Contextual Based Image Inpainting: Infer, Match and Translate

Anonymous ECCV submission

Paper ID 1442

Abstract. We study the task of image inpainting, which is to fill in the missing region of an incomplete image with plausible contents. To this end, we propose a learning-based approach to generate visually coherent completion given a high-resolution image with missing components. In order to overcome the difficulty to directly learn the distribution of high-dimensional image data, we divide the task into inference, translation as two separate steps and model each step with a deep neural network. We also use simple heuristics to guide matching of textures from boundary to the hole. We show that, by using such techniques, inpainting reduces to the problem of learning two image-feature translation functions of much smaller dimensionality. We evaluate our method on several public datasets and show that we not only generate results of comparable or better visual quality, but are orders of magnitude faster than previous state-of-the-art methods.

1 Introduction

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

Fig. 1. Our result comparing with GL inpainting [1]. (a) & (d) The input image with missing hole. (b) & (d) Inpainting result given by GL inpainting [1]. (c) & (f) Final inpainting result using our approach. The size of images are 512x512.

.

The problem of generating photo-realistic images from sampled noise or conditioning on other inputs such as images, texts or labels has been heavily investigated. In spite of recent progress of deep generative models such as PixelCNN [2], VAE [3] and GANs [4], generating high-resolution images remains a difficult task. This is mainly because modeling the distribution of pixels is difficult and the trained models easily introduce blurry components and artifacts when the dimensionality becomes high. Several approaches have been proposed to alleviate the problem, usually by leveraging multi-scale training [5, 6] or incorporating prior information [7].

Instead of tackling the general image synthesis problem, we are interested in the task of image inpainting. The task can be described as: given an incomplete image as input, how do we fill in the missing parts with semantically and visually plausible contents. It can also be interpreted as the problem of image synthesis conditioned on a set of known pixels. We are interested in this problem for several reasons. First, it is a well-motivated task. It is a common scenario where we may want to remove unwanted objects from pictures or videos, or we may want to restore damaged photographs. Second, while purely unsupervised learning may be challenging for large inputs, we show in this work that the problem becomes more constrained and tractable when we train in a multi-stage self-supervised manner and leverage the high-frequency information in the known region.

Context-encoder [8] is one of the first works that apply deep neural networks for image inpainting. It trains a deep generative model that maps an incomplete image to a complete image using reconstruction loss and adversarial loss. While combining adversarial loss significantly improves the inpainting quality, the results still lack high-frequency details and contain notable artifacts. In addition, we found it fails to train on larger inputs like 256x256 or 512x512. Hence it cannot generalize to the high-resolution inpainting task. More recently [1] improved the result by using dilated convolution and an additional local discriminator. However it is still limited to relatively small images and holes due to the spatial support of the model.

Yang *et al.* [9] proposes to use style transfer for image inpainting. More specifically, it initializes the hole with the output of context-encoder, and then improves the texture by using style transfer techniques [10] to propagate the high-frequency textures from the boundary to the hole. It shows that matching the neural features not only transfers artistic styles, but can also synthesize real-world images. The approach is optimization-based and applicable to images of arbitrary sizes. However, the computation is costly and it takes long time to inpaint a large image.

Our approach overcomes the limitation of the aforementioned methods. Being similar to [9], we decouple the inpainting process into two stages: inferrence and translation. In the inferrence stage, we train an *Image2Feature* network that initializes the hole with coarse prediction and extract its features. The prediction is blurry but contains high-level structure information in the hole. In the translation stage, we train a *Feature2Image* network that transforms the feature back into a complete image. It refines the contents in the hole and outputs a complete image with sharp and realistic texture. Its main difference with [9] is that, instead of relying on optimization, we model texture refinement as a learning problem. Both networks can be trained end-to-end and, with the trained models, the inference can be done in a single forward pass, which is much faster than iterative optimization.

To ease the difficulty of training the Feature2Image network, we design a "patch-swap" layer that propagates the high-frequency texture details from the boundary to the hole. The patch-swap layer takes the feature map as input, and replaces each neural patch inside the hole with the most similar patch on the boundary. We then use the new feature map as the input to the Feature2Image network. Presumably by re-using the neural patches on the boundary, the feature map contains sufficient details making the high-resolution image reconstruction feasible.

Our experience is that it is difficult to directly train a generative model for high-resolution inpainting. This might be because the space of mapping from an incomplete image to a complete image is overly large. We address this by reducing the dimensionality of either the input or the output. For the Image2Feature network, we only produce a blurry and coarse inpainting so that the output space is constrained. For the Feature2Image network, the space of input is curtailed when we initialize with the high-level prior from the Image2Feature network and the low-level prior from patch-swap. We observe that reducing the dimensionality of input or output space enables us to train both models much more easily at higher resolutions.

