ԵՐԵՎԱՆԻ ՊԵՏԱԿԱՆ ՀԱՄԱԼՍԱՐԱՆ

ՄԱԹԵՄԱՏԻԿԱՅԻ ԵՎ ՄԵԽԱՆԻԿԱՅԻ ՖԱԿՈՒԼՏԵՏ

ԿԻՐԱՌԱԿԱՆ ՎԻՃԱԿԱԳՐՈՒԹՅՈՒՆ ԵՎ ՏՎՅԱԼՆԵՐԻ ԳԻՏՈՒԹՅՈՒՆ ՄԱԳԻՍՏՐՈՍԱԿԱՆ ԾՐԱԳԻՐ

ՕՀԱՆՅԱՆ ՄԱՐԻԱՆՆԱ ՏԻԳՐԱՆԻ

ՄԱԳԻՍՏՐՈՍԱԿԱՆ ԹԵԶ

Պատկերի վերակառուցում խորը ուսուցման մեթոդներով

«Կիրառական վիճակագրություն և տվյալների գիտություն» մասնագիտությամբ վիճակագրության մագիստրոսի որակավորման աստիճանի հայցման համար

ԵՐԵՎԱՆ 2022

Ուսանող՝ ————————— , Օհանյան Մարիաննա

Գիտական ղեկավար՝ ————————— , Նավասարդյան Շանթ

«Թույլատրել պաշտպանության»

Մագիստրոսական ծրագրի ղեկավար՝

ֆ.մ.գ.դ., ասիստենտ Կարեն Քեռյան

Թեզի վերնագիրը

Հայերենով՝ Պատկերի վերակառուցում խորը ուսուցման մեթոդներով,

Ռուսերենով՝ Восстановление изображений с использованием методов глубокого обучения,

Անգլերենով՝ Image inpainting using deep learning methods

Համառոտագիր

Վերջերս խորը ուսուցման մեթոդները մեծ հաջողությունների են հասել պատկերի վերակառուցման խնդրում: Այնուամենայնիվ, բարդ կառուցվածքների անընդհատ վերակառուցումը մնում է պատկերների մշակման ոլորտի համար դժվար խնդիր: Այս աշխատանքում ներկայացված է «onion convolution»-ների ընտանիքը, որի գաղափարն առաջանում է, երբ դիտարկում ենք «patch-based» և «attention-based» մեթոդների միջև կապերը: «Onion convolution»-ները նեյրոնային ցանցերի մեջ օգտագործվող բլոկներ են, որոնք նախատեսված են լրացնելու նկարի անհայտ տիրույթը իտերատիվ եղանակով: Դրանք թույլ են տալիս անընդհատ տարածել կառուցվածքներն ու տեքստուրաները հայտնի տիրույթից դեպի բացակայող տիրույթ՝ վերջնական արդյունքը համապատասխանեցնելով պատկերների բարձրորակ վերակառուցման մարդկային չափանիշներին: Ինչպես ցույց են տալիս որակական և քանակական համեմատությունները, «onion convolution»-ներ օգտագործող մեր մեթոդը գերազանցում է ժամանակակից գոյություն ունեցող մեթոդներին՝ տալով ավելի իրական թվացող, աչքի համար հաճելի և բովանդակության առումով իմաստալից արդյունքներ:

Abstract

Recently deep learning methods have achieved 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 the family of onion convolutions is presented, the concept of which arises from a connection between patch-based techniques and attention mechanisms.

The onion convolutions are building blocks designed for the iterative completion of the missing region from its boundary to the center.

It allows to continuously propagate structures and textures from the known region to the missing one and meet human criteria on high-quality image completions.

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.

Image Inpainting Using Deep Learning methods

Marianna Ohanyan

May 24, 2022

Contents

1 Introduction **2**

2 Related Work **4**
2.1 Image Inpainting Approaches . . . 4
2.1.1 Traditional Computer Vision Methods . . . 4
2.1.2 Learning-Based Methods . . . 6
2.2 The Attention Mechanism . . . 7
2.3 Patch-Based Techniques . . . 8
2.4 The Connection Between Attention Mechanisms and Patch-Based Techniques . . . 9

3 Approach **10**
3.1 The Family of Onion Convolution Modules . . . 11
3.1.1 Onion-Peel Patch-Match . . . 11
3.1.2 Convolution and Updating the Missing Region . . . 13
3.2 Defining the Similarities $s_{p\hat{p}}$ . . . 13
3.2.1 Taking the Best Matching Patches . . . 14
3.2.2 Turning Distances into Similarities by the Function $f(x) = -x$ . . . 15
3.2.3 Turning Distances into Similarities by the Function $f(x) = 1/(1+x)$ . . . 15
3.2.4 Taking the Cosine Similarity . . . 15
3.3 Discussion on Implementation . . . 15
3.4 The Network Architecture . . . 16
3.4.1 Coarse Network . . . 17
3.4.2 Refinement Network . . . 17
3.5 Loss Functions . . . 17
3.5.1 Pixel-Wise Reconstruction Loss . . . 18
3.5.2 Adversarial Loss . . . 18
3.5.3 Perceptual Loss . . . 19

4 Experiments **19**
4.1 Implementation Details . . . 19
4.2 Comparison with State-of-the-Art Methods . . . 21
4.2.1 Qualitative Comparison . . . 21
4.2.2 Quantitative Comparison . . . 25
4.3 A Study on the Family of Onion Convolutions . . . 25
4.4 Study on Error Propagation in Onion Convolution . . . 27

5 Conclusion **27**

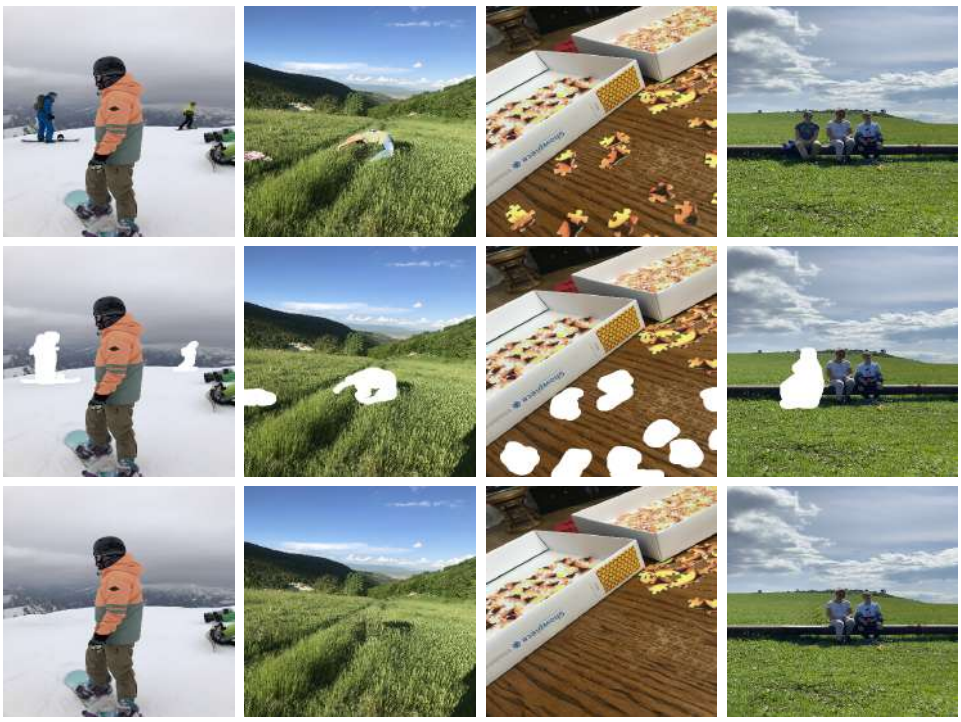


Figure 1: Image inpainting results of our method. Each vertical triplet of images is composed of the original image, the image with the missing region, and the inpainted result. The method allows users to remove unwanted objects or fill missing parts in images. Please see in color.

