This ACCV 2020 paper, provided here by the Computer Vision Foundation, is the author-created version.
The content of this paper is identical to the content of the officially published ACCV 2020
LNCS version of the paper as available on SpringerLink: https://link.springer.com/conference/accv

Image Inpainting with Onion Convolutions

Shant Navasardyan[1][0000−0002−1999−9999] and Marianna Ohanyan[2][0000−0002−4815−3802]

[1] Picsart Inc., Armenia, Yerevan shant.navasardyan@picsart.com
[2] Picsart Inc., Armenia, Yerevan marianna.ohanyan@picsart.com

Fig. 1. Image inpainting results of our method. Each triplet of images is composed of the original image, the image with the missing region, and our result. The method allows users to remove unwanted objects or fill missing parts in images. Please see in color.

Abstract. Recently deep learning methods have achieved a great success in image inpainting problem. However, reconstructing continuities of complex structures with non-stationary textures remains a challenging task for computer vision. In this paper, a novel approach to image inpainting problem is presented, which adapts exemplar-based methods for deep convolutional neural networks. The concept of *onion convolution* is introduced with the purpose of preserving feature continuities and semantic coherence. Similar to recent approaches, our onion convolution is able to capture *long-range spatial correlations*. In general, the implementation of modules with such ability in low-level features leads to impractically high latency and complexity. To address this limitations, the onion convolution suggests an efficient implementation. As qualitative and quantitative comparisons show, our method with onion convolutions outperforms state-of-the-art methods by producing more realistic, visually plausible and semantically coherent results.

Keywords: Inpainting · Onion Convolution · Patch-Match

1 Introduction

Image inpainting is the problem of completing missing or damaged regions in images resulting in a realistic, visually plausible, and semantically meaningful output. It can be utilized in many applications such as recovering spots or other damaged parts in images, removing unwanted objects or parts of them.

The main challenge of obtaining a high-quality image inpainting algorithm is getting both semantically consistent and visually realistic results, especially if the image contains complex scenes and structures.

Some traditional approaches [1–8] are based on texture synthesis techniques and achieve great success in producing realistic-looking texture details. However, these methods rarely give structurally or semantically reasonable outputs.

Later, some methods [9–11] were proposed to address the structural inconsistencies resulting in perfect image completions in some cases. Yet, generating coherent semantics remains beyond the abilities of traditional methods. Hence, deep learning techniques come to fill the missing region in a semantically plausible way [12–16]. Early methods [17, 12, 18, 13] use vanilla convolutions and manage to gain fine results for small missing regions. However, for complex structures, these methods introduce blur, edge artifacts, and color inconsistencies. The reason is that vanilla convolutions treat all pixels equally, so outputs depend on the initialization of the missing pixels.

Some methods [19, 14, 15, 20] introduce special convolutional layers which are designed to operate with only valid pixels in the input. This approach allows to gain a drastic improvement over using only vanilla convolutions. Though, these mechanisms introduce some patterns or distorted structures near the boundary of the missing region due to ineffectiveness of convolutions in modeling long-range pixel correlations.

To address this problem, two main approaches were introduced. The first approach [21–24] adopts patch-based traditional image inpainting techniques in the learning process, while the second one [25, 15, 16] utilizes the self-attention mechanism [26]. Both approaches perform state-of-the-art results by completing complex structures and semantics. However, in all these methods, the modules responsible for capturing long-range dependencies are designed to process tensors in which the initial missing region is roughly filled. So the output of these methods highly depends on the *coarse estimation* of the missing pixels. In some cases, networks fail to make a continuity-preserving coarse estimations, resulting in outputs with structure discontinuities (see Fig. 4).

To address this problem we introduce the onion convolution layer, which is able to preserve feature continuities and does not depend on a coarse estimation of the missing region. To make the layer keep feature continuities, we supply it with an iterative filling of the missing region starting from its boundary to the center, similar to the process of peeling an onion, so we call it *onion convolution*. Some results can be found in Fig. 1.

In general, modules capturing long-range pixel dependencies come with a high computational cost, since their implementations often require (all-to-all) pixel-wise dot-product computations. This makes the usage of such modules

impractical in high resolution features. On the other hand, in the case of image inpainting, some of the computations mentioned above can be avoided. Such adaptation of these modules to the image inpainting problem was introduced in [25] for *rectangular-formed* missing regions. However it still does unnecessary computations when the missing region has an *irregular* form. To eliminate the redundant computational cost we make our own implementation with a common deep learning framework *TensorFlow* [27].

To summarize, our main contributions are the following:

- We introduce the *onion convolution* mechanism, which enables us to continuously propagate feature activations from valid pixels to the missing ones.
- To the best of our knowledge, this is the first work on using exemplar-based image inpainting techniques in deep features without coarsely estimating the pixels in the missing region.
- Onion convolution mechanism, which can capture long-range pixel dependencies, is implemented in a low computational and memory-efficient way.

The structure of the further part of this work is the following: in Section 2 some existing approaches to the image inpainting problem are reviewed. Then in Section 3 our approach is introduced in detail. In Section 4 our experiments, quantitative and qualitative comparisons with state-of-the-art methods, and the ablation study are presented. In Section 5 we make a conclusion.

2 Related Work

The existing approaches to the problem of image inpainting can be roughly divided into two groups. First group uses traditional diffusion-based or exemplar-based techniques, while the second one utilizes learning processes. In this section, a brief introduction of the mentioned groups will be held.