When being compared with the GL inpainting [1], we generate sharper and better inpainting results at size 256x256. Our approach also scales to larger resolution at 512x512, while GL inpainting cannot handle large hole generations. As compared with neural inpainting [9], our results have comparable or better visual quality in most examples. Especially our synthesized contents blends with the boundary more seamlessly. On top of that, our approach is much faster.

The main contributions of this paper are summarized as follows:

1. We design a learning-based inpainting system that is able to synthesize substantial missing parts in a high-resolution image with high-quality contents and textures.
2. We propose a novel and robust training scheme that addresses the issue of noisy input and avoids under-fitting.
3. We show that our trained model can be directly used on other tasks like style transfer and achieve performance comparable with state-of-the-art.

2 Related Work

Image synthesis using deep learning Using deep learning techniques for generative image modeling has gain remarkable progress recently. Based on how we model the density for image sampling, we can classify those methods into different categories. An important category is to *implicitly* model the density, predominantly based on the generative adversarial networks (GANs) [4]. The vanilla GANs has shown promising performance to generate sharper images, but training instability makes it hard to scale to higher resolution images. Several techniques have been proposed to stabilize the training process, including DC-GAN [11], energy-based GAN [12], Wasserstein GAN (WGAN) [13, 14], WGAN-GP [15], BEGAN [16], LSGAN [17] and the more recent Progressive GANs [18]. A complete list and benchmark of various GANs can be found at [19].

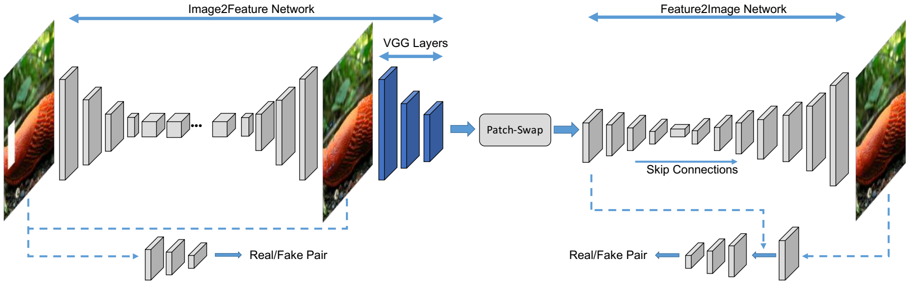


Fig. 2. Overview of our network architecture. We use Image2Feature network as coarse inferrence and use VGG network to extract a feature map. Then patch-swap matches neural patches from boundary to the hole. Finally the Feature2Image network translates to a complete, high-resolution image.

There are also plenty of literatures that address the problem of conditioning image generation, which is also more relevant to the inpainting task. For example, [5, 20] use GANs to generate images from texts. [21–23] study the problem of image super-resolution, which can be interpreted as image synthesis conditioning on low-resolution image. Related to our image-feature translation models are Pix2Pix [24] and CycleGAN [25] which translate images across different domains. Different from their work, our networks translate between feature space and image space. Finally using deep neural network for image inpainting has been studied in [26, 8, 9, 27, 1].

Neural style transfer Similar to [9], our method is based on recent works in neural style transfer. Gatys *et al.* [28] first formulates style transfer as an optimization problem that combines texture synthesis with content reconstruction. [29, 30, 2] alternatively use neural-patch based similarity matching between the content and style images. In particular, Li and Wand [10] optimize the output image such that each of its neural patch matches with a similar neural patch in the style image. This enables arbitrary style transfer at the cost of expensive computation. [31] proposes a fast approximation to [10] where it constructs the feature map directly and uses an inverse network to synthesize the image in feed-forward manner. However, we observe using off-the-shelf style transfer techniques for inpainting is not sufficient to generate high-quality result and we must propose new training principals and architectures to address the specific property of the inpainting problem.

Non-neural image inpainting Traditional non-neural inpainting algorithms [32, 33] mostly work on the image space. While they share similar ideas of patch matching and propagation, they are usually agnostic to high-level semantic and structural information. A complete comparison with non-neural inpainting algorithms is beyond the scope of our paper.

3 Methodology

3.1 Problem description

We formalize the task of image inpainting as follows: suppose we are given an incomplete input image I_0, with R and $\bar{R}$ representing the missing region (the hole) and the known region (the boundary) respectively. We would like to fill in R with plausible contents I_R and combine it with I_0 as a new, complete image I. Evaluating the quality of inpainting is mostly subject to human perception but ideally I_R should meet the following criteria: 1. It has sharp and realistic-looking textures; 2. It contains meaningful content and is coherent with $I_{\bar{R}}$ and 3. It looks like what appears in the ground truth image I_{gt} (if available). In our context, R can be either a single hole or multiple holes. It may also come with arbitrary shape, placed on random location of the image.