1 Introduction

Image inpainting is the process 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 (see Fig. 1).

Understanding the human perception of high quality image restoration, perhaps, is the first step to any solution of the image inpainting problem. In fact, different image restoration experts may have different understandings of the *visual plausibility* of image completion. However, after consulting with some of them, the following general and intuitive criteria on image inpainting quality were obtained.

- The completed region must have a *semantically correct* logic, i.e. the generated objects or their parts must be logically possible.
- The *structures* in the image *should preserve continuities* after the inpainting is done, i.e. lines and curves in the known region must be continued to the generated region.
- The *textures* in the generated region should be *visually realistic* and *coherent with* the textures in the known region.

Some traditional approaches (Efros and Leung, 1999; Wei and Levoy, 2000; Harrison, 2001; Ashikhmin, 2001; Efros and Freeman, 2001; Criminisi et al., 2003, 2004; Barnes et al., 2009) are based on texture synthesis techniques and achieve great success in meeting the *realistic-looking texture criterion*. Due to their patch-based nature, these methods also meet the criterion of *preserving structural continuities* in many cases. However, for complex structures, traditional approaches rarely give reasonable outputs.

Later, some methods (Sun et al., 2005; Hung et al., 2008; Huang et al., 2014) were proposed to deal with the cases of complex structures and achieved notable improvements. Yet, generating *semantically correct* outputs remains beyond the abilities of traditional methods. Hence, deep learning techniques are proposed

for semantic image inpainting. Early methods (Xie et al., 2012; Pathak et al., 2016; Yeh et al., 2017; Iizuka et al., 2017) use vanilla convolutions and manage to gain *semantically meaningful* results for small missing regions. However, in the case of complicated real-world images these methods introduce blur, structural distortions, edge artifacts, and color inconsistencies.

The reason is that during the *convolution operation* there may be sliding windows (see Fig. 2 (*d*), marked with red cross) containing missing/invalid pixels. In fact, such pixels contain *misleading information* that vanilla convolutions treat as useful information. This phenomenon harms the results especially in cases of large missing regions.

Some methods (Ren et al., 2015; Liu et al., 2018; Yu et al., 2019; Xie et al., 2019; Yi et al., 2020) introduce *special building blocks* which are designed to overcome this issue by their ability to *ignore the values* of the missing region. The family of onion convolutions also has such ability. This approach allows to gain a drastic improvement over using only vanilla convolutions. Thus in order to gain high quality results in image inpainting problem a network architecture needs to contain *building blocks* with the following property:

(*C*1) Valid/known pixels have a higher impact on the output of the block than missing ones.

Although the issue with the irrelevant/unknown information is addressed by the *special blocks* mentioned above, this is still insufficient for obtaining *semantically correct* image completions for complicated images with excessive content. Indeed, to obtain semantically meaningful results, it is natural to expect the image inpainting algorithms to look at the relevant semantic information in the image while generating a certain part of the missing region.

Experiments show that just having *a large receptive field* is not enough, and one needs to have building blocks that can generate *each pixel* by incorporating a sufficiently large region instead of only looking at the neighborhood of that pixel.

For having such building blocks, some approaches (Yu et al., 2018, 2019; Zheng et al., 2019; Yi et al., 2020) adopted the self-attention mechanism (Zhang et al., 2019) for image inpainting problem.

The attention mechanism, with its ability to *capture long-range pixel dependencies*, enables image inpainting algorithms to meet the criterion of *semantically correct completions* by so proving the necessity of containing building blocks with the following property:

(*C*2) For producing the output at each pixel, the block takes into account a sufficiently large neighborhood of the missing region.

Notwithstanding the great success of the learning-based approaches mentioned so far, all these methods complete the missing region in a *one-shot manner* when all missing pixels are filled simultaneously without any order. This sometimes may lead to structural distortions. Therefore, motivated by human image completions, some methods (Oh et al., 2019; Li et al., 2020a) propose *iteratively filling the hole* from its boundary to its center. However, at each iteration they roughly fill *the boundary* of the remaining missing region with *convolutional encoders*, then use *pixel-level attention mechanisms* to refine the completions in the boundary areas. So the outputs of these methods highly depend on the *coarse estimations* (obtained by the *convolutional encoders*) of the missing pixels at each iteration. In some cases networks fail to progressively make *continuity-preserving* coarse estimations, which leads to outputs with structure discontinuities (see Fig. 6).

To address this problem the *onion convolution layer* was proposed (Navasardyan and Ohanyan, 2020) (the conference version of this work), which iteratively fills the missing region (starting from its boundary to the center) without any coarse estimation of the missing boundary at each step. Moreover, when filling each boundary location only the *known part* of its neighborhood is incorporated for computing *patch-level similarities*. This forces the feature activations to be propagated from the known region to the missing one without any risks coming from deteriorated coarse estimations.

Hence the onion convolution block is designed to satisfy the following condition motivated by the criterion of *preserving structure continuities*:

($C3$) The block continuously propagates the information (e.g. texture, structure, semantics) of the know region to the unknown.

Later we have noticed a connection between the *patch-based* nature of the onion convolution and the attention mechanisms. This connection is discussed in Section 2.4 and leads us to the *family of onion convolutions*. Being a hybrid method, the family of onion convolutions inherits the *long-range pixel dependency* capturing ability from attention mechanisms and the *structure and texture continuity preserving* ability from patch-based techniques. This allows us to significantly outperform our initial results with the onion convolution layer (Navasardyan and Ohanyan, 2020) (see Section 4).

To summarize, our main contributions are the following:

- We *reveal a connection* between *patch-based techniques* and *attention mechanisms*, which leads us to an extension of the concept of the *onion convolution layer* (Navasardyan and Ohanyan, 2020) to the concept of the *family of onion convolutions*.
- Our onion convolution family is motivated by human criteria on a visually plausible image completion and is designed to satisfy the conditions $(C1), (C2)$ and $(C3)$ (some results can be found in Fig. 1).
- Our solution to the image inpainting problem gains notable improvements over recent state-of-the-art image inpainting approaches.

The structure of the further part of this work is the following: in Section 2 some existing approaches to the image inpainting problem, the attention mechanism, and patch-based techniques are reviewed. This is followed by Section 2.4 where we discuss the connection between patch-based techniques, and attention mechanisms. In Section 3, a detailed description of our approach can be found. 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 draw a conclusion.

2 Related Work

This section starts with a literature review on existing solutions of the image inpainting problem. Then with a brief introduction the attention mechanism and patch-based techniques are recalled to prepare the ground for discussing the connection between them in the next section.

2.1 Image Inpainting Approaches

The existing approaches to the image inpainting problem can be roughly divided into two groups. The first group uses traditional computer vision techniques while the second one utilizes learning processes. In this Subsection a brief discussion on these groups is conducted.

2.1.1 Traditional Computer Vision Methods

Historically, the image inpainting problem was proposed for *image restoration*, when scratches, torn parts or text in images were needed to be removed. Some algorithms (Bertalmío et al., 2000; Bertalmio et al., 2003) were suggested for these purposes which rely on the idea of propagating image structures from the known region to the unknown by *diffusion processes*.