2.1 Traditional Non-Learning Methods

Historically, image inpainting was a problem of image restoration, with the aim of removing scratches, torn parts, or text in images. Several algorithms [28, 29] were suggested for this purpose, and rely on the idea of propagating image structures from the known region to the unknown by diffusion processes. Diffusion based approaches are able to produce high quality results when restoring small damaged parts in images. However, they sometimes over-smooth outputs, especially in cases when the missing region is large. To address this problem, exemplar-based methods [1–8, 30–32] replicate the texture by copying patches from the known region. In [1], authors apply a patch-based technique to sample from the conditional distribution of each pixel, conditioned on its neighborhood. Later some methods [3–5] were proposed to optimize the algorithm introduced in [1]. Also, the *randomized search*, introduced in [2], can be adopted for this purpose. However, the method described in [1] completes a *single pixel at a time*, which can lead some textures to "occasionally 'slip' into a wrong part of the

search space and start growing garbage" (as mentioned in limitations in [1]). To address the problem of "growing garbage", some *texture synthesis* algorithms suggest copying regions with more than one pixel at a time [6, 30–32].

Though these methods perform excellent results in providing detailed textures, they fail in reconstructing complex structures or semantics.

2.2 Learning-Based Methods

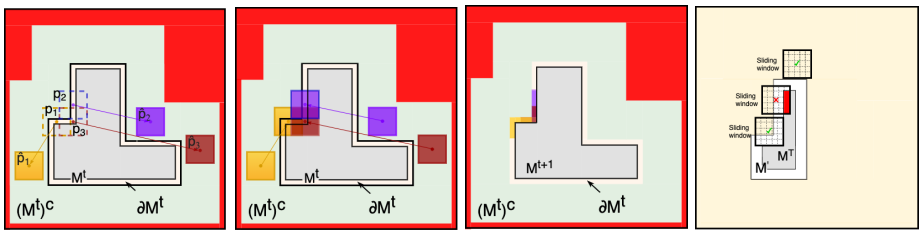
After computer vision adopted deep learning techniques, researchers started experimenting with convolutional neural networks in the image inpainting problem. Deep learning allows modeling of high-level semantics, which is used to generate missing parts of objects based on the semantic information of images. Early works on image inpainting with neural networks [17, 33, 19] concentrate on completing locally corrupted images when missing regions are small. Later, for obtaining more structure-preserving results, some learning-based methods [34–36] have adopted structure-guidance techniques inside networks. However, some of the methods mentioned above poorly generate texture details.

Through *generative adversarial networks*, *GANs* [37], researchers have reached new heights in solving computer vision tasks. The ability of GANs to generate visually plausible results helps to coherently hallucinate the missing region [12, 13, 18, 24, 15]. Nevertheless, in some cases boundary artifacts and inconsistencies occur. This is caused mostly by treating all pixels equally when processing an incomplete image trough stacked vanilla convolutions. Hence, the methods with *partial convolutions* [14] and *gated convolutions* [15] are introduced. The partial convolution [14] convolves with only known pixels at each sliding window and normalize the output by the number of these known pixels. Moreover, it narrows the missing region in a rule-based manner. In contrast, the gated convolution [15] lets the network to learn how to update the missing region for each layer. Both works perform excellent results on many complicated examples.

Despite the high quality of the algorithms mentioned above, some of them fail to complete complex structures continuously. The main reason is the inability to capture long-range spatial dependencies. To eliminate this cause, some methods either adopt exemplar-based techniques [21–24] or benefit from self-attention mechanisms [25, 15, 16]. In [23] the authors use the technique described in [38], which replace the patches in the unknown region with their closest (in terms of similarity defined by cross-correlation) patches from the known region. Similarly, [22] uses a special case of this technique, by considering patches of size 1×1 (i.e. pixels) and combining features from both encoder and decoder for computing similarities. [25, 15] introduce the *contextual attention* layer for completing the missing region based on the soft self-attention mechanism [39, 26]. These works obtain great results, outperforming previous state-of-the-art methods. Though, their *long-range spatial dependency capturing* modules need to be fed with a tensor in which the missing region is coarsely initialized. This may cause structure discontinuities due to unreasonable coarse estimations. In this work the onion convolution layer is introduced to continuously propagate the information without any coarse estimation of the missing region.

3 Approach

In this section, all components of our approach are described in detail. First, we introduce our *onion convolution* mechanism and discuss some implementations. Then the inpainting network architecture and loss functions are presented.



(a) *(b)* *(c)* *(d)*

Fig. 2. Onion convolution. First we perform onion-peel patch-match in the following way: for each iteration $t = 1, \ldots, T$ the pixels in the boundary ∂M^t of the missing region M^t are considered. (*a*) $k_m \times k_m$ patches, centered in the boundary pixels (e.g. p_1, p_2, p_3), are matched to their corresponding patches from the source region $(M^t)^c$ (corresponding source patches are centered in $\hat{p}_1, \hat{p}_2, \hat{p}_3$). (*b*) $k_f \times k_f$ patches centered in the pixels $p_1, p_2, p_3, \ldots$ are replaced with their corresponding $k_f \times k_f$ patches centered in $\hat{p}_1, \hat{p}_2, \hat{p}_3, \ldots$. (*c*) Then the overlapping parts are aggregated by averaging, and the next missing region M^{t+1} is computed. After the onion-peel patch-match, a convolution followed by an updating of the initial missing region $M^1 = M$ is applied. (*d*) Some $k_c \times k_c$ convolution sliding windows, centered in the filled region $(1 - M^T)$, may overlap with the missing region M^T, hence their centers are also treated as non-valid pixels, resulting in the updating of the initial missing region $M \mapsto M'$ by the Eq. 8.

3.1 Onion Convolution

During our research, it has been validated that for getting semantically meaningful and visually realistic results, our network needs to contain a block which satisfies the following conditions:

- $(C1)$ Valid/known pixels has higher impact on the output of the block than missing ones.
- $(C2)$ The block continuously propagates the information (e.g. texture, structure, semantics) of the know region to the unknown.

All deep-learning-based methods that use only vanilla convolutions do not satisfy the condition $(C1)$, since they treat all pixels as valid ones.