3.2 System Overview

Our system divides the image inpainting tasks into three steps:

1. **Inferrence:** We use an Image2Feature network to fill an incomplete image with coarse contents as inference and extract a feature map from the inpainted image.
2. **Matching:** We use patch-swap on the feature map to match the neural patches from the high-resolution boundary to the hole with coarse inference.
3. **Translation:** We use a Feature2Image network to translate the feature map to a complete image.

The entire pipeline is illustrated in Fig. 2.

3.3 Training

We introduce separate steps of training the Image2Feature and Feature2Image network. For illustration purpose we assume the size of I_0 is 256x256x3 and the hole R has size 128x128.

3.3.1 Inference: Training Image2Feature Network The goal of the Image2Feature network is to fill in the hole with coarse prediction. During training, the input to the Image2Feature translation network is the 256x256x3 incomplete image I_0 and the output is a feature map F_1 of size 64x64x256. The network consists of an FCN-based module G_1, which consists of an down-sampling front end, multiple intermediate residual blocks and an up-sampling back end. G_1 is followed by the initial layers of the 19-layer VGG network [34]. Here we use the filter pyramid of the VGG network as a higher-level representation of images similar to [28]. At first, I_0 is given as input to G_1 which produces a coarse prediction I_1^R of size 128x128. I_1^R is then embedded into R forming a complete image I_1, which again passes through the VGG19 network to get the activation of *relu3_1* as F_1. F_1 has size 64x64x256. We also use an additional PatchGAN discriminator D_1 to facilitate adversarial training, which takes a pair of images as input, and outputs a vector of true/fake probabilities.

For G_1, the down-sampling front-end consists of three convolutional layers, and each layer has stride 2. The intermediate part has 9 residual blocks stacked together. The up-sampling back-end is the reverse of the front-end and consists of three transposed convolution with stride 2. Every convolutional layer is followed by batch normalization (BN) and ReLu activation, except for the last layer which outputs the image. We also use dilated convolution in all residual blocks. Similar architecture has been used in [35] for image synthesis and [1] for inpainting. Different from [35], we use dilated layer to increase the size of receptive field. Comparing with [1], our receptive field is also larger given we have more down-sampling blocks and more dilated layers in residual blocks.

During training, the overall loss function is defined as:

$$L_{G_1} = \lambda_1 L_{perceptual} + \lambda_2 L_{adv}. \tag{1}$$

The first term is the perceptual loss, which is shown to correspond better with human perception of similarity [36] and has been widely used in many tasks [37–39, 31]:

$$L_{perceptual}(F, I_g t) = \parallel \mathcal{M}_F \circ (F_1 - vgg(I_{gt})) \parallel_1 . \tag{2}$$

Here $\mathcal{M}_F$ are the weighted masks yielding the loss to be computed only on the hole r. We also assign higher weight to the overlapping pixels between the hole and the boundary to ensure the composite is coherent. The weights of VGG19 network are loaded from the ImageNet pre-trained model and are fixed during training.

The adversarial loss is based on Generative Adversarial Networks (GANs) and is defined as:

$$L_{adv} = \max_{D_1} E[\log(D_1(I_0, I_{gt})) + \log(1 - D_1(I_0, I_1))]. \tag{3}$$

We use a pair of images as input to the discriminator. Under the setting of adversarial training, the real pair is the incomplete image I_0 and the original image I_{gt}, while the fake pair is I_0 and the prediction I_1.

To align the absolute value of each loss, we set the weight $\lambda_1 = 10$ and $\lambda_2 = 1$ respectively. We use Adam optimizer for training. The learning rate is set as $lr_G = 2\text{e}{-3}$ and $lr_D = 2\text{e}{-4}$ and the momentum is set to 0.5.

3.3.2 Match: Patch-swap operation Patch-swap is an operation which transforms F_1 into a new feature map F_1'. The idea is that the prediction I_1^R is blurry, lacking many of the high-frequency details. Intuitively, we would like to propagate the textures from $I_1^{\bar{R}}$ onto I_1^R but still preserves the high-level information of I_1^R. Instead of operating on I_1 directly, we use F_1 as a surrogate for texture propagation. Similarly, we use r and $\bar{r}$ to denote the region on F_1

corresponding to R and $\bar{R}$ on I_1. For each 3x3 neural patch $p_i(i = 1, 2, ..., N)$ of F_1 overlapping with r, we find the closest-matching neural patch in $\bar{r}$ based on the following cross-correlation metric:

$$d(p, p') = \frac{< p, p' >}{\| p \| \cdot \| p' \|}$$

Suppose the closest-matching patch of p_i is q_i, we then replace p_i with q_i. After each patch in r is swapped with its most similar patch in $\bar{r}$, overlapping patches are averaged and the output is a new feature map F_1'. We illustrate the process in Fig. 3.