In (Bertalmío et al., 2000) the authors have examined the steps which image restoration experts follow during the restoration process. They roughly divide the process into four steps. The first three, concerning the structure propagation, are covered by their algorithm, which predicts the *isophotes* (lines of equal gray

values) direction field in the missing region. Inspired by (Bertalmío et al., 2000), *T. F. Chan and J. Shen* (Shen and Chan, 2002) introduce the *total variation (TV) inpainting*. In (Chan and Shen, 2000) the authors set out the major drawback of (Shen and Chan, 2002): it can not recover the missing part of an object, when its known parts are far from each other. They address this issue by taking into account the *curvatures of the isophotes* resulting with the *Curvature-Driven Diffusion (CDD)* inpainting model. Image gradient magnitudes and pairwise gradient angles are used in (Levin et al., 2003) to compute global image statistics for the inpainting of non-linear structures.

All these diffusion-based methods are devised for restoring small damaged parts in images and sometimes cause blurry results in cases when the missing regions are with *large connected components*. To address this problem, a part of previous work benefits from *patch-based techniques*. These methods use the idea of *texture replication with patches*, i.e. totally or partially replacing patches of the missing region with their *matching patches* from the known region of the image. This technique allows *hallucinating coherent texture background* for the missing region. To the best of our knowledge, such a patch-based technique was firstly introduced in (Garber, 1981) for texture synthesis, but was discarded due to its infeasibility at that time. Later in (Efros and Leung, 1999), authors apply this technique in order to sample from the *conditional distribution of each pixel given its neighborhood*. Despite the effectiveness for texture propagation, the algorithm described in (Efros and Leung, 1999) is extremely slow due to time-consuming estimations of conditional distributions at each iteration. To address the costly computations, some methods (Wei and Levoy, 2000; Harrison, 2001; Ashikhmin, 2001) were proposed to optimize the algorithm. Also, the *approximate nearest neighbor algorithm* from (Barnes et al., 2009) can be adopted and significantly speed-up the technique. However, as in the basic algorithm, all above-mentioned optimizations complete 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 (Efros and Leung, 1999)).

Later, to address the problem of "growing garbage", some *texture synthesis* algorithms suggest copying regions with more than one pixel at a time. On the other hand, pasting patches into neighboring locations may cause some patches to *overlap*, so a technique is needed to stitch together overlapping patches. In (Efros and Freeman, 2001) authors refer to this stitching problem as *image quilting* and propose a method with *minimum error boundary cut*. Later in (Kwatra et al., 2003) an algorithm for coherently stitching patches was introduced using graph cut techniques (Boykov et al., 2001).

Apart from successfully hallucinating textures, patch-based methods also can recover the structure information in the missing region. However, some algorithms try to improve patch-based methods in this direction. Criminisi et al. (2003, 2004) show that a significant improvement can be achieved by just specifying *the order* of patch-based filling of the missing region. They assign to each pixel a priority to be filled, and the patches with more priority in their centers are replaced with their matching patches first. *Rane et al.* (Rane et al., 2003) consider the problem of missing block recovery after JPEG compression and propose to classify whether a block is "texture" or "structure" block and fill the missing part by the algorithm from (Efros and Leung, 1999) in the case of "texture"-block and by the algorithm from (Bertalmío et al., 2000) in the case of "structure"-block. A similar approach was adopted also in the paper (Bertalmio et al., 2003), but instead of classifying a missing region into texture or structure, the initial image was decomposed into two images - texture image and structure image. After filling the missing regions in these images by corresponding algorithms, the results were summed up to obtain the final result. To the best of our knowledge, (Bertalmio et al., 2003) is the first attempt to apply patch-based inpainting techniques not on the original image but on *features* obtained from the original image, and our algorithm is also in this manner.

Later, the topic of structure preserving image inpainting has been gaining more and more attention from researchers. This motivation is based on the human perception of coherent completion of scenes. Especially, there are some units such as planes, curves, circles, which can be missed by methods discussed so far, but are very important for our scene understanding and visual perception. (Hung et al., 2008) extracts the edges of the known part of the image, uses *Bézier curves* for edge completion, and guides an exemplar-based technique

with completed edges. (Sun et al., 2005) requires users to provide a part of the *missing structure* by drawing some important missing edges in the image and (Barnes et al., 2009) shows how to use random search to speed up this algorithm. In (Huang et al., 2014) authors guide a patch-based technique by planar structures present in the image. Though these methods improve the quality of inpainting complicated structures, they are mostly either limited to some specific structures or do not take into account the semantics of the image.

For such purposes the *learning-based methods* were introduced.

2.1.2 Learning-Based Methods

After computer vision adopted deep learning techniques with bells and whistles, researchers started to experiment with *convolutional neural networks* in the image inpainting problem. The advantage of deep learning techniques is the ability of modeling high-level semantics of images, which could not be done before. Early work on image inpainting with neural networks (Xie et al., 2012; Köhler et al., 2014; Ren et al., 2015) concentrates on completing locally corrupted images, when missing regions are small but possibly cover a big part of images. Moreover, (Xie et al., 2012) proposes an algorithm for *blind inpainting* (when the missing region or damaged part of the image must be detected automatically and completed), and (Köhler et al., 2014) has an option for it. In (Ren et al., 2015) *Shepard Convolution* is introduced to condition the convolution output only on valid pixels. Later, (Liu et al., 2018) has developed this idea and obtained state-of-the-art results.

By means of *generative adversarial networks*, *GANs* (Goodfellow et al., 2014), researchers have reached new heights in solving computer vision tasks. The ability of GANs to generate visually plausible results helps neural networks in the image inpainting problem to coherently hallucinate the missing region. In (Pathak et al., 2016) authors use encoder-decoder architecture to fill large missing regions. They show the advantage of training with adversarial loss and start a new direction for the learning-based image inpainting algorithms. Another GAN-based method (Yeh et al., 2017) trains a GAN to fit the real data distribution, then, at inference time, tries to fit the known region of the input by optimizing over the latent image manifold. Later, with the aim of keeping both globally and locally consistencies of the generated missing regions, (Iizuka et al., 2017) introduced global and local context discriminators and performs promising results on missing regions of arbitrary forms.

Another work that is worth mentioning is (Yang et al., 2016). The authors initialize the missing region by using the output of their coarse network, then iteratively update the values in the missing region with a *patch-based* technique similar to (Li and Wand, 2016) (minimizing the distances between matching patches). Later, (Song et al., 2018a) used the *patch-based* technique described in (Chen and Schmidt, 2016), which replaces the patches in the unknown regions with their matching patches from the known regions. Similarly, Yan et al. (2018) use a special case of this technique, by considering patches of size 1×1 and combining features from both encoder and decoder for computing similarities. Similar to our method, Yi et al. (2020) use *patch-level attention* mechanism encapsulated in their *Attention Computing Module* and *Attention Transfer Module*. The methods described in (Yang et al., 2016; Song et al., 2018a; Yan et al., 2018; Yi et al., 2020) utilize patch-based techniques as we do for our *family of onion convolutions*, but in contrast with them, we do not need to roughly fill the missing region to apply our patch-based technique. Moreover, despite the usage of a parallel implementation of the technique (described in (Chen and Schmidt, 2016)), some of their implementations remain inefficient in the case of arbitrary-formed missing regions due to unnecessary computations, which we avoid in our implementation.

As the image inpainting problem is *under-constrained*, it can be solved in different visually plausible ways. So it is natural seeking for a method with the ability to keep diversity while generating the missing content. In (Zheng et al., 2019) authors have managed to combine *variational auto-encoder*, *VAE* (Kingma and Welling, 2013), and *GAN* (Goodfellow et al., 2014) techniques for *pluralistic image completion*.

