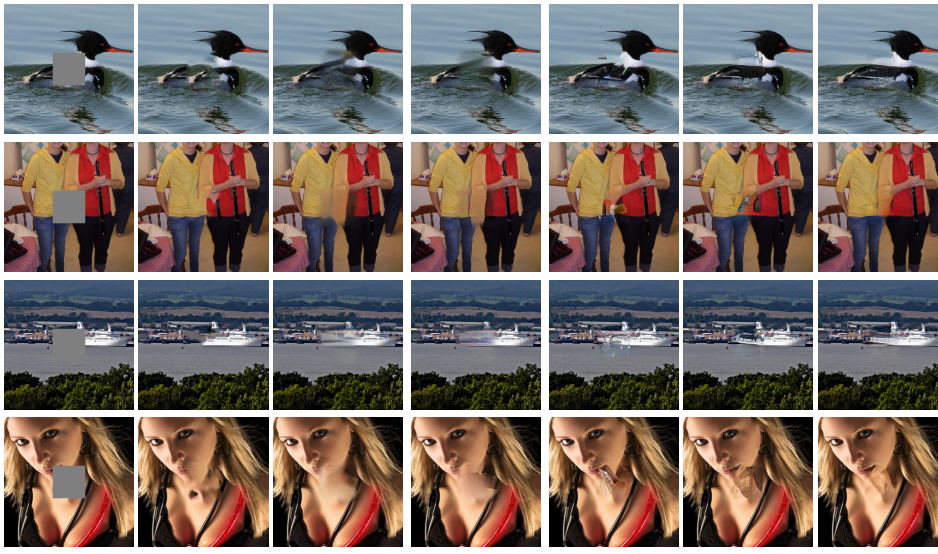
We evaluate our inpainting method on ImageNet [26], COCO [27], Place2 [28] and CelebA [29]. CelebA consists of 202,599 face images with various viewpoints and expressions. For a fair comparison, we follow the standard split with 162,770 images for training, 19,867 for validation and 19,962 for testing.

In order to compare with existing methods, we train on images of size 256x256. We also train another network at a larger scale of 512x512 to demonstrate its ability to handle higher resolutions (see supplementary material). For each image, we apply a mask with a single hole or multiple holes placed at random locations. The size of hole is between 1/4 and 1/2 of the image's size. We set the batch size to 8, and regardless of the actual dataset size, we train 150,000 iterations for the generator head and another 1,500 iterations for each additional residual block. For each dataset, the training takes around 2 days to finish on a single Titan X GPU, which is significantly less time than [10].

3.2 Comparison with Existing Methods

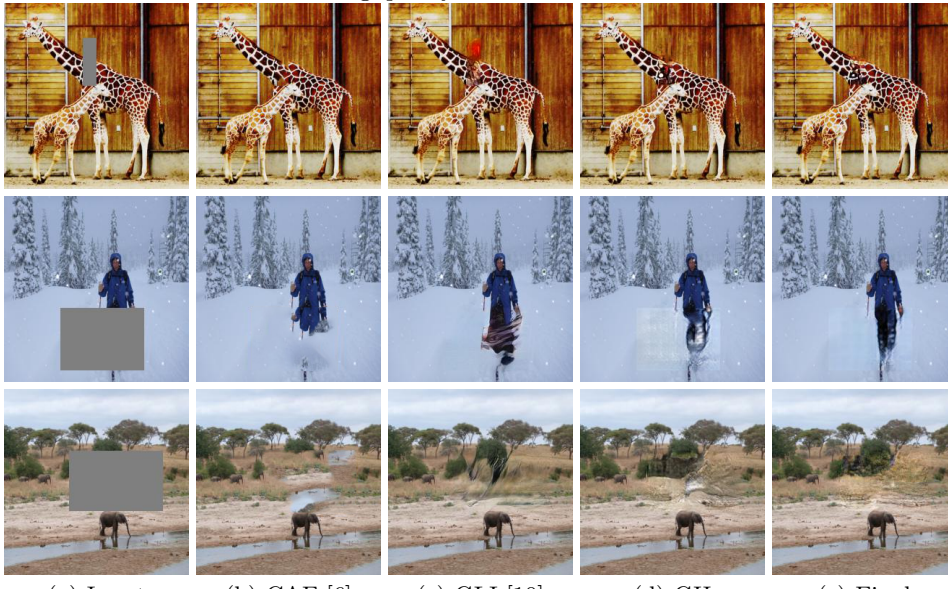
For ImageNet, COCO and Place2, we compare our results with Content-Aware Fill (CAF) [6], Context Encoder (CE) [11], Neural Patch Synthesis (NPS) [14] and Global Local Inpainting (GLI) [10]. For CE, NPS, and GLI, we use pre-trained models made available by the authors. CE and NPS are only trained with holes with fixed size and location (image center), while CAF and GLI can handle arbitrary holes. For fair comparisons, we evaluate on both settings. For center hole completion, we compare with CAF, CE, NPS and GLI on ImageNet [26] test images. For random hole completion, we compare with CAF and GLI using test images from COCO and Place2. Only GLI applies Poisson Blending as post-processing. We also show the results by our *Generator Head* and compare. The images shown in Fig. 4, 5 are randomly sampled from the test set.