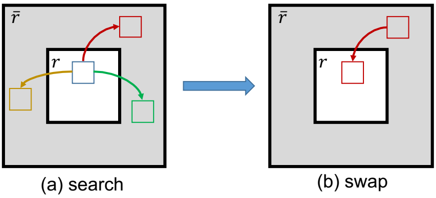


Fig. 3. Illustration of patch-swap operation. Each neural patch in the hole r searches for the most similar neural patch on the boundary $\bar{r}$, and then swaps with that patch.

Measuring the cross-correlations for all the neural patch pairs between the hole and boundary is computationally expensive. To address this issue, we follow similar implementation in [31] and speed up the computation using paralleled convolution. We summarize the algorithm as following steps. First, we normalize and stack the neural patches on $\bar{r}$ and view the stacked vector as a convolution filter. Next we apply the convolution filter on F_r. The result is that at each location of r we get a vector of values which is the cross-correlation between the neural patch centered at that location and all patches in $\bar{r}$. Finally we replace the patch in r with the patch in $\bar{r}$ of maximum cross-correlation. Since the whole process can be parallelized, the amount of time is significantly reduced. In practice it only takes about 0.1 seconds to process a 64x64x256 feature map.

3.3.3 Translate: Training Feature2Image translation network The goal of the Feature2Image network is to learn a mapping from the swapped feature map to a complete and sharp image. It has a U-Net style generator G_2 which is similar to G_1, except the number of hidden layers are different. The input to G_2 is a feature map of size 64x64x256. The generator has seven convolution blocks and eight deconvolution blocks, and the first six deconvolutional layers are connected with the convolutional layers using skip connection. The output is a complete 256x256x3 image. It also consists of a Patch-GAN based discriminator D_2 for adversarial training. However different from the Image2Feature network which takes a pair of images as input, the input to D_2 is a pair of image and feature map.

A straightforward training paradigm is to use the output of the Image2Feature network F_1 as input to the patch-swap layer, and then use the swapped feature F'_1 to train the Feature2Image model. In this way, the feature map is derived from the coarse prediction I_1 and the whole system can be trained end-to-end. However in practice, we found that this leads to poor-quality reconstruction I with notable noise and artifacts (Sec. 4). We further observed that using the ground truth as training input gives rise to results of significantly improved visual quality. That is, we use the feature map $F_{gt} = \text{vgg}(I_{gt})$ as input to the patch-swap layer, and then use the swapped feature $F'_{gt} = \text{patch\_swap}(F_{gt})$ to train the Feature2Image model. Since I_{gt} is not accessible at test time, we still use $F'_1 = \text{patch\_swap}(F_1)$ as input for inference. Note that now the Feature2Image model trains and tests with different types of input, which is not a usual practice to train a machine learning model.

Here we provide some intuition for this phenomenon. Essentially by training the Feature2Image network, we are learning a mapping from the feature space to the image space. Since F_1 is the output of the Image2Feature network, it inherently contains a significant amount of noise and ambiguity. Therefore the feature space made up of F'_1 has much higher dimensionality than the feature space made up of F'_{gt}. The outcome is that the model easily under-fits F'_1, making it difficult to learn a good mapping. Alternatively by using F'_{gt}, we are selecting a clean, compact subset of features such that the space of mapping is much smaller, making it easier to learn. Our experiment also shows that the model trained with ground truth generalizes well to noisy input F'_1 at test time. Similar to [40], we can further improve the robustness by sampling from both the ground truth and Image2Feature prediction.

The overall loss function for the Feature2Image translation network is defined as:

$$L_{G_2} = \lambda_1 L_{perceptual} + \lambda_2 L_{adv}. \tag{4}$$

The reconstruction loss is defined on the entire image between the final output I and the ground truth I_{gt}:

$$L_{perceptual}(I, I_{gt}) = \| vgg(I) - vgg(I_{gt}) \|_2 . \tag{5}$$

The adversarial loss is given by the discriminator D_2 and is defined as:

$$L_{adv} = \max_{D_2} E[\log(D_2(F'_{gt}, I_{gt})) + \log(1 - D_2(F'_{gt}, I))]. \tag{6}$$

The real and fake pair for adversarial training are (F'_{gt}, I_{gt}) and (F'_{gt}, I).