Though the *partial* [14] and *gated* [15] convolutions are designed to satisfy the condition $(C1)$, they are unable to model long-range pixel correlations, which is necessary for satisfying the condition $(C2)$. As we have already mentioned, some methods [22, 23, 15] capture long-range pixel dependencies, but do not pay

a special attention on *continuously propagation of the information*, so they also do not satisfy $(C2)$.

To make both $(C1)$ and $(C2)$ hold, we propose the *onion convolution layer*, illustrated in Fig. 2. It takes two arguments as input: a tensor $X \in \mathbb{R}^{H\times W\times c}$ and a binary mask $M \in \{0,1\}^{H\times W}$ indicating the missing region in the tensor X ($M_{ij} = 1$ means, that the pixel $X_{ij} \in \mathbb{R}^c$ lies in the missing region). The onion convolution layer is composed of three stages: *onion-peel patch-match*, *convolution* and *updating the missing region*. Below we describe each stage in detail.

Onion-Peel Patch-Match As the name prompts, the *onion-peel patch-match* is an algorithm, which deals with *patches*. Let $\mathcal{T} \in \mathbb{R}^{H\times W\times c}$ be a tensor, k be a positive integer and $p = (i, j)$ be a position of a pixel in $\mathcal{T}$ ($1 \le i \le H,\ 1 \le j \le W$). Then by $patch^k_{\mathcal{T}}(p)$ we denote the $k \times k$ patch in $\mathcal{T}$ centered at p.

With a usage of concepts introduced in [1], our onion-peel patch-match aims to preserve feature continuities when filling the missing region. Illustration of the approach is shown in Fig. 2 *(a), (b), (c)*.

The missing region in X is filled iteratively, initially taking $X^1 = X, M^1 = M$. For each iteration $t = 1, \dots, T$, the boundary of the missing region M^t is filled, resulting in a tensor X^{t+1} with a missing region, indicated by M^{t+1}. So at first we need to identify the boundary of the missing region. Since the missing region is indicated by M^t, the boundary can be obtained by the morphological erosion operation on M^t with a window of size 3×3:

$$\partial M^t = M^t - erode(M^t, 3)\ . \tag{1}$$

Then for each point p in the boundary ∂M^t of the missing region we fill the unknown pixel X^t_p by sampling from its conditional distribution $P(X^t_p \mid X^t_{n(p)})$ given its known neighborhood $X^t_{n(p)}$.

In order to estimate $P(X^t_p \mid X^t_{n(p)})$, similar with [1], we consider the histogram of pixels from a set $\Omega^\varepsilon(p)$. The set $\Omega^\varepsilon(p)$ is composed of pixels, neighborhoods of which are close to the neighborhood of p. More precisely, we consider $patch^{k_m}_{X^t}(\hat{p})$ as a neighborhood of a pixel $X^t_{\hat{p}}$, and fix a function $d(\cdot,\cdot)$ (which will be detailed later) for measuring distances between patches. Then we compute the distances

$$d_{p\hat{p}} = d(patch^{k_m}_{X^t}(p)\ , patch^{k_m}_{X^t}(\hat{p})) \tag{2}$$

for points $\hat{p}$, which are in the known region, but are also in a neighborhood of the missing region. This neighborhood, indicated by a binary mask $(M^t)^c$ can be obtained by the morphological dilation operation:

$$(M^t)^c = dilate(M^t, dil) - M^t\ , \tag{3}$$

where dil is a hyper-parameter determining the region of valid pixels which are used for filling the missing ones.

After computing the distances $d_{p\hat{p}}$, the minimum distance d^* is found (the corresponding matching patches are illustrated in Fig. 2 (a)). Then the set $\Omega^{\varepsilon}(p)$ is defined as follows:

$$\Omega^{\varepsilon}(p) = \{X^t_{\hat{p}} \mid (M^t)^c_{\hat{p}} = 1,\ d(patch^{k_m}_{X^t}(p), patch^{k_m}_{X^t}(\hat{p})) \leq (1+\varepsilon)d^*\}\ , \quad (4)$$

where ε is a hyper-parameter.

After the $\Omega^{\varepsilon}(p)$ is determined, one can sample from it's histogram. For simplicity we sample a pixel $X^t_{\hat{p}}$ uniformly from the set $\Omega^{\varepsilon}(p)$.

When $X^t_{\hat{p}}$ is chosen, $patch^{k_f}_{X^t}(p)$ is replaced with $patch^{k_f}_{X^t}(\hat{p})$ in the tensor X^t (see Fig. 2 (b)). Notice that if $k_f > 1$, and we replace the corresponding patches simultaneously for all points p (in the boundary of the missing region), we will end up with multiple candidates for missing pixels X^t_p. In our algorithm these candidates are averaged to fill the pixel X^t_p for each point p in the boundary of the missing region (see Fig. 2 (c)).

After replacing patches centered in all boundary points, we result in a tensor $\hat{X}^t$, and take

$$X^{t+1} = (1 - \partial M^t) \odot X^t + \partial M^t \odot \hat{X}^t\ . \quad (5)$$

As the boundary pixels in X^{t+1} are filled, we also update the missing region by taking

$$M^{t+1} = M^t - \partial M^t\ . \quad (6)$$

Now it only remains to define the distance $d(\cdot,\cdot)$. As it is crucial to satisfy the condition $(C1)$, for measuring the distance between $patch^{k_m}_X(p)$ and $patch^{k_m}_X(\hat{p})$, we use only valid pixels in $patch^{k_m}_X(p)$. More precisely, d is defined as a normalized sum of squared distances between valid pixels in $patch^{k_m}_X(p)$ and corresponding pixels in $patch^{k_m}_X(\hat{p})$:

$$d_{p\hat{p}} = \frac{1}{sum(patch^{k_m}_M(p))} ||(patch^{k_m}_X(p) - patch^{k_m}_X(\hat{p})) \odot patch^{k_m}_M(p)||^2_2\ . \quad (7)$$

The result of onion-peel patch-match is denoted by O.