We see that, while CAF can generate realist-looking details using patch propagation, they often fail to capture the global structure or synthesize new contents. CE's result is significantly blurrier as it can only inpaint low-resolution images. NPS's results heavily depend on CE's initialization. Our approach is much faster and generates results of higher visual quality most of the time. Comparing with GLI, our results do not need post-processing, and are more coherent with less artifacts. Our final results also show significant improvement over Generator Head, demonstrating the effectiveness block-wise procedural fine-tuning.



(a) Input (b) CAF [6] (c) CE [11] (d) NPS [14] (e) GLI [10] (f) GH (g) Final

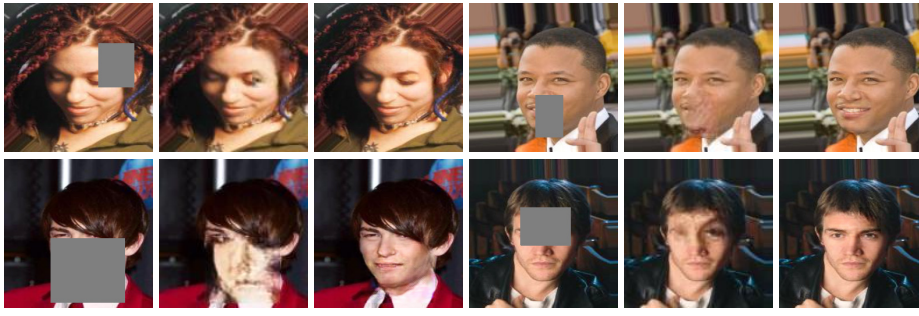
Fig. 4. Fixed hole result comparisons. GH are results generated by generator head, and Final are results generated with block procedural training. All images have original size 256x256. Zoom in for best viewing quality.



(a) Input (b) CAF [6] (c) GLI [10] (d) GH (e) Final

Fig. 5. Random hole completion comparison. GH are results generated by generator head, and Final are results generated with block procedural training. All images have original size 256x256. Zoom in for best viewing quality.

For CelebA, we compare our results with Generative Face Completion [13] in Fig. 6. Since [13]'s model inpaints images of 128x128, we directly up-sample the results to 256x256 to compare. The images shown are also chosen at random. We can see that while [13] is an approach specific for face inpainting, our method for general inpainting tasks actually generates much better face completion results.



(a) Input (b) GFC [13] (c) Ours (d) Input (e) GFC [13] (f) Ours

Fig. 6. Face completion results comparing with [13].

Quantitative Evaluation Table 2 shows quantitative comparison between CAF, CE, GLI and our approach. The values are computed based on a random subset of 200 images selected from the test set. Both ℓ_1 and ℓ_2 errors are computed with pixel values normalized between $[0, 1]$ and are summed up using all pixels of the image. We can see that our method performs better than other methods in terms of SSIM and ℓ_1 error. For ℓ_2, CAF and GLI have smaller errors. This is understandable as our model is trained with perceptual loss rather than ℓ_2 loss. Besides, ℓ_2 loss awards averaging color values and does not faithfully reflect perceptual quality. From the numerical values, we can also see that procedural fine-tuning reduces the errors from the generator head.

Method	Mean ℓ_1 Error	Mean ℓ_2 Error	SSIM
CAF [6]	968.8	**209.5**	0.9415
	1660	**363.5**	0.9010
CE [11]	2693	545.8	0.7719
	N/A	N/A	N/A
GLI [10]	868.6	269.7	0.9452
	1640	378.7	0.9020
Ours (Generator Head)	913.8	245.6	0.9458
	1629	439.4	0.9073
Ours (Final)	**838.3**	253.3	**0.9486**
	1609	427.0	**0.9090**

Table 2. Numerical comparison between CAF, CE and GLI, our generator head results and our final results. Up/down are results of center/random region completion. Note that for SSIM, larger values mean greater similarity in terms of content structure and indicate better performance.