Some learning-based methods have also adopted *structure guidance techniques*. In (Nazeri et al., 2019)

authors find edges of the incomplete image by *Canny edge detector* (Canny, 1986), then learn to complete the edges (with the guidance of the incomplete image) and guide the image completion with the completed edges. A similar approach is adopted by Xiong et al. (2019), where the authors complete foreground semantic contours in the incomplete image. In (Song et al., 2018b) authors use a semantic segmentation network Deeplabv3+ (Chen et al., 2018) to predict the segmentation map of the incomplete image, then (by the guidance of the incomplete image) learn to complete it and give as a guidance for image completion.

The most above-mentioned learning-based methods use vanilla convolutions which has the drawback of implicating irrelevant missing information in computations. To address this issue, Liu et al. (2018) introduced the *partial convolution layer*, which uses only the known region in every sliding window and updates the missing region. Despite partial convolutions achieve state-of-the-art results, they update the missing region mask in a *rule-based* manner, which leads to boundary artifacts and structural discontinuities. Later, Yu et al. (2019) introduced the *gated convolution layer*, which allows the network to learn the way of mask updating and achieves *state-of-the-art* results. Gated convolutions have been further successfully modified to the family of *light weight gated convolutions* (Yi et al., 2020), which helps the authors gain drastic improvements in efficient *high-resolution image inpainting*. Thus (Liu et al., 2018; Yu et al., 2019; Yi et al., 2020) show that distinguishing between valid and non-valid pixels is essential in the image inpainting problem, and we also adopt a similar approach in this work.

Motivated by the human workflow of image restoration, some recent works (Oh et al., 2019; Li et al., 2020a) fill the missing region iteratively from the boundary to the center instead of one-shot image completion. The success of these approaches shows the importance of propagating structural and textural continuities from the known region to the missing one. However, at each iteration these methods utilize *pixel-level attention mechanisms* which requires the *missing boundary* to be roughly filled before the processing. In some cases when this coarse estimation of the missing boundary content fails to continue the lines and curves, results may contain structural distortions (see Figures 5, 6, 7). In contrast with this, the *family of onion convolutions* is supplied with a *patch-level iterative filling* process, which does not require any coarse estimation of the missing boundaries and is designed to prevent structural discontinuities.

2.2 The Attention Mechanism

The attention mechanism, introduced by Bahdanau et al. (2015), has its widespread usage in deep learning after the paper by Vaswani et al. (2017).

A lot of work has benefited from the *long-range dependency capturing* ability of the attention.

Particularly in computer vision the attention modules are utilized in such tasks as *semantic segmentation* (Fu et al., 2019), *image matting* (Li and Lu, 2020; Li et al., 2020b), *neural style transfer* (Yao et al., 2019; Park and Lee, 2019), *image deblurring* (Suin et al., 2020; Zamir et al., 2021), *image inpainting* (Yu et al., 2018, 2019; Yi et al., 2020), etc..

Early work on adopting the self-attention to the vision tasks was (Hu et al., 2018), using channel-based attention. In (Zhang et al., 2019) authors managed to train a *GAN* combined with self-attention module for generating images. Recently, Bello et al. (2019) combined convolution operations with self-attention mechanism and outperforms state-of-the-art image classification and object detection approaches. These and many other works on attention use a *global attention mechanism* when all pixels participate in formulating all pixels (all-to-all). This global attention mechanism is computationally expensive, so can not be applied in low-level features of the image in neural networks. To overcome this limitation of computations, the *local attention mechanism* was introduced (Gregor et al., 2015; Luong et al., 2015), where every element looks only at its neighborhood. Ramachandran et al. (2019) built a *fully attentional* network for image classification and object detection performing state-of-the-art results. In (Cordonnier et al., 2020) authors proved an elegant theorem that claimed that *every convolutional layer is a special case of a multi-head attentional layer with a relative positional encoding.*

Now let us describe the core idea behind the attention mechanism. Let *information* is given by a sequence of d-dimensional vectors $V_1, V_2, \ldots, V_n \in \mathbb{R}^d$. Also let a new sequence $C_1, C_2, \ldots, C_m \in \mathbb{R}^d$ is needed to be generated. Then, at each time step t, for generating the information which will be held in C_t, *the attention mechanism* uses the *whole* information in $V_1, \ldots, V_n$ but with *different importances*. These *importances* are called *attention scores*. In other words, for each V_i its *importance value* α_{ti} is somehow estimated and the vector C_t is taken:

$$C_t = \sum_{i=1}^{n} \alpha_{ti} V_i. \tag{1}$$

In order to determine the attention scores α_{ti} the attention block takes two more sequences as input, namely the *key sequence* $K_1, K_2, \ldots, K_n$ and the *query sequence* $Q_1, Q_2, \ldots, Q_m$. The intuition behind the key and query sequences is that *for computing the importance of* V_i *in formulating* C_t *one can consider the similarity of the query* Q_t *to the key* K_i. Hence, in the classical attention mechanism the attention scores are determined in the following way:

$$\alpha_{ij} = Softmax(K \cdot Q_i)_j, \tag{2}$$

where K is the matrix with rows $K_1, \ldots, K_n$.

Notice that the attention scores are taken in such a way, that $\sum_{i=1}^{n} \alpha_{ti} = 1$ and $\alpha_{ti} \geq 0$. It means they can be interpreted as *probabilities of taking the values* $V_1, V_2, \ldots, V_n$. Moreover, obtaining C_t can be interpreted as *taking the expectation* over the conditional distribution $\{P(V_i \mid t) = \alpha_{ti},\ i = 1, \ldots, n\}$. Another way of determining C_t is *sampling from this conditional distribution*:

$$C_t = RandomSample(\{V_i\}, probs = \{\alpha_{ti}\}). \tag{3}$$

In the case when C_t is formulated by taking the expectation of the above-mentioned distribution, the attention mechanism is used to call *deterministic or soft attention*, while the version of random sampling is called *stochastic or hard attention* (Xu et al., 2015).

In the case of hard attention, some weights (which participate only in formulating sampling probabilities) can not be updated due to *differentiability issues w.r.t. these variables*, so Xu et al. (2015) maximize a lower bound of the marginal log-likelihood $\log p(\mathbf{y} \mid \mathbf{a})$ of observing the sequence of words $\mathbf{y}$ given image features $\mathbf{a}$. In this work, some *types of onion convolutions* can be interpreted as a kind of hard attention mechanism. However, these onion convolution modules have no such weights, which participate only in formulating sampling probabilities, so in contrast with (Xu et al., 2015) the onion convolutions do not need a trick like *REINFORCE* (Williams, 1992).

2.3 Patch-Based Techniques

Image manipulation techniques formulating their outputs based on *patch similarities or distances* are referred to as *patch-based techniques*.

Patch-based techniques are actively used in traditional computer vision algorithms for such tasks as *texture synthesis* (Efros and Leung, 1999; Wei and Levoy, 2000; Ashikhmin, 2001; Efros and Freeman, 2001; Kwatra et al., 2003), *style transfer* (Hertzmann et al., 2001; Efros and Freeman, 2001) and *image inpainting* (Criminisi et al., 2004; Sun et al., 2005; Xu and Sun, 2010). Later deep learning approaches to these tasks have also adopted patch-based techniques (Li and Wand, 2016; Chen and Schmidt, 2016; Yang et al., 2016; Liao et al., 2017; Song et al., 2018a; Yan et al., 2018; Yi et al., 2020). Our method with onion convolutions is also in this manner, where a *hybrid approach* of deep learning, patch-based methods, and attention mechanisms is adopted for the *image inpainting problem*.