Convolution and Updating the Missing Region After the onion-peel patch-match is performed, we apply a convolution with a kernel size $k_c \times k_c$ to the tensor O, resulting in a tensor we denote by C. On the other hand, some pixels in the tensor O may remain unknown (the new missing region is indicated by M^T). So during the convolution some $k_c \times k_c$ sliding windows will contain missing pixels. Hence, in the tensor C we eliminate the results of convolving in such sliding windows. To obtain the centers of such sliding windows, one can use the morphological erosion operation with the kernel size $T - [k_c/2]$:

$$M' = erode(M, T - [k_c/2])\ . \quad (8)$$

We refer to M' as the *updated missing region after the onion convolution* (see Fig. 2 (d)).

So, the result of the onion convolution, with parameters $k_m, k_f, k_c, dil, \varepsilon$ is the tuple $(C \odot M', M')$.

Note that the onion convolution layer satisfies conditions $(C1)$ and $(C2)$. Indeed, by designing the $M \mapsto M'$ mask updating procedure, we keep from the tensor C only the signals, which are results of convolution in sliding windows with only valid pixels. So the condition $(C1)$ holds. On the other hand, our onion-peel patch-match is a generalization of the non-parametric sampling technique [1] designed to have no visual discontinuities between the boundary of the known and generated regions. As the onion-peel patch-match is applied to deep features, it has the property of preserving feature continuities, so the condition $(C2)$ also holds.

3.2 Discussion on Implementation

Note, that the onion convolution mechanism requires pairwise distance computations between patches centered in the regions indicated by ∂M^t and $(M^t)^c$ (for each iteration $t = 1, \dots, T$). Moreover, the $L2$-norm component in the Eq. 7 can be computed by using dot product due to the formula

$$||u - v||_2^2 = ||u||_2^2 - 2\langle u, v\rangle + ||v||_2^2 \tag{9}$$

for any vectors (patches) u, v. Therefore, similar to other long-range pixel dependency capturing mechanisms, we also need to calculate patch-wise dot products. In general, it is done by extracting patches

$$\mathcal{P}_{X^t}^{k_m}((M^t)^c) = \{patch_{X^t}^{k_m}(p) \mid (M^t)_p^c = 1\} \tag{10}$$

then convolving the tensor X with filters from $\mathcal{P}_{X^t}^{k_m}((M^t)^c)$.[3] This procedure contains dot product calculations also for pairs of patches, each of which is centered in the region $(M^t)^c$. To avoid these redundant computations, we merely extract patches $\mathcal{P}_{X^t}^{k_m}(\partial M^t)$ and compute distances for each pair

$$(patch_{X^t}^{k_m}(p), patch_{X^t}^{k_m}(\hat{p})) \in \mathcal{P}_{X^t}^{k_m}(\partial M^t) \times \mathcal{P}_{X^t}^{k_m}((M^t)^c)\ . \tag{11}$$

Our implementation is done purely with *TensorFlow* [27], resulting in an end-to-end pipeline for training and inference.

3.3 The Network Architecture

In our method, a generative adversarial network is used, the generator of which is the main inpainting network. As a discriminator, our approach uses the *SN-PatchGAN* introduced in [15] and based on the concept of *SN-GAN* [40].

The generator network consists of two parts: *coarse* and *refinement networks*.

[3] Moreover, replacing patches followed by averaging of overlapping regions, also can be done by using *transposed convolution* operation and $\mathcal{P}_{X^t}^{k_f}((M^t)^c)$ (see [38] for details).

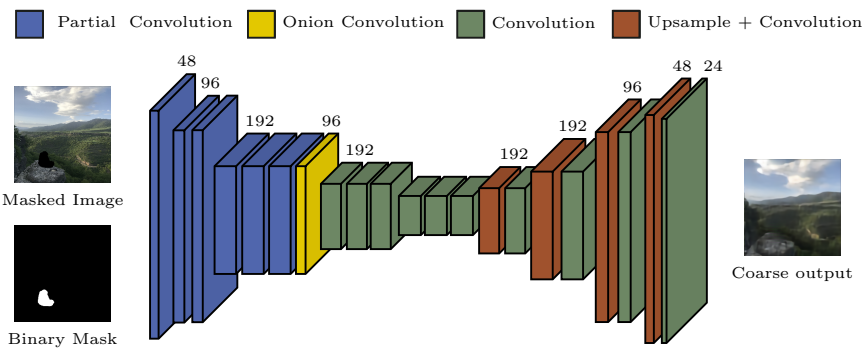


Fig. 3. The architecture of our coarse network. After each convolution the corresponding activation is used. The onion convolution layer is used with parameters $k_m = 4, k_f = 2, dil = 8, \varepsilon = 0.1, k_c = 3, T = \infty$, where $T = \infty$ means that we iteratively complete the missing region until it is filled completely.

Coarse Network Let I be an image normalized to the range $[-1, 1]$, and M be a binary mask indicating the missing region in the image. The network is fed by two inputs: $I \odot (1 - M)$ and M. The overall architecture of our coarse model is presented in Fig. 3. Six partial convolution layers with ELU [41] activation are used at the begining of the network to reduce the tensor sizes. Let's denote the output of sixth partial convolution by X. The binary mask M is resized to the size of X and is referred as the missing region indicator in the tensor X. Then the onion convolution layer (see Fig. 3). We have experimented with the hyper-parameters and the onion convolution layer position and find this is the optimal usage of it in our case. After the onion convolution layer the ELU activation is used. The rest of our coarse network is composed of convolutional layers and Nearest Neighbor Upsamplings, followed by convolutions. All convolutions, except the last one, are followed by activation functions ELU. In the end, tanh activation is used to obtain output in the range $[-1, 1]$. For details see the supplementary material.