When training the Feature2Image network we set $\lambda_1 = 10$ and $\lambda_2 = 1$. For the learning rate, we set $lr_G = 2\text{e}{-4}$ and $lr_D = 2\text{e}{-4}$. Same as the Image2Feature network, the momentum is set to 0.5.

3.4 Multi-scale Inference

Given the trained models, inference is straight-forward and can be done in a single forward pass. The input I_0 successively passes through the Image2Feature network to get I_1 and $F_1 = \text{vgg}(I_1)$, then the patch-swap layer (F_1'), and then finally the Feature2Image network (I). We then use the center of I and blend with I_0 as the output.

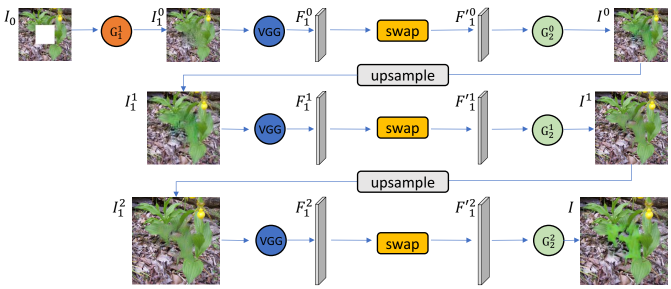


Fig. 4. Illustration of multi-scale inference.

Our framework can be easily adapted to multi-scale. The key is that we directly upsample the output of the lower scale as the input to the Feature2Image network of the next scale (after using VGG network to extract features and apply patch-swap). In this way, we will only need the Image2Feature network at the smallest scale s_0 to get I_1^0 and F_1^0. At higher scales $s_i (i > 0)$ we simply set $I_1^{s_i} = \text{upsample}(I^{s_{i-1}})$ and let $F_1^{s_i} = \text{vgg}(I_1^{s_i})$ (Fig. 4). Training Image2Feature network can be challenging at high resolution. However by using multi-scale approach we are able to initialize from lower scales instead, allowing us to handle large inputs effectively. We use multi-scale inference on all our experiments.

4 Experiments

4.1 Experiment Setup

We separately train and test on two public datasets: COCO [41] and ImageNet CLS-LOC [42]. The number of training images in each dataset are: 118,287 for COCO and 1,281,167 for ImageNet CLS-LOC. We compare with content aware fill (CAF) [32], context encoder (CE) [8], neural patch synthesis (NPS) [9] and global local inpainting (GLI) [1]. For CE, NPS and GLI, we used the public available trained model. CE and NPS are trained to handle fixed holes, while GLI and CAF can handle arbitrary holes. To fairly evaluate, we experimented on both settings of fixed hole and random hole. For fixed hole, we compare with CAF, CE, NPS and GLI on image size 512x512 from ImageNet test set. The hole is set to be 224x224 located at the image center. For random hole, we compare with CAF and GLI, using COCO test images resized to 256x256. In the case of random hole, the hole size ranges from 32 to 128 and is placed anywhere on the image. We observed that for small holes on 256x256 images, using patch-swap and Feature2Image network to refine is optional as our Image2Feature network already generates satisfying results most of the time. While for 512x512 images, it is necessary to apply multi-scale inpainting, starting from size 256x256. To address both sizes and to apply multi-scale, we train the Image2Feature network at 256x256 and train the Feature2Image network at both 256x256 and 512x512. During training, we use early stopping, meaning we terminate the training when the loss on the held-out validation set converges. On our NVIDIA GeForce GTX 1080Ti GPU, training typically takes one day to finish for each model.

4.2 Results

Quantitative comparison Table 1 shows numerical comparison result between our approach, CE [8], GLI [1] and NPS [9]. We adopt two quality measurements: mean ℓ_1 error and SSIM. Since context encoder only inpaints 128x128 images and we failed to train the model for larger inputs, we directly use the 128x128 results and bilinearly upsample them to 512x512. We see that although our mean ℓ_1 error is higher, we achieve best SSIM among all the methods, showing our results are closer to ground truth by human perception. Besides, mean ℓ_1 error is not an optimal measure for inpainting, as it favors averaged colors and blurry results and does not directly account for the end goal of perceptual quality.

Method	**Mean ℓ_1 Error**	**SSIM**
CE [8]	15.46%	0.87
NPS [9]	**15.13%**	0.88
GLI [1]	15.81%	**0.89**
our approach	15.61%	**0.89**

Table 1. Numerical comparison on 200 test images of ImageNet.