A representative of patch-based techniques is the *onion-peel patch-match*, based on (Efros and Leung, 1999) and introduced as the first step of the *onion convolution layer* in the conference version of this work (Navasardyan and Ohanyan, 2020). The onion-peel patch-match fills the missing region $M^1 = M$ iteratively

by filling at each step $t = 1, \ldots$ the boundary ∂M^t of the remaining missing region M^t. The filling process is done by replacing the missing parts of the $k_f \times k_f$-sized boundary patches by their matching patches from a known region. The detailed description of the onion-peel patch-match can be found in Section 3.

Another representative of the patch-based methods is the *style-swap* algorithm (Chen and Schmidt, 2016), designed for the *arbitrary style transfer* task. The style-swap takes the *encoded features* of a style and a content images and produces the feature of the *stylized* image. The final stylized image is obtained by passing this feature to a pretrained decoder. The style-swap replaces $k \times k$ patches in the content feature with their matching patches from the style feature and aggregates the overlapping parts by averaging. More precisely, let C, S be the content and style images respectively, $\Phi(C), \Phi(S) \in \mathbb{R}^{H \times W \times D}$ be their features obtained by a convolutional encoder. The style-swap operation consists of the following steps.

- Extract the set of all $k \times k$ patches from both $\Phi(C)$ and $\Phi(S)$, denote these sets by $\{\phi_i(C)\}_{i=1}^{HW}$, $\{\phi_i(S)\}_{i=1}^{HW}$ respectively.
- For each content patch $\phi_i(C) \in \mathbb{R}^{k \times k \times D}$ find the most similar patch among the patches $\phi_j(S) \in \mathbb{R}^{k \times k \times D}$:

$$\phi_i^{ss}(C,S) = argmax\ \phi_j(S) \frac{< \phi_i(C), \phi_j(S) >}{||\phi_i(C)|| \cdot ||\phi_j(S)||}. \tag{4}$$

- In the content feature $\Phi(C)$ replace the patch $\phi_i(C)$ with the patch $\phi_i^{ss}(C,S)$.
- Due to simultaneously replacing all patches $\{\phi_i(C)\}_{i=1}^{HW}$ with their *matching patches* $\{\phi_i^{ss}(C,S)\}_{i=1}^{HW}$ some pixels will have multiple replacement candidates (coming from different patches to replace). One just needs *to average* the candidate pixels to get the final replacement for a certain pixel.

2.4 The Connection Between Attention Mechanisms and Patch-Based Techniques

As we have already mentioned there is a connection between the patch-based methods and attention mechanisms. This connection leads us to the extension of our initial concept of onion convolution (Navasardyan and Ohanyan, 2020) to the *family of onion convolutions*. In this section we discuss the similarity between the attention mechanism and two patch-based techniques, namely *style-swap* and *onion-peel patch-match*.

Both style-swap and onion-peel patch-match can be considered as *types of attention mechanisms*. Indeed in the case of style-swap a *kind of hard attention* can be obtained if one takes

- the patch-size $k = 1$;
- the query sequence as normalized $k \times k$ patches (pixels, since $k = 1$)

$$Q_i = \frac{\phi_i(C)}{||\phi_i(C)||}, \quad i = 1, 2, \ldots, HW, \tag{5}$$

- the key and value sequences as

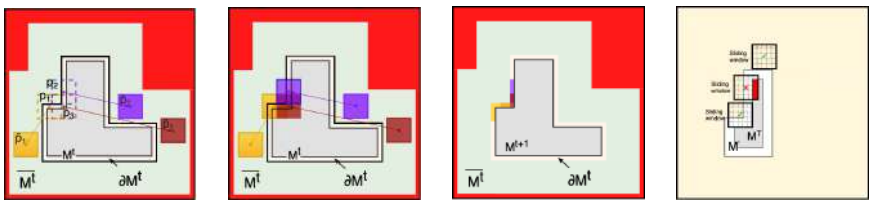
$$K_j = V_j = \frac{\phi_j(S)}{||\phi_j(S)||}, \quad j = 1, 2, \ldots, HW, \tag{6}$$

- the attention scores $\alpha_{ij} = \mathbb{1}_{j=j^*(i)}$, where

$$j^*(i) = argmax\ j < Q_i, K_j >, \tag{7}$$

and $\mathbb{1}$ is the *indicator function*.

This connection may lead to some modifications in both, the attention and the style-swap mechanisms. For example, like (Yi et al., 2020), one may come with the idea for the attention mechanism to compute



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

Figure 2: Onion convolution. First we perform onion-peel patch-match in the following way: for each iteration $t = 1,\dots,T$ 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 $\overline{M^t}$ (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 $p_1, p_2, p_3, \dots$ are replaced with their corresponding $k_f \times k_f$ patches centered in $\hat{p}_1, \hat{p}_2, \hat{p}_3, \dots$. (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 updating the remaining missing region M^T 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 updating the remaining missing region $M^T \mapsto M'$ by the Eq. 18.

patch-level similarities instead of pixel-level. Or for the style-swap instead of taking the most similar patch take one randomly with the probabilities

$$Softmax(\varphi_{i1}, \varphi_{i2}, \dots, \varphi_{i,HW}),\ \varphi_{ij} = \left(\frac{< \phi_i(C), \phi_j(S) >}{||\phi_i(C)|| \cdot ||\phi_j(S)||} \right). \tag{8}$$

For the onion-peel patch-match the situation is similar. For each iteration $t = 1, 2, \dots, T$ a kind of attention mechanism can be obtained if one takes[1]

- the patch-size $k_f = 1$,
- the query sequence as $Q_p = patch^{k_m}_{X^t}(p)$ for all positions p such that $M^t_p = 1$,
- the key sequence as $K_{\hat{p}} = patch^{k_m}_{X^t}(\hat{p})$ for all positions $\hat{p}$ such that $\overline{M^t}_{\hat{p}} = 1$,
- the value sequence as $V_{\hat{p}} = X^t_{\hat{p}}$ for all positions $\hat{p}$ such that $\overline{M^t}_{\hat{p}} = 1$,
- the attention scores $\alpha_{p\hat{p}} = P_{p\hat{p}}$.

Emphasize that depending on taking the expectation or random sampling in the onion-peel patch-match, one will obtain the similarity either with the *soft* or with the *hard* attention mechanisms.

As we have mentioned earlier, the connection between the attention mechanism and the onion-peel patch-match leads us to the extension of the concept of *onion convolution layer* (Navasardyan and Ohanyan, 2020). This extension is obtained by considering various definitions of *similarities between patches* and *soft or hard versions* of the attention mechanism.

Thus, being hybrid modules, our onion convolutions inherit the *feature-continuity preserving* property from the patch-based technique (onion-peel patch-match) and the ability of *capturing long-range pixel dependencies* from the attention mechanism by so satisfying both $(C2)$ and $(C3)$ conditions.

3 Approach

We start this section with a detailed description of the family of onion convolution modules and hold a discussion on implementation. Then the architecture of the proposed network and the loss functions are presented.

[1] For notations and other details, please, see Section 3.

3.1 The Family of Onion Convolution Modules

Human criteria for high quality image inpainting lead some image restoration experts/artists to develop their workflow for manual image inpainting. A study on such workflows was conducted by *Bertalmio et al.* (Bertalmío et al., 2000) and summarized to the following steps for manual image inapainting[2]: "(1.) The global picture determines how to fill in the gap, the purpose of inpainting being to restore the unity of the work; (2.) The structure of the area surrounding Ω is continued into the gap, contour lines are drawn via the prolongation of those arriving at $\partial\Omega$; (3.) The different regions inside Ω, as defined by the contour lines, are filled with color, matching those of $\partial\Omega$; and (4.) The small details are painted (e.g. little white spots on an otherwise uniformly blue sky): in other words, 'texture' is added".