User Study To more rigorously evaluate the performance, we conduct a user study based on the random hole results. We asked for feedback from 20 users, by giving each user 30 tuples of images to rank. Each tuple contains results of NPS, GLI, and ours. The user study shows that our method have highest scores, outperforming NPS and GLI by a large margin. Among 600 total comparisons, our results are ranked the best 72.3% of the time. Our results are overwhelmingly better than NPS, as our results are ranked better 95.8% of the time. Comparing with GLI, our results are ranked better 73.5% of the time and are ranked the same 15.5% of the time.

3.3 Ablation Study

Comparison of Convolutional Layer Choosing the proper convolutional layer improves the inpainting quality and also reduces the noise. We consider three types of convolutional layers: original, dilated [16] and interpolated [30]. We train three networks under the same setting to specifically test their effects. We observed that using dilation significantly improves the inpainting quality while using interpolated convolution tends to generate over-smoothed results. Fig. 7 shows a qualitative comparison.

(a) (b) (c) (d) (e)

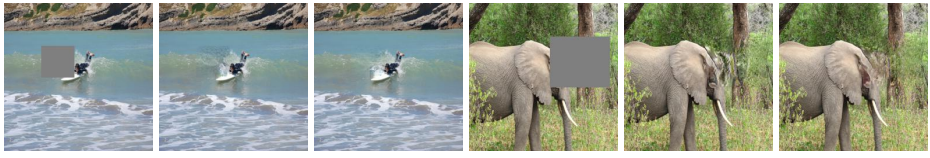
Fig. 7. Visual comparison of a result using different types of convolutional layers. (a) Input; (b) Original Conv; (c) Interpolated Conv; (d) Dilated+Interpolated Conv; (e) Dilated Conv (ours).

Effect of Procedural Training Qualitative comparisons in Sec. 3.2 shows that procedurally adding residual blocks to fine-tune improves the results. However, one may wonder whether we can directly train a deeper network. To find out, we trained a network with 12 residual blocks from scratch, keeping all other hyper-parameters the same. Fig. 8 shows two examples of the result comparisons. We found that further increasing the number of residual blocks has a negative effect on inpainting, possibly due to a variety of factors such as gradient vanishing, optimization difficulty, etc. This demonstrates the necessity and importance of the procedural training scheme.

Effect of Adversarial Loss Annealing We also investigate the effect of adversarial loss annealing and train another network without using ALA. Visually, we observe that using adversarial loss annealing reduces the noise level without sacrificing the overall sharpness and realism. We show two randomly selected examples in Fig. 9 to illustrate the effects.

(a) Input (b) (c) (d) Input (e) (f)

Fig. 8. Visual comparison of results by directly training a network of 12 blocks ((b),(e)) and procedural training ((c), (f)).



(a) Input (b) w/o ALA (c) ALA (d) Input (e) w/o ALA (f) ALA

Fig. 9. Visual comparison of results by training training without and with ALA.

Due to space constraint, more ablation study can be found in supplementary material.

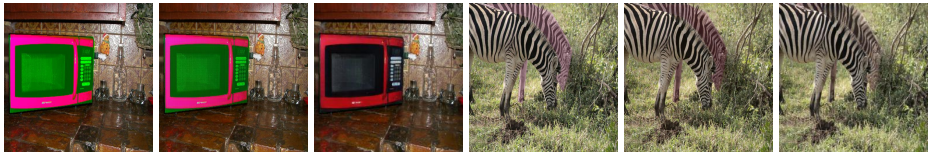
3.4 Interactive Guided Inpainting

In this section, we consider object-based guided inpainting as a practical scenario. Specifically, we assume the user would like to add to the input image with objects from another guide image. Given an input image I and a guide image I_g, we allow the user to select a region of I_s by dragging a bounding box B_g containing the object that he desires to add. Note that unlike previous settings of image composition, we do not require the user to accurately segment the object, but instead only providing a bounding box is sufficient. This greatly reduces the workload and simplifies the task. Next, we perform segmentation use a network to extract the object foreground. The segmentation network is adapted from Mask R-CNN [31] and trained on COCO, but is agnostic to object categories and only considers the foreground/background classification. Finally, we resize and paste B_g to I. To make the composition look natural and realistic, we need to remove and inpaint the background of B_g and also perform harmonization on the foreground object such that it is coherent the overall image appearance.