Refinement Network After passing the image and the missing region through the coarse network, we obtain a rough estimation of pixels in the missing region. Let us denote the output of the coarse network by I_c. For getting more detailed output, the image $I_{comp} = I_c \odot M + I \odot (1 - M)$ is formulated and passed through another network, which we call *refinement network*. The architecture of the refinement network is very similar to the refinement network used in [15]. The only difference is using vanilla convolutions instead of gated convolutions (this difference is discussed in our ablation study, see Section 4.3).

3.4 Loss Functions

Our loss function consists of three terms: *pixel-wise reconstruction loss* $\mathcal{L}_1$, *adversarial loss* $\mathcal{L}_{ad}$ and perceptual loss $\mathcal{L}_p$. The total loss is a weighted sum of these losses:

$$\mathcal{L} = \lambda_1 \mathcal{L}_1 + \lambda_2 \mathcal{L}_{ad} + \lambda_3 \mathcal{L}_p \ , \tag{12}$$

where $\lambda_1, \lambda_2, \lambda_3$ are training hyperparameters. For optimizing the inpainting network G, the loss function $\mathcal{L}$ is minimized w.r.t. the generator's weights ω_G. At the same time, as there is an adversarial loss, $\mathcal{L}_{ad}$, it is maximized w.r.t. *SN-PatchGAN discriminator's* weights ω_D. We update ω_G and ω_D one after another at each step resulting in an equilibrium point for the GAN.

Let I_{orig} be the image, which is needed to be reconstructed given an image I with a missing region indicated by a binary mask M. Let I_c and I_r be the coarse and refinement networks' outputs, respectively. Each of our losses is discussed below in detail.

Pixel-Wise Reconstruction Loss We penalize each of our inpainting networks in all spatial locations by minimizing the *mean absolute error* between the original image I_{orig} and reconstructions I_c and I_r (similarly as in [15]):

$$\mathcal{L}_1 = ||I_c - I_{orig}||_1 + ||I_r - I_{orig}||_1 \ . \tag{13}$$

Adversarial Loss Our discriminator gets the original images I_{orig} as real examples and composition images $I_{compos} = I_{orig} \odot (1 - M) + I_r \odot M$ as fake examples. Since the discriminator belongs to the family of *PatchGAN*s, it outputs $3D$ tensors D_{real} and D_{fake}. As in [15], the *hinge loss* between the outputs of our discriminator is computed:

$$\mathcal{L}_{ad} = -\mathbb{E}[ReLU(1 - D_{real}) + ReLU(1 + D_{fake})] \ . \tag{14}$$

Perceptual Loss We also use the *perceptual (content) loss* introduced in [42], which minimizes the distance between the features of the original and the completed images obtained by the VGG-16 [43] network. Similar to [14], we compute distances between the vgg-features of three images: the original image I_{orig}, the output of the refinement network I_r and the composition image I_{compos}. More precisely, let $MP_i(X)$ be the output of the $MaxPool$ layer in the i^{th} block, when feeding the VGG network with an image X. Then our perceptuall loss is defined as follows:

$$\mathcal{L}_p = \sum_{i=1}^{3} \Big[||MP_i(I_r) - MP_i(I_{orig})||_1 + ||MP_i(I_{compos}) - MP_i(I_{orig})||_1 \Big] \ . \tag{15}$$

Thus, the total loss $\mathcal{L}$ is a weighted sum of above-mentioned three losses. In our experiments $\lambda_1 = 1, \lambda_2 = 1, \lambda_3 = 0.05$ are taken.

4 Experiments

We have experimented with the dataset Places2 [44]. Our model is trained by the ADAM optimizer [45] with learning rate 10^{-4} and parameters $\beta_1 = 0.5$, $\beta_2 = 0.999$. For all experiments, a single *NVIDIA V100 GPU* (with 16GB RAM) is used with batch size equal to 28. Images and binary masks are resized to the size 256×256 before feeding the network; no augmentations are used. The training lasted 14 days.

To evaluate our model, we have randomly chosen a test-set from 500 images from the test set of the *Mapillary* [46] dataset. We have created random free-form masks as it was done in [15]. The masks are in different sizes equiprobable from area ranges covering $10 - 20$, $20 - 30$, $30 - 40$ or $40 - 50$ percents of the image area. All masks and images are resized to the size of 256×256.

Due to our efficient implementation of the onion convolution layer and the fact that our network is fully convolutional, we can also process images with resolution 512×512 (see Fig. 1).

We compare our method with state-of-the-art methods[4]: Gated Convolutions (GC) [15] and Partial Convolutions (PC) [14].

4.1 Qualitative Comparison

As shown in Fig. 4, PC [14] reconstructs semantics, but introduce some blur (e.g. rows 5 and 7 in Fig. 4), pattern (e.g. rows $1, 6$ in Fig. 4) or sometimes does not preserve continuous lines (e.g. rows $1, 2, 3$ in Fig. 4). GC [15] shows plausible results, but sometimes does not keep the image structure (rows $3, 4$ in Fig. 4), introduce some "dirt" (row 1 in Fig. 4) or generates some strange artifacts (e.g. rows $2, 6, 7$ in Fig. 4).

In comparison with these methods, our method can reconstruct detailed textures and coherently generate structures due to its property of feature continuity. Also, our method does not generate artifacts not related to the context of the image.

Table 1. Comparison of our method with PC [14] and GC [15].

	PSNR	*SSIM*	*MAE*
Partial Convolutions (PC)	**22.14**	0.794	0.036
Gated Convolutions (GC)	21.97	0.83	**0.029**
Onion Convolutions (our)	22	**0.835**	**0.029**

[4] For comparison we take the pretrained GC [15] model from the official repository. As there is no official implementation of the method [14] PC, we make our own, which benefits a lot from https://github.com/MathiasGruber/PConv-Keras.