It is worth noting that the family of onion convolutions also follows this workflow in some sense. Indeed, at each iteration t for filling the boundary ∂M^t of the missing region the onion convolutions look at the sufficiently large neighborhood of the hole in order to *"see the global picture"*. The steps (2.),(3.), and (4.) are unified and implicitly performed after, due to the patch-based nature of the onion convolutions.

The principle of the onion convolutions is illustrated in Fig. 2 and is summarized in Alg. 1. Each onion convolution layer takes two main 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).

Each onion convolution block is composed of three stages: *onion-peel patch-match*, *convolution*, and *updating the missing region*. The blocks differ from each other only at the stage of *onion-peel patch-match*. Below we describe each stage in detail.

3.1.1 Onion-Peel Patch-Match

As it is crucial for the family of onion convolutions to satisfy the condition $(C3)$, i.e. to preserve structure and texture continuities while generating the missing region, we adopt a *patch-based technique onion-peel patch-match*. The *onion-peel patch-match* is motivated by the *patch-based texture synthesis* algorithm described in (Efros and Leung, 1999) and *iteratively* fills the missing region of X, initially taking $X^1=X, M^1=M$. For each iteration $t=1,\dots,T$, the boundary ∂M^t of the missing region M^t is filled, resulting in a tensor X^{t+1} with a missing region, indicated by $M^{t+1}=M^t-\partial M^t$. This iterative process allows *gradually prolongations* of lines and curves entering the missing region by so *preventing feature discontinuities*. Thus at first we need to determine the boundary ∂M^t. 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:

$$\partial M^t = M^t - erode(M^t, 3). \tag{9}$$

For simplicity in the further reading, we indicate the *spatial locations by single index* instead of double index (for rows and columns), e.g. for the c-dimensional pixels of the tensor $X^t\in\mathbb{R}^{H\times W\times c}$ the notations $\{X^t_p \mid p=1,2,\dots HW\}$ will be used instead of the notations $\{X^t_{ij} \mid i=1,2,\dots,H, j=1,2,\dots,W\}$.

For each location p in the boundary ∂M^t (i.e. for such $p=1,\dots,HW$ that $\partial M^t_p=1$) in order to fill the missing pixel X^t_p let us consider its conditional distribution given the known part of its $k_m\times k_m$ neighborhood, where $k_m\in\mathbb{N}$ is a hyper-parameter.

More precisely, let $P(X^t_p \mid X^t_{p_1}, X^t_{p_2},\dots,X^t_{p_n})$ be the *conditional distribution* of the unknown pixel X^t_p given *the known neighboring pixels* $X^t_{p_1}, X^t_{p_2},\dots,X^t_{p_n}$, i.e. the locations p_i are all locations in the $k_m\times k_m$-sized neighborhood of p such that $M^t_{p_i}=0$.

After the conditional distributions $P(X^t_p \mid X^t_{p_1},\dots,X^t_{p_n})$ are known for each boundary location p, one can follow the patch-based method from (Efros and Leung, 1999) and take for X^t_p a randomly generated sample from its conditional distribution.

[2]here the authors denote by Ω the region to be inpainted and by $\partial\Omega$ its boundary

Another way to estimate the missing pixel X_p^t (motivated by the connection between patch-based techniques and attention mechanisms) is taking the *conditional expectation* $\mathbb{E}[X_p^t \mid X_{p_1}^t, \ldots, X_{p_n}^t]$. We explore this approach in our family of onion convolutions and provide a study in Sec. 4.

To model the distribution $P(X_p^t \mid X_{p_1}^t, \ldots, X_{p_n}^t)$ one needs to determine the *possible values* X_p^t can take. Since the condition $(C3)$ is expected to be satisfied by continuously propagating structural and textural information from the known region to the unknown, it is natural to assume that the information to fill in the pixel X_p^t is present in the known part of the tensor X^t. Instead of searching for this information in the whole known part of X^t we will consider *a sufficiently large* neighborhood of the missing region M^t:

$$\overline{M^t} = dilate(M^t, dil) - M^t, \tag{10}$$

where the hyper-parameter $dil \in \mathbb{N}$ controls how large the neighborhood $\overline{M^t}$ can be. Experiments show that the information of X^t in the region $\overline{M^t}$ is sufficient to *recover the missing information* in the pixel X_p^t. Hence we may assume that the pixel X_p^t may take the possible values $X_{\hat{p}}^t$, where the locations $\hat{p}$ are from the known region $\overline{M^t}$, i.e. $\overline{M^t}_{\hat{p}} = 1$.

Thus, similar with (Efros and Leung, 1999) we consider the conditional distribution $P(X_p^t \mid X_{p_1}^t, \ldots, X_{p_n}^t)$ as a *categorical distribution*, where X_p^t can take the values $\{X_{\hat{p}}^t \mid \overline{M^t}_{\hat{p}} = 1\}$ with some probabilities we denote by $P_{p\hat{p}}$.

In order to define the probabilities $P_{p\hat{p}}$ let us consider the $k_m \times k_m$ $(k_m > 1)$ patches centered at the locations p and $\hat{p}$, and introduce some notations. Let $\mathscr{T} \in \mathbb{R}^{H \times W \times c}$ be a tensor, k be a positive integer and $p \in \{1, 2, \ldots, HW\}$ be a position of a pixel in $\mathscr{T}$. Then by $patch_{\mathscr{T}}^k(p)$ we denote the $k \times k$ *patch in* $\mathscr{T}$ *centered at* p.

Motivated by the *connection* with attention mechanisms, for each location p, $\partial M_p^t = 1$ to model the categorical distribution $\{P_{p\hat{p}} \mid \overline{M^t}_{\hat{p}} = 1\}$ we compute *similarities* $s_{p\hat{p}}$ between *the known part of* $patch_{X^t}^{k_m}(p)$ and the patches $\{patch_{X^t}^{k_m}(\hat{p}) \mid \overline{M^t}_{\hat{p}} = 1\}$ followed by *softmax*:

$$P_{p\hat{p}} = Softmax(\{s_{p\hat{p}} \mid \overline{M^t}_{\hat{p}} = 1\})_{\hat{p}}. \tag{11}$$

Hence, we get *higher probabilities for those* $X_{\hat{p}}^t$*, neighborhoods of which are more similar to the neighborhood of* X_p^t.

The various ways these similarities $s_{p\hat{p}}$ can be defined is discussed separately in Sec. 3.2. Now let's assume these similarities (hence the probabilities $P_{p\hat{p}}$) are known. In this case, as we have already mentioned, one can either take a random sample

$$X_p^t = RandomSample(\{X_{\hat{p}}^t\}, probs = \{P_{p\hat{p}}\}), \tag{12}$$

or take the conditional expectation

$$X_p^t = \mathbb{E}[X_p^t \mid X_{p_1}^t, \ldots, X_{p_n}^t] = \sum_{\hat{p}, \overline{M^t}_{\hat{p}} = 1} P_{p\hat{p}} X_{\hat{p}}^t. \tag{13}$$

In the onion convolution mechanism, a hyperparameter $k_f \in \mathbb{N}$ is introduced for filling the missing parts of $k_f \times k_f$ patches at once instead of filling one unknown pixel at a time.