To accomplish this task with known segmentation, we need to address two separate problems: inpainting and harmonization. Our model can be easily extended to train for both tasks at the same time, only requiring changing the input and output. To train this model, the data acquisition is similar to [9], where we create artificially composited images using statistics color transfer [32] and another randomly chosen guide image. More detailed data acquisition steps can be found in supplementary material.

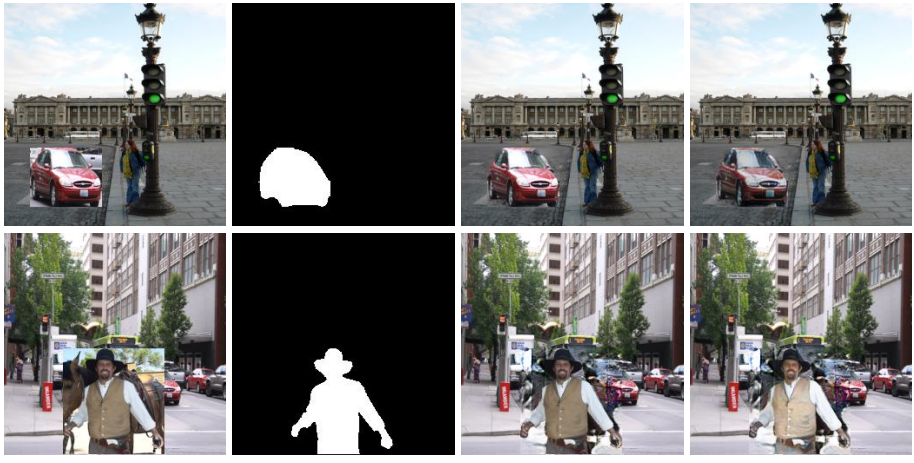
The network jointly trained for inpainting and harmonization performs well in both tasks. Fig. 10 shows that it generates harmonization results better

than [9], which is the state-of-the-art deep harmonization method. Finally, in Fig. 11 we show several examples of interactive guided inpainting results.



(a) Input (b) DH [9] (c) Ours (d) Input (e) DH [9] (f) Ours

Fig. 10. Examples of image harmonization results. For (a) and (d), the microwave and the zebra on the back have unusual color. Our method correctly adjusts their appearance and makes the images coherent and realistic.



(a) Input (b) Segmentation (c) Inpainting (d) Final result

Fig. 11. Examples of interactive guided inpainting result. The segmentation mask is given by our foreground/background segmentation network trained on COCO. The final result combines the outputs of harmonization and inpainting.

4 Conclusions

We propose a novel model and training scheme to address the inpainting problem and achieve excellent performance on several tasks, including general inpainting, image harmonization and guided inpainting. Although we only explore the ability of our approach in image editing related tasks, we believe it is a general methodology that could be easily applied to other generative problems, such as image translation, image synthesis, etc. As future work, we plan to study the effectiveness of the proposed network and losses, and especially the procedural training scheme on a larger scope.