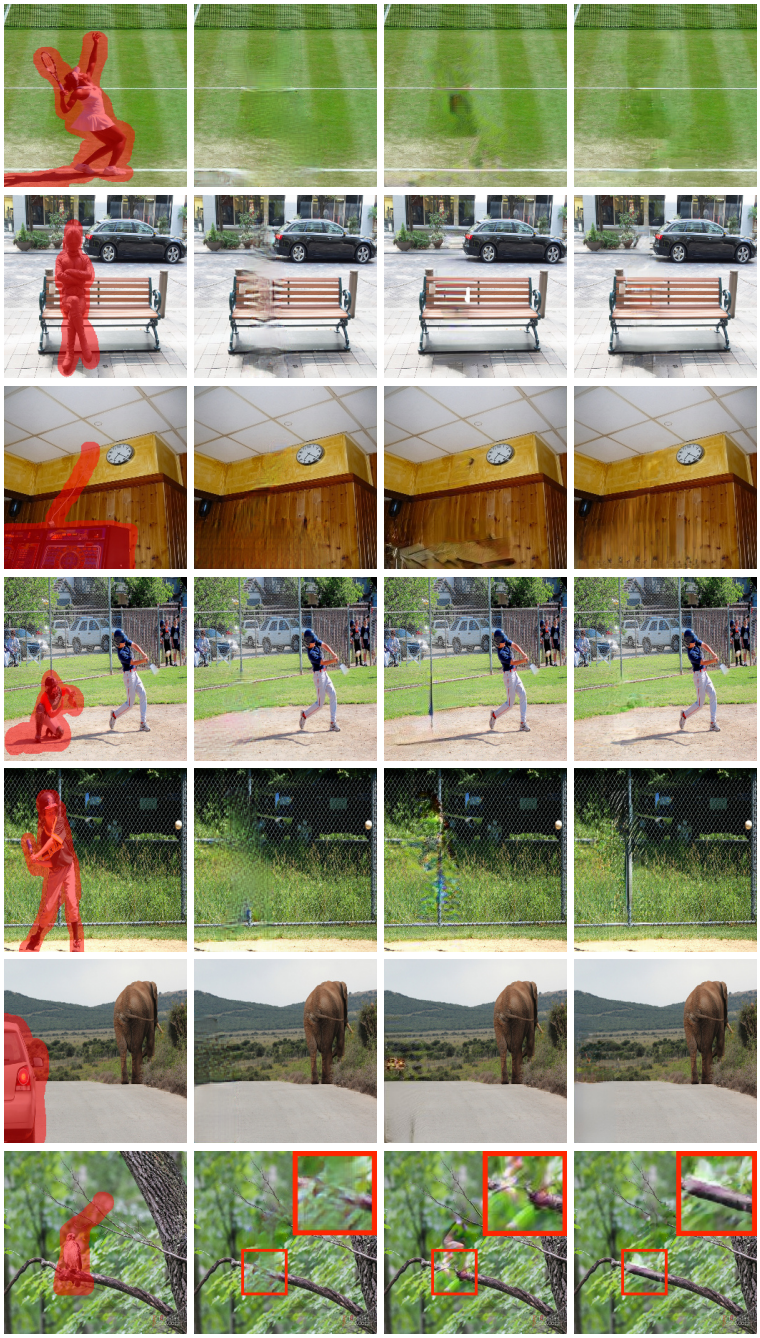


Fig. 4. Comparisons of our method with PC [14] and GC [15]. For more images please see the supplementary material.

4.2 Quantitative Comparison

To compare our method with PC [14] and GC [15], we consider 500 images from *Mapillary* dataset [46], generate irregular missing regions and evaluate methods by calculating standard evaluation metrics *PSNR*, *SSIM* [47] and *Mean Absolute Error (MAE)*. The evaluation results are presented in Table 1 and show that our method outperforms the others.

As shown in the Table 1 in terms of SSIM, our method outperforms the other two. In terms of MAE, our method performs similarly with GC and better than PC. In terms of PSNR, PC is better, but it mostly comes from the fact that PC sometimes gives more smooth and blurry results, which is not looking realistic.

4.3 Ablation Study

The main idea of our approach is hooded under the mechanism of onion convolution: filling the missing region iteratively by only using valid pixels without any pre-estimating a coarse output in that region. So we start to analyze the effect of our *onion convolution layer*. Also, we analyze the impact of replacing the onion convolution with consecutive onion convolutions. Then we study the effects of the perceptual loss, the replacing gated convolutions with vanilla convolutions in the refinement network.

Effect of Onion Convolution Layer As shown in Fig. 5, our model with onion convolution Fig. 5(b) helps the model coherently fill the missing region. Especially in the case of large missing regions, the model without onion convolutions Fig. 5(c) cannot propagate sufficient information from the known region to the unknown.

Effect of a Single Onion Convolution Instead of Consecutive Onion Convolutions We also perform experiments with replacing the II-VI partial convolution layers with 6 onion convolution layers in our coarse model. The results are shown in Fig. 5(d). It can be noticed that sometimes this architecture leads to more structure-preserving results (e.g. column 4 in Fig. 5(d)), but in some cases it can introduce black (white) artifacts or blur.

Effect of Perceptual Loss We also train our model without perceptual loss we have mentioned in Section 3. The results, presented in Fig. 5(e), show that using the perceptual loss in our case helps to avoid some patterns and inconsistencies.

Effect of Vanilla Convolutions Instead of Gated Convolutions We have also experimented with gated convolutions [15] in the refinement network and find out that, in our case, gated convolutions introduce artifacts as shown in Fig. 5(f). We suppose this is due to the property of "vanishing" some regions when using gated convolution (sometimes the gate of a gated convolution does not allow some information to pass through the layer).