Visual result Fig. 9 shows our comparison with GLI [1] in random hole cases. We can see that our method could handle multiple situations better, such as object removal, object completion and texture generation, while GLIs results are noisier and less coherent. From Fig. 10, we could also find that our results are better than GLI most of the time for large holes. This shows that directly training a network for large hole inpainting is difficult, and it is where our "patch-swap" can be most helpful. In addition, our results have significantly less artifacts than GLI. Comparing with CAF, we can better predict the global structure and fill in contents more coherent with the surrounding context. Comparing with CE, we can handle much larger images and the synthesized contents are much sharper. Comparing with NPS whose results mostly depend on CE, we have similar or better quality most of the time, and our algorithm also runs much faster. Meanwhile, our final results improve over the intermediate output of Image2Feature. This demonstrates that using patch-swap and Feature2Image transformation is beneficial and necessary.

User study To better evaluate and compare with other methods, we perform a user study by seeking feedback from 20 users. We give each user a set of 20 questions. In each question the user is presented with an incomplete image with center fixed hole and is asked to compare the inpainting result of NPS, GLI and ours. We found our results are ranked best most of the time: in 75.9% of the rankings our result receives highest score. In particular, our results are overwhelmingly better than GLI, receiving higher score 91.2% of the time. This is largely because GLI does not handle large holes well. Our results are also comparable with NPS, ranking higher or the same 86.2% of the time.

4.3 Analysis

Comparison Comparing with [9], not only our approach is much faster but also has several advantages. First, the Feature2Image network synthesizes the entire image while [9] only optimizes the hole part. By aligning the color of the boundary between the output and the input, we can slightly adjust the tone to make the hole blend with the boundary more seamlessly and naturally (Fig. 10). Second, our model is trained to directly model the statistics of real-world images and works well on all resolutions, while [9] is unable to produce sharp results when the image is small. Comparing with other learning-based inpainting methods, our approach is more general as we can handle larger inputs like 512x512. In contrast, [8] can only inpaint 128x128 images while [1] is limited to 256x256 images and the holes are limited to be smaller than 128x128.

Ablation study For the Feature2Image network, we observed that replacing the deconvolutional layers in the decoder part with resize-convolution layers resolves the checkerboard patterns as described in [43] (Fig. 5 left). We also tried only using ℓ_2 loss instead of perceptual loss, which gives blurrier inpainting (Fig. 5 middle). Additionally, we experimented different activation layers of VGG19 to extract features and found that *relu3_1* works better than *relu2_1* and *relu4_1*.

We may also use iterative inference by running Feature2Image network multiple times. At each iteration the final output is used as input to VGG and patch-swap, and then again given to Feature2Image network for inference. We found iteratively applying Feature2Image improves the sharpness of the texture but sometimes aggregates the artifacts near the boundary.

For the Image2Feature network, an alternative is to use vanilla context encoder [8] to generate I_0^0 as initial inference. However we found our model produces better results as it is much deeper, and leverages the fully convolutional network and dilated layer.

As discussed in Sec. 3.3, an important practice to guarantee successful training of the Feature2Image network is to use ground truth image as input rather than using the output of the Image2Feature network. Fig. 5 also shows that training with the prediction from the Image2Feature network gives very noisy results, while the models trained with ground truth or further fine-tuned with ground-truth and prediction mixtures can produce satisfying inpainting.

Our framework can be easily applied to real-world tasks. Fig. 6 shows examples of using our approach to remove unwanted objects in photography. Given our network is fully convolutional, it is straight-forward to apply it to photos of arbitrary sizes. It is also able to fill in holes of arbitrary shapes, and can handle much larger holes than [44].

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

Fig. 5. Left: using deconvolution (a) vs resize-convolution (b). Middle: using ℓ_2 reconstruction loss (c) vs using perceptual loss (d). Right: Training Feature2Image network using different input data. (e) Result when trained with the Image2Feature prediction. (f) Result when trained with ground truth. (g) Result when fine-tuned with ground truth and prediction mixtures.

The Feature2Image network essentially learns a universal function to reconstruct an image from a swapped feature map, therefore can also be applied on other tasks. For example by first constructing a swapped feature map from a content and a style image, we can use the network to reconstruct a new image for style transfer. Fig. 7 shows examples of using our Feature2Image network trained on COCO towards arbitrary style transfer. Although the network is agnostic to the styles being transfered, it is still capable of generating satisfying results and runs in real-time. This shows the strong generalization ability of our learned model, as it's only trained on a single COCO dataset unlike other style transfer methods.