References

1. Gatys, L., Ecker, A.S., Bethge, M.: Texture synthesis using convolutional neural networks. In: Advances in Neural Information Processing Systems. (2015) 262–270
2. Komodakis, N.: Image completion using global optimization. In: Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on. Volume 1., IEEE (2006) 442–452
3. Hays, J., Efros, A.A.: Scene completion using millions of photographs. In: ACM Transactions on Graphics (TOG). Volume 26., ACM (2007) 4
4. Bertalmio, M., Sapiro, G., Caselles, V., Ballester, C.: Image inpainting. In: Proceedings of the 27th annual conference on Computer graphics and interactive techniques, ACM Press/Addison-Wesley Publishing Co. (2000) 417–424
5. Wexler, Y., Shechtman, E., Irani, M.: Space-time video completion. In: Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on. Volume 1., IEEE (2004) I–I
6. Barnes, C., Shechtman, E., Finkelstein, A., Goldman, D.B.: Patchmatch: A randomized correspondence algorithm for structural image editing. ACM Transactions on Graphics-TOG **28**(3) (2009) 24
7. Bertalmio, M., Vese, L., Sapiro, G., Osher, S.: Simultaneous structure and texture image inpainting. IEEE transactions on image processing **12**(8) (2003) 882–889
8. Wilczkowiak, M., Brostow, G.J., Tordoff, B., Cipolla, R.: Hole filling through photomontage. In: BMVC 2005-Proceedings of the British Machine Vision Conference 2005. (2005)
9. Tsai, Y.H., Shen, X., Lin, Z., Sunkavalli, K., Lu, X., Yang, M.H.: Deep image harmonization. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2017)
10. Iizuka, S., Simo-Serra, E., Ishikawa, H.: Globally and locally consistent image completion. ACM Transactions on Graphics (TOG) **36**(4) (2017) 107
11. Pathak, D., Krahenbuhl, P., Donahue, J., Darrell, T., Efros, A.A.: Context encoders: Feature learning by inpainting. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2016) 2536–2544
12. Yeh, R., Chen, C., Lim, T.Y., Hasegawa-Johnson, M., Do, M.N.: Semantic image inpainting with perceptual and contextual losses. arXiv preprint arXiv:1607.07539 (2016)
13. Li, Y., Liu, S., Yang, J., Yang, M.H.: Generative face completion. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Volume 1. (2017) 6
14. Yang, C., Lu, X., Lin, Z., Shechtman, E., Wang, O., Li, H.: High-resolution image inpainting using multi-scale neural patch synthesis. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Volume 1. (2017) 3
15. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2016) 770–778
16. Yu, F., Koltun, V.: Multi-scale context aggregation by dilated convolutions. arXiv preprint arXiv:1511.07122 (2015)
17. Wang, T.C., Liu, M.Y., Zhu, J.Y., Tao, A., Kautz, J., Catanzaro, B.: High-resolution image synthesis and semantic manipulation with conditional gans. arXiv preprint arXiv:1711.11585 (2017)
18. Zhang, R., Isola, P., Efros, A.A., Shechtman, E., Wang, O.: The unreasonable effectiveness of deep features as a perceptual metric. arXiv preprint arXiv:1801.03924 (2018)

19. Gatys, L.A., Ecker, A.S., Bethge, M.: Image style transfer using convolutional neural networks. In: Computer Vision and Pattern Recognition (CVPR), 2016 IEEE Conference on, IEEE (2016) 2414–2423
20. Johnson, J., Alahi, A., Fei-Fei, L.: Perceptual losses for real-time style transfer and super-resolution. In: European Conference on Computer Vision, Springer (2016) 694–711
21. Dosovitskiy, A., Brox, T.: Generating images with perceptual similarity metrics based on deep networks. In: Advances in Neural Information Processing Systems. (2016) 658–666
22. Chen, Q., Koltun, V.: Photographic image synthesis with cascaded refinement networks. In: The IEEE International Conference on Computer Vision (ICCV). Volume 1. (2017)
23. Isola, P., Zhu, J.Y., Zhou, T., Efros, A.A.: Image-to-image translation with conditional adversarial networks. arXiv preprint arXiv:1611.07004 (2016)
24. Durugkar, I., Gemp, I., Mahadevan, S.: Generative multi-adversarial networks. arXiv preprint arXiv:1611.01673 (2016)
25. Karras, T., Aila, T., Laine, S., Lehtinen, J.: Progressive growing of gans for improved quality, stability, and variation. arXiv preprint arXiv:1710.10196 (2017)
26. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al.: Imagenet large scale visual recognition challenge. International Journal of Computer Vision **115**(3) (2015) 211–252
27. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: European conference on computer vision, Springer (2014) 740–755
28. Zhou, B., Khosla, A., Lapedriza, A., Torralba, A., Oliva, A.: Places: An image database for deep scene understanding. arXiv preprint arXiv:1610.02055 (2016)
29. Ziwei Liu, Ping Luo, X.W., Tang, X.: Deep learning face attributes in the wild. In: Proceedings of International Conference on Computer Vision (ICCV). (2015)
30. Odena, A., Dumoulin, V., Olah, C.: Deconvolution and checkerboard artifacts. Distill (2016)
31. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask r-cnn. In: Computer Vision (ICCV), 2017 IEEE International Conference on, IEEE (2017) 2980–2988
32. Reinhard, E., Adhikhmin, M., Gooch, B., Shirley, P.: Color transfer between images. IEEE Computer graphics and applications **21**(5) (2001) 34–41