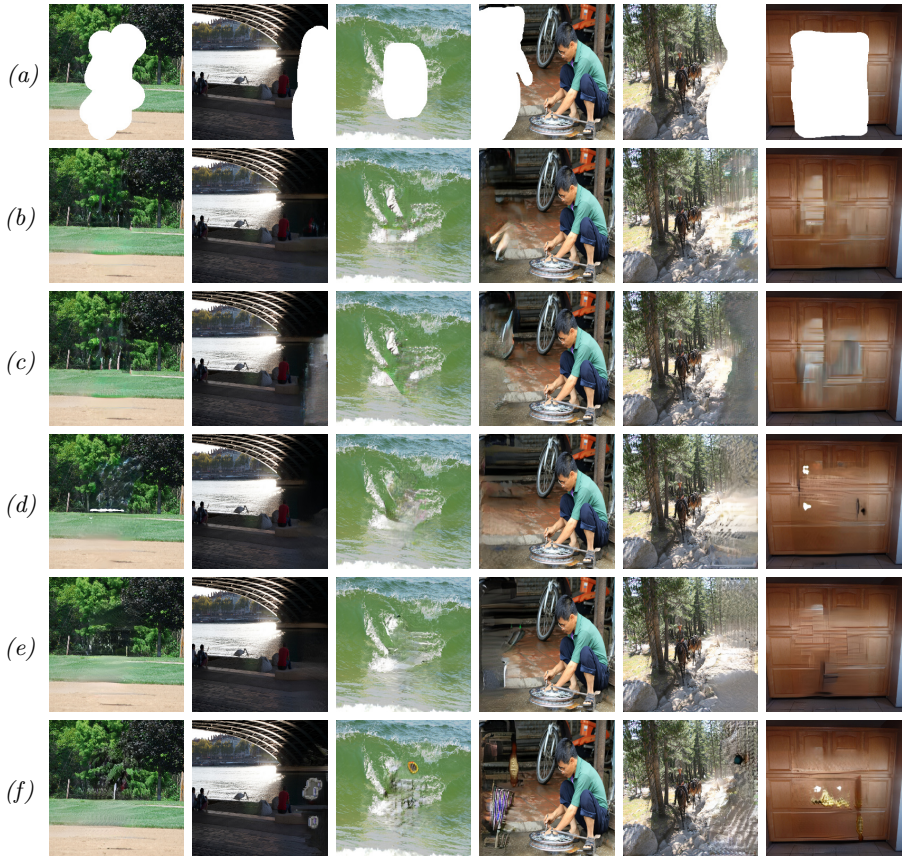


Fig. 5. *(a)* Masked image, *(b)* our model, *(c)* model without onion convolutions, *(d)* model with using a sequence of onion convolutions, *(e)* model without perceptual loss, *(f)* model with gated convolutions.

5 Conclusion

We present a novel patch-based onion convolution mechanism for image inpainting problem. By using the ability to capture long-range pixel dependencies, the onion convolution is designed to propagate the information from known regions to missing ones. We show that our method quantitatively and qualitatively outperforms existing state-of-the-art approaches of image inpainting. It is worth noting that our onion convolution can be adopted with various architectures and learning techniques. In the future, we plan to extend this method to face completion and image super-resolution tasks.

References

1. Efros, A., Leung, T.: Texture synthesis by non-parametric sampling. In: In International Conference on Computer Vision. (1999) 1033–1038
2. Barnes, C., Shechtman, E., Finkelstein, A., Goldman, D.B.: PatchMatch: A randomized correspondence algorithm for structural image editing. ACM Transactions on Graphics (Proc. SIGGRAPH) **28** (2009)
3. Ashikhmin, M.: Synthesizing natural textures. In: Proceedings of the 2001 Symposium on Interactive 3D Graphics. I3D '01, New York, NY, USA, Association for Computing Machinery (2001) 217–226
4. Wei, L.Y., Levoy, M.: Fast texture synthesis using tree-structured vector quantization. In: Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques. SIGGRAPH '00, USA, ACM Press/Addison-Wesley Publishing Co. (2000) 479–488
5. Harrison, P.: A non-hierarchical procedure for re-synthesis of complex textures. (2000)
6. Efros, A.A., Freeman, W.T.: Image quilting for texture synthesis and transfer. In: Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques. SIGGRAPH '01, New York, NY, USA, Association for Computing Machinery (2001) 341–346
7. Criminisi, A., Pérez, P., Toyama, K.: Object removal by exemplar-based inpainting. Volume 2. (2003) 721–728
8. Criminisi, A., Perez, P., Toyama, K.: Region filling and object removal by exemplar-based image inpainting. Trans. Img. Proc. **13** (2004) 1200–1212
9. Sun, J., Yuan, L., Jia, J., Shum, H.Y.: Image completion with structure propagation. ACM Trans. Graph. **24** (2005) 861–868
10. Hung, J., Chun-Hong, H., Yi-Chun, L., Tang, N., Ta-Jen, C.: Exemplar-based image inpainting base on structure construction. Journal of Software **3** (2008)
11. Huang, J.B., Kang, S.B., Ahuja, N., Kopf, J.: Image completion using planar structure guidance. ACM Trans. Graph. **33** (2014)
12. Pathak, D., Krähenbühl, P., Donahue, J., Darrell, T., Efros, A.A.: Context encoders: Feature learning by inpainting. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2016) 2536–2544
13. Iizuka, S., Simo-Serra, E., Ishikawa, H.: Globally and locally consistent image completion. ACM Transactions on Graphics **36** (2017) 1–14
14. Liu, G., Reda, F.A., Shih, K.J., Wang, T.C., Tao, A., Catanzaro, B.: Image inpainting for irregular holes using partial convolutions. In: ECCV. (2018)
15. Yu, J., Lin, Z., Yang, J., Shen, X., Lu, X., Huang, T.: Free-form image inpainting with gated convolution. In: 2019 IEEE/CVF International Conference on Computer Vision (ICCV). (2019) 4470–4479
16. Zheng, C., Cham, T.J., Cai, J.: Pluralistic image completion. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2019) 1438–1447
17. Xie, J., Xu, L., Chen, E.: Image denoising and inpainting with deep neural networks. In: Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1. NIPS'12, Red Hook, NY, USA, Curran Associates Inc. (2012) 341–349
18. 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: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2017) 6882–6890