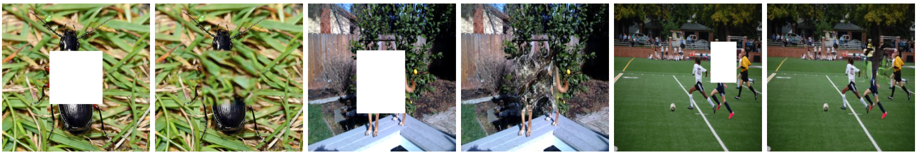
(a) (b) (c) (d) (e) (f)

Fig. 6. Arbitrary shape inpainting of real-world photography. (a) & (d) Input. (b) & (e) Inpainting mask. (c) & (f) Output.

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

Fig. 7. Arbitrary style transfer. (a)(d) Content. (b)(e) Style. (c)(f) Style transfer result.

Our approach is very good at recovering a partially missing object like a plane or a lamp post. However, it can fail if the image has overly complicated structures and patterns, or a major part of an object is missing such that Image2Feature network is unable to provide a good inference (Fig. 8).



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

Fig. 8. Failure cases. (a)(c)(e) Input. (b)(d)(f) Output.

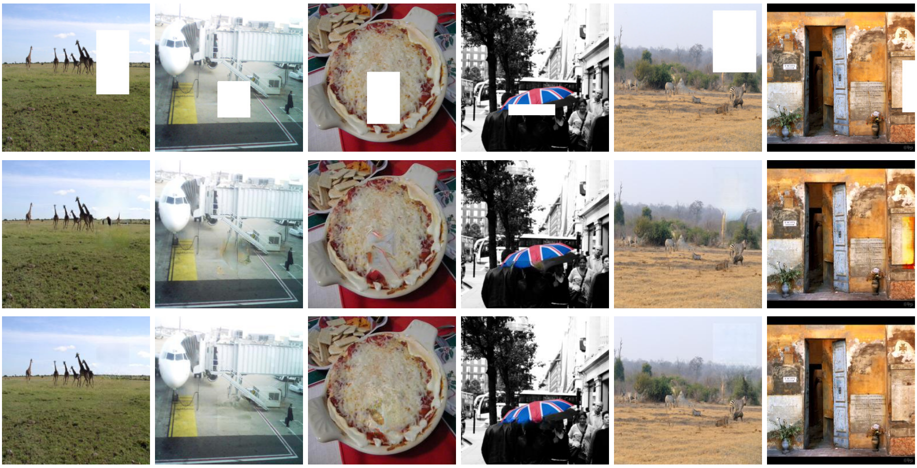


Fig. 9. Visual comparisons of ImageNet results with random hole. Each example from top to bottom: input image, GLI [1], our result. All images have size 256×256.

5 Conclusion

We propose a learning-based approach to synthesize missing contents in a high-resolution image. Our model is able to inpaint an image with realistic and sharp contents in a feed-forward manner. We show that we can simplify training by breaking down the task into multiple stages, where the mapping function in each stage has smaller dimensionality. It is worth noting that our approach is a meta algorithm and naturally we could explore a variety of network architectures and training techniques to improve the inference and the final result. We also expect that similar idea of multi-stage, multi-scale training could be used to directly synthesize high-resolution images from sampling.

Fig. 10. Visual comparisons of ImageNet and COCO results. Each example from left to right: input image, CAF [32], CE [8],NPS [9], GLI [1], our result w/o Feature2Image, our final result. All images have size 512×512.

References

1. Iizuka, S., Simo-Serra, E., Ishikawa, H.: Globally and Locally Consistent Image Completion. ACM Transactions on Graphics (Proc. of SIGGRAPH 2017) **36**(4) (2017) 107:1–107:14
2. van den Oord, A., Kalchbrenner, N., Espeholt, L., Vinyals, O., Graves, A., et al.: Conditional image generation with pixelcnn decoders. In: Advances in Neural Information Processing Systems. (2016) 4790–4798
3. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114 (2013)
4. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: Advances in neural information processing systems. (2014) 2672–2680
5. Zhang, H., Xu, T., Li, H., Zhang, S., Huang, X., Wang, X., Metaxas, D.: Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. arXiv preprint arXiv:1612.03242 (2016)
6. Denton, E.L., Chintala, S., Fergus, R., et al.: Deep generative image models using a laplacian pyramid of adversarial networks. In: Advances in neural information processing systems. (2015) 1486–1494
7. Nguyen, A., Yosinski, J., Bengio, Y., Dosovitskiy, A., Clune, J.: Plug & play generative networks: Conditional iterative generation of images in latent space. arXiv preprint arXiv:1612.00005 (2016)
8. 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
9. 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). (July 2017)
10. Li, C., Wand, M.: Combining markov random fields and convolutional neural networks for image synthesis. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2016) 2479–2486
11. Radford, A., Metz, L., Chintala, S.: Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint arXiv:1511.06434 (2015)
12. Zhao, J., Mathieu, M., LeCun, Y.: Energy-based generative adversarial network. arXiv preprint arXiv:1609.03126 (2016)
13. Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., Chen, X.: Improved techniques for training gans. In: Advances in Neural Information Processing Systems. (2016) 2234–2242
14. Arjovsky, M., Chintala, S., Bottou, L.: Wasserstein gan. arXiv preprint arXiv:1701.07875 (2017)
15. Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., Courville, A.: Improved training of wasserstein gans. arXiv preprint arXiv:1704.00028 (2017)
16. Berthelot, D., Schumm, T., Metz, L.: Began: Boundary equilibrium generative adversarial networks. arXiv preprint arXiv:1703.10717 (2017)
17. Mao, X., Li, Q., Xie, H., Lau, R.Y., Wang, Z., Smolley, S.P.: Least squares generative adversarial networks. arXiv preprint ArXiv:1611.04076 (2016)
18. Karras, T., Aila, T., Laine, S., Lehtinen, J.: Progressive growing of gans for improved quality, stability, and variation. arXiv preprint arXiv:1710.10196 (2017)