So, after the probabilities $P_{p\hat{p}}$ are determined the onion convolution mechanism replaces $patch_{X^t}^{k_f}(p)$ by either random sampled patch

$$patch_{X^t}^{k_f}(p) = RandomSample(\{patch_{X^t}^{k_f}(\hat{p})\}, probs = \{P_{p\hat{p}}\}), \tag{14}$$

or the expected patch

$$patch^{k_f}_{X^t}(p) = \mathbb{E}[patch^{k_f}_{X^t}(p) \mid X^t_{p_1}, \ldots, X^t_{p_n}] = \sum_{\hat{p}, \overline{M^t}_{\hat{p}}=1} P_{p\hat{p}} patch^{k_f}_{X^t}(\hat{p}). \tag{15}$$

Notice that if $k_f > 1$, and the replacements of patches are done simultaneously for all boundary locations p, the algorithm will end up with multiple candidates for missing pixels X^t_p. In the onion convolution algorithm these candidates are averaged to fill the pixels X^t_p for each boundary location p (see Fig. 2 (c) as well as Alg. 1, lines $15-17$).

After replacing patches centered in all boundary points a tensor $\hat{X}^t$ is obtained. The onion-peel patch-match takes the tensor X^{t+1} for the next iteration:

$$X^{t+1} = (1 - \partial M^t) \odot X^t + \partial M^t \odot \hat{X}^t. \tag{16}$$

As the boundary pixels in X^{t+1} are filled, the missing region M^t is also updated:

$$M^{t+1} = M^t - \partial M^t. \tag{17}$$

The onion-peel patch-match does this procedure for the iterations $t = 1, 2, \ldots, T$ resulting in a tensor we denote by O.

The definition of the similarities $s_{p\hat{p}}$, and taking either the expectation or random sampling from the distributions $P(X^t_p \mid X^t_{p_1}, \ldots, X^t_{p_n})$ make the variations between the types of onion convolutions which will be discussed later in Subsection 3.2.

3.1.2 Convolution and Updating the Missing Region

After the onion-peel patch-match is performed resulting in the tensor O, convolution with a kernel size $k_c \times k_c$ is applied to the tensor O. Let's denote the resulting tensor by C. As shown in Fig. 2 (d), 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 the results of convolving in such sliding windows should be eliminated. To obtain the centers of such sliding windows, one can use the morphological dilation operation with the kernel size $k_c \times k_c$:

$$M' = dilate(M^T, k_c). \tag{18}$$

We refer to M' as the *updated missing region after the onion convolution.*

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

3.2 Defining the Similarities $s_{p\hat{p}}$

In this Subsection, various types of onion convolutions will be discussed forming a *family of onion convolutions.* As we have already mentioned the members of this family differs from each other by

- the ways of modeling the similarities $s_{p\hat{p}}$, hence categorical distributions $P(X^t_p \mid X^t_{p_1}, \ldots, X^t_{p_n})$,
- taking either the expectation or random sampling from these distributions (in order to find the best replacement for $patch^{k_f}_{X^t}(p)$).

Below we introduce four methods to define the similarities $s_{p\hat{p}}$. The first three methods of computing patch similarities are based on the euclidian distance between these patches.

Algorithm 1 Onion Convolution Pseudo-Code

Require: $X \in \mathbb{R}^{H\times W\times c}$, $M \in \{0,1\}^{H\times W}$, $T, k_m, k_f, k_c, dil \in \mathbb{N}$, $smpl \in \{True, False\}$

$X^1 \leftarrow X$
$M^1 \leftarrow M$
for $t = 1, \dots, T-1$ **do**
  $\partial M^t \leftarrow M^t - erode(M^t, 3)$ ▷ *the boundary of the hole*
  $\overline{M^t} \leftarrow dilate(M^t, dil) - M^t$ ▷ *a known neighborhood of the hole*
  $\hat{X} \leftarrow ZerosLike(X^t)$ ▷ *initializing an array of zeros with the size $H \times W \times c$*
  for $p = 1, \dots, HW$ such that $\partial M^t_p = 1$ **do**
    for $\hat{p} = 1, \dots, HW$ such that $\overline{M^t}_{\hat{p}} = 1$ **do**
      $s_{p\hat{p}} \leftarrow Similarity(patch^{k_m}_{X^t}(p), patch^{k_m}_{X^t}(\hat{p}))$ ▷ *see Sec. 3.1.1 for details*
    $P_p \leftarrow Softmax([s_{p\hat{p}} \mid \overline{M^t}_{\hat{p}} = 1])$ ▷ *modeling the distribution for X^t_p*
    if smpl **then**
      $patch \leftarrow RandomSample(\{patch^{k_f}_{X^t}(\hat{p})\}, probs = P_p)$ ▷ *see Eq. 14*
    else
      $patch \leftarrow Expectation(\{patch^{k_f}_{X^t}(\hat{p})\}, probs = P_p)$ ▷ *see Eq. 15*
    $patch^{k_f}_{\hat{X}}(p) \leftarrow patch^{k_f}_{\hat{X}}(p) + patch$ ▷ *patch replacement and summation*
  for $p = 1, \dots, HW$ such that $\partial M^t_p = 1$ **do**
    $patch^{k_f}_{\hat{X}}(p) \leftarrow patch^{k_f}_{\hat{X}}(p) / sum(patch^{k_f}_{\partial M^t}(p))$ ▷ *aggregation with averaging*
  $X^{t+1} \leftarrow (1 - \partial M^t) \odot X^t + \partial M^t \odot \hat{X}$
  $M^{t+1} \leftarrow M^t - \partial M^t$
$C \leftarrow Conv(X^T, k_c \times k_c)$ ▷ *a convolution with kernel size k_c*
$M' \leftarrow dilate(M^T, k_c)$ ▷ *keeping only valid-window convolution results*
$Result \leftarrow (C \odot M', M')$

As it is crucial for onion convolutions to satisfy the condition $(C1)$, for measuring the distance between $patch^{k_m}_{X^t}(p)$ and $patch^{k_m}_{X^t}(\hat{p})$, we use only *valid pixel positions* in $patch^{k_m}_{X^t}(p)$. More precisely, the normalized sum of squared distances between valid pixels in $patch^{k_m}_{X^t}(p)$ and corresponding pixels in $patch^{k_m}_{X^t}(\hat{p})$ is considered:

$$d_{p\hat{p}} = \frac{||(patch^{k_m}_{X^t}(p) - patch^{k_m}_{X^t}(\hat{p})) \odot patch^{k_m}_{\overline{M^t}}(p)||_2^2}{sum(patch^{k_m}_{\overline{M^t}}(p))}. \tag{19}$$

3.2.1 Taking the Best Matching Patches

This approach of determining the similarities $s_{p\hat{p}}$ (hence the probabilities $P_{p\hat{p}}$) in essence is the method initially described in the conference version of this work (Navasardyan and Ohanyan, 2020). First, we consider the closest (in terms of the patch distance described above) patches of $patch^{k_m}_{X^t}(p)$ then treat these patches as *equiprobable possibilities* for the patch $patch^{k_m}_{X^t}(p)$. More precisely, the set of the closest patches is defined as follows:

$$\Omega^{\varepsilon}(p) = \{patch^{k_m}_{X^t}(\hat{p}) \mid \overline{M^t}_{\hat{p}} = 1, d_{p\hat{p}} \leq (1+\varepsilon)d^*\}, \tag{20}$$

where d^* is the minimal distance among all patches

$$d^* = \min_{\hat{p}, \overline{M^t}_{\hat{p}}=1} d_{p\hat{p}} \tag{21}$$

and ε is a hyper-parameter.