19. Ren, J.S.J., Xu, L., Yan, Q., Sun, W.: Shepard convolutional neural networks. In: NIPS. (2015)
20. Xie, C., Liu, S., Li, C., Cheng, M., Zuo, W., Liu, X., Wen, S., Ding, E.: Image inpainting with learnable bidirectional attention maps. In: 2019 IEEE/CVF International Conference on Computer Vision (ICCV). (2019) 8857–8866
21. Yang, C., Lu, X., Lin, Z., Shechtman, E., Wang, O., Li, H.: High-resolution image inpainting using multi-scale neural patch synthesis. (2016)
22. Yan, Z., Li, X., Li, M., Zuo, W., Shan, S.: Shift-net: Image inpainting via deep feature rearrangement. In: The European Conference on Computer Vision (ECCV). (2018)
23. Song, Y., Yang, C., Lin, Z., Liu, X., Huang, Q., Li, H., Jay Kuo, C.C.: Contextual-based image inpainting: Infer, match, and translate. In: Proceedings of the European Conference on Computer Vision (ECCV). (2018) 3–19
24. Liu, H., Jiang, B., Xiao, Y., Yang, C.: Coherent semantic attention for image inpainting. 2019 IEEE/CVF International Conference on Computer Vision (ICCV) (2019) 4169–4178
25. Yu, J., Lin, Z., Yang, J., Shen, X., Lu, X., Huang, T.S.: Generative image inpainting with contextual attention. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. (2018) 5505–5514
26. Zhang, H., Goodfellow, I., Metaxas, D., Odena, A.: Self-attention generative adversarial networks. In: Proceedings of the 36th International Conference on Machine Learning. Volume 97 of Proceedings of Machine Learning Research., Long Beach, California, USA, PMLR (2019) 7354–7363
27. Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., Zheng, X.: TensorFlow: Large-scale machine learning on heterogeneous systems (2015) Software available from tensorflow.org.
28. Bertalmío, M., Sapiro, G., Caselles, V., Ballester, C.: Image inpainting. In: SIGGRAPH '00. (2000)
29. Bertalmio, M., Vese, L., Sapiro, G., Osher, S.: Simultaneous structure and texture image inpainting. IEEE Transactions on Image Processing **12** (2003) 882–889
30. Liang, L., Liu, C., Xu, Y.Q., Guo, B., Shum, H.Y.: Real-time texture synthesis by patch-based sampling. ACM Trans. Graph. **20** (2001) 127–150
31. Kwatra, V., Schödl, A., Essa, I., Turk, G., Bobick, A.: Graphcut textures: Image and video synthesis using graph cuts. ACM Trans. Graph. **22** (2003) 277–286
32. Boykov, Y., Veksler, O., Zabih, R.: Fast approximate energy minimization via graph cuts. IEEE Transactions on Pattern Analysis and Machine Intelligence **23** (2001) 1222–1239
33. Köhler, R., Schuler, C., Schölkopf, B., Harmeling, S.: Mask-specific inpainting with deep neural networks. In: German Conference on Pattern Recognition, Cham, Springer International Publishing (2014) 523–534
34. Nazeri, K., Ng, E., Joseph, T., Qureshi, F.Z., Ebrahimi, M.: Edgeconnect: Generative image inpainting with adversarial edge learning. ArXiv **abs/1901.00212** (2019)
35. Xiong, W., Yu, J., Lin, Z., Yang, J., Lu, X., Barnes, C., Luo, J.: Foreground-aware image inpainting. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2019)

36. Song, Y., Yang, C., Shen, Y., Wang, P., Huang, Q., Kuo, C.J.: Spg-net: Segmentation prediction and guidance network for image inpainting. CoRR **abs/1805.03356** (2018)
37. Goodfellow, I.J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2. NIPS'14, Cambridge, MA, USA, MIT Press (2014) 2672–2680
38. Chen, T.Q., Schmidt, M.: Fast patch-based style transfer of arbitrary style. arXiv preprint arXiv:1612.04337 (2016)
39. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L.u., Polosukhin, I.: Attention is all you need. In: Advances in Neural Information Processing Systems 30. Curran Associates, Inc. (2017) 5998–6008
40. Miyato, T., Kataoka, T., Koyama, M., Yoshida, Y.: Spectral normalization for generative adversarial networks. ArXiv **abs/1802.05957** (2018)
41. Clevert, D.A., Unterthiner, T., Hochreiter, S.: Fast and accurate deep network learning by exponential linear units (elus). CoRR **abs/1511.07289** (2015)
42. Gatys, L.A., Ecker, A.S., Bethge, M.: Image style transfer using convolutional neural networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2016) 2414–2423
43. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
44. Zhou, B., Lapedriza, A., Khosla, A., Oliva, A., Torralba, A.: Places: A 10 million image database for scene recognition. IEEE transactions on pattern analysis and machine intelligence **40** (2017) 1452–1464
45. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. CoRR **abs/1412.6980** (2015)
46. Neuhold, G., Ollmann, T., Rota Bulò, S., Kontschieder, P.: The mapillary vistas dataset for semantic understanding of street scenes. In: International Conference on Computer Vision (ICCV). (2017)
47. Zhou Wang, Bovik, A.C., Sheikh, H.R., Simoncelli, E.P.: Image quality assessment: from error visibility to structural similarity. IEEE Transactions on Image Processing **13** (2004) 600–612