19. : Gans comparison without cherry-picking. https://github.com/khanrc/tf.gans-comparison Accessed: 2017-10-29.
20. Reed, S., Akata, Z., Yan, X., Logeswaran, L., Schiele, B., Lee, H.: Generative adversarial text to image synthesis. arXiv preprint arXiv:1605.05396 (2016)
21. Kim, J., Kwon Lee, J., Mu Lee, K.: Accurate image super-resolution using very deep convolutional networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2016) 1646–1654
22. Dong, C., Loy, C.C., He, K., Tang, X.: Learning a deep convolutional network for image super-resolution. In: European Conference on Computer Vision, Springer (2014) 184–199
23. Ledig, C., Theis, L., Huszár, F., Caballero, J., Cunningham, A., Acosta, A., Aitken, A., Tejani, A., Totz, J., Wang, Z., et al.: Photo-realistic single image super-resolution using a generative adversarial network. arXiv preprint arXiv:1609.04802 (2016)
24. Isola, P., Zhu, J.Y., Zhou, T., Efros, A.A.: Image-to-image translation with conditional adversarial networks. arXiv preprint arXiv:1611.07004 (2016)
25. Zhu, J.Y., Park, T., Isola, P., Efros, A.A.: Unpaired image-to-image translation using cycle-consistent adversarial networks. arXiv preprint arXiv:1703.10593 (2017)
26. Yeh, R.A., Chen, C., Lim, T.Y., Schwing, A.G., Hasegawa-Johnson, M., Do, M.N.: Semantic image inpainting with deep generative models. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2017) 5485–5493
27. Wang, W., Huang, Q., You, S., Yang, C., Neumann, U.: Shape inpainting using 3d generative adversarial network and recurrent convolutional networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2017) 2298–2306
28. Gatys, L.A., Ecker, A.S., Bethge, M.: A neural algorithm of artistic style. arXiv preprint arXiv:1508.06576 (2015)
29. Elad, M., Milanfar, P.: Style transfer via texture synthesis. IEEE Transactions on Image Processing **26**(5) (2017) 2338–2351
30. Frigo, O., Sabater, N., Delon, J., Hellier, P.: Split and match: Example-based adaptive patch sampling for unsupervised style transfer. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2016) 553–561
31. Chen, T.Q., Schmidt, M.: Fast patch-based style transfer of arbitrary style. arXiv preprint arXiv:1612.04337 (2016)
32. Barnes, C., Shechtman, E., Finkelstein, A., Goldman, D.B.: Patchmatch: A randomized correspondence algorithm for structural image editing. ACM Trans. Graph. **28**(3) (2009) 24–1
33. Barnes, C., Shechtman, E., Goldman, D.B., Finkelstein, A.: The generalized patchmatch correspondence algorithm. In: European Conference on Computer Vision, Springer (2010) 29–43
34. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
35. 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)
36. 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)
37. 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

38. 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
39. Dosovitskiy, A., Brox, T.: Generating images with perceptual similarity metrics based on deep networks. In: Advances in Neural Information Processing Systems. (2016) 658–666
40. Zheng, S., Song, Y., Leung, T., Goodfellow, I.: Improving the robustness of deep neural networks via stability training. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2016) 4480–4488
41. 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
42. 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
43. Odena, A., Dumoulin, V., Olah, C.: Deconvolution and checkerboard artifacts. Distill (2016)
44. Iizuka, S., Simo-Serra, E., Ishikawa, H.: Globally and locally consistent image completion. ACM Transactions on Graphics (TOG) **36**(4) (2017) 107