After the minimal distance d^* is computed, one can take the similarities

$$s_{p\hat{p}} = \begin{cases} 1 & \text{if } patch^{k_m}_{X^t}(\hat{p}) \in \Omega^{\varepsilon}(p) \\ -\infty & \text{otherwise} \end{cases}, \tag{22}$$

hence, after applying the *softmax* operation, one will get

$$P_{p\hat{p}} = \begin{cases} \frac{1}{|\Omega^{\varepsilon}(p)|} & \text{if } patch_{X^t}^{k_m}(\hat{p}) \in \Omega^{\varepsilon}(p) \\ 0 & \text{otherwise} \end{cases} . \tag{23}$$

We will shortly call this method of choosing similarities by *BestMatchSim*.

3.2.2 Turning Distances into Similarities by the Function $f(x) = -x$

One can turn the distances $d_{p\hat{p}}$ into similarities by a decreasing function. In this approach the function $f(x) = -x$ is chosen, and the similarities $s_{p\hat{p}} = -d_{p\hat{p}}$ are considered. Hence, we get

$$P_{p\hat{p}} = Softmax(\{-d_{p\hat{p}} \mid \overline{M^t}_{\hat{p}} = 1\})_{\hat{p}}. \tag{24}$$

We will shortly call this method of choosing similarities by *OppositeSim*.

3.2.3 Turning Distances into Similarities by the Function $f(x) = 1/(1+x)$

Here is the same situation as in the previous case but for the function $f(x) = 1/(1+x)$, so the similarities will be $s_{p\hat{p}} = 1/(1+d_{p\hat{p}})$. Hence, we get

$$P_{p\hat{p}} = Softmax\left(\left\{\frac{1}{1+d_{p\hat{p}}} \mid \overline{M^t}_{\hat{p}} = 1\right\}\right)_{\hat{p}}. \tag{25}$$

We will shortly call this method of choosing similarities by *InverseSim*.

3.2.4 Taking the Cosine Similarity

Here the most common approach for the attention modules is used. The similarities are determined by using the *cosine similarity* between patches. As when computing the patch distances, here also only the *valid pixel positions* in $patch_{X^t}^{k_m}(p)$ are used, i.e.

$$s_{p\hat{p}} = \left\langle \frac{patch_{X^t}^{k_m}(p) \odot patch_{\overline{M^t}}^{k_m}(p)}{||patch_{X^t}^{k_m}(p)||}, \frac{patch_{X^t}^{k_m}(\hat{p})}{||patch_{X^t}^{k_m}(\hat{p})||} \right\rangle. \tag{26}$$

We will shortly call this method of choosing similarities by *CosSim*.

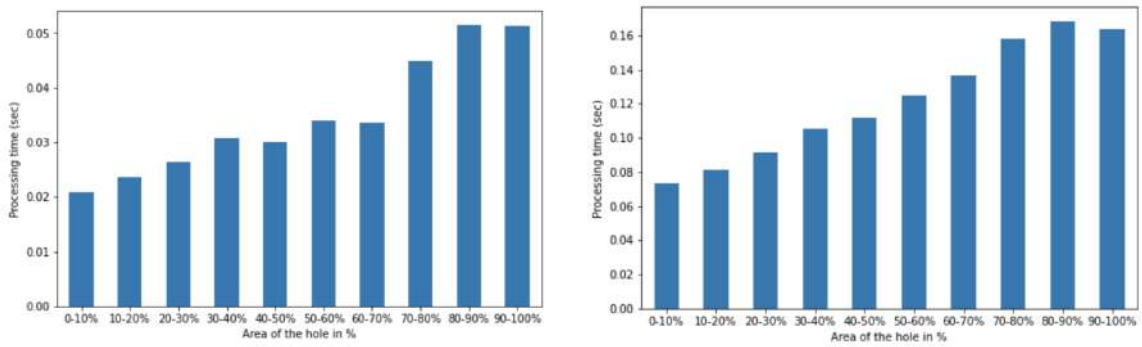
In summary, we get the *onion convolution family* consisting of *eight* types of onion convolutions presented in Table 1.

Table 1: The Family of Onion Convolutions. Each onion convolution can be obtained by choosing an appropriate method for filling the missing pixels (i.e. taking the expectation or random sampling from the patches in the known region) and choosing the method of defining the similarities $s_{p\hat{p}}$.

	Distances (BestMatchSim)	$-x$ (OppositeSim)	$1/(1+x)$ (InverseSim)	Cosine similarities (CosSim)
ES	EV-BestMatchSim	EV-OppositeSim	EV-InverseSim	EV-CosSim
RS	RS-BestMatchSim	RS-OppositeSim	RS-InverseSim	RS-CosSim

3.3 Discussion on Implementation

As can be noticed from Alg. 1 the naive implementation of the onion convolution blocks contains *3 nested for-loops* (lines 3, 7 and 8) which makes the algorithm computationally complex and time-consuming especially for high-resolution inputs. Since the onion-convolution layers contain *cosine-similarity computations* between



256×256 *resolution* 512×512 *resolution*

Figure 3: Inference time measurements of our method. The time computations are done on a single Nvidia Quadro RTX 8000 GPU. As can be noticed from the graphs, even for large missing regions on 512×512 resolution the network is still practical to use in real-world applications.

patches[3], the two nested loops in lines 7 and 8 (in Alg. 1) can be parallelized using convolutional layers as some existing works do when implementing mechanisms similar to the spatial global attention (Chen and Schmidt, 2016; Yu et al., 2018; Yan et al., 2018; Yi et al., 2020).

In general, it is done by extracting patches

$$\mathscr{P}^{k_m}_{X^t}(\overline{M^t}) = \{patch^{k_m}_{X^t}(p) \mid \overline{M^t}_p = 1\} \tag{27}$$

then convolving the tensor X^t with filters from $\mathscr{P}^{k_m}_{X^t}(\overline{M^t})$.

However this procedure contains dot product calculations also for pairs of patches, each of which is centered in the region $\overline{M^t}$. To avoid these redundant computations, we merely extract patches $\mathscr{P}^{k_m}_{X^t}(\partial M^t)$ and compute the dot products for each pair

$$(patch^{k_m}_{X^t}(p), patch^{k_m}_{X^t}(\hat{p})) \in \mathscr{P}^{k_m}_{X^t}(\partial M^t) \times \mathscr{P}^{k_m}_{X^t}(\overline{M^t}) \tag{28}$$

while keeping the computations parallel.

Our implementation is done purely with *TensorFlow* (Abadi et al., 2015), resulting in an end-to-end efficient pipeline for training and inference.

Moreover, the actual processing time of our network with onion convolutions is presented in Fig. 3 for different sizes of the missing region and resolutions 256×256 and 512×512. From the bar-plots in Fig. 3 can be noticed that even for large missing regions (more than $80-90\%$ of the whole image area) the processing time of high-resolution 512×512 images in average is quite small: less than 0.17 seconds, which makes our approach practical for real-world use-cases.

3.4 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 by Yu et al. (2019) and is based on the concept of *SN-GAN* (Miyato et al., 2018).

The generator network consists of two parts: *coarse* and *refinement networks*. The input to the generator is a tuple consisting of an image *with zeros in its missing region* and *a binary mask* indicating the missing region. The *coarse network* produces a coarse estimation of the inpainting result, which is passed to the *refinement network* to obtain the final high-quality result.

[3] a calculation of the euclidean distance also can be derived to calculations of dot products by the equality $||u-v||_2^2 = ||u||_2^2 - 2\langle u, v\rangle + ||v||_2^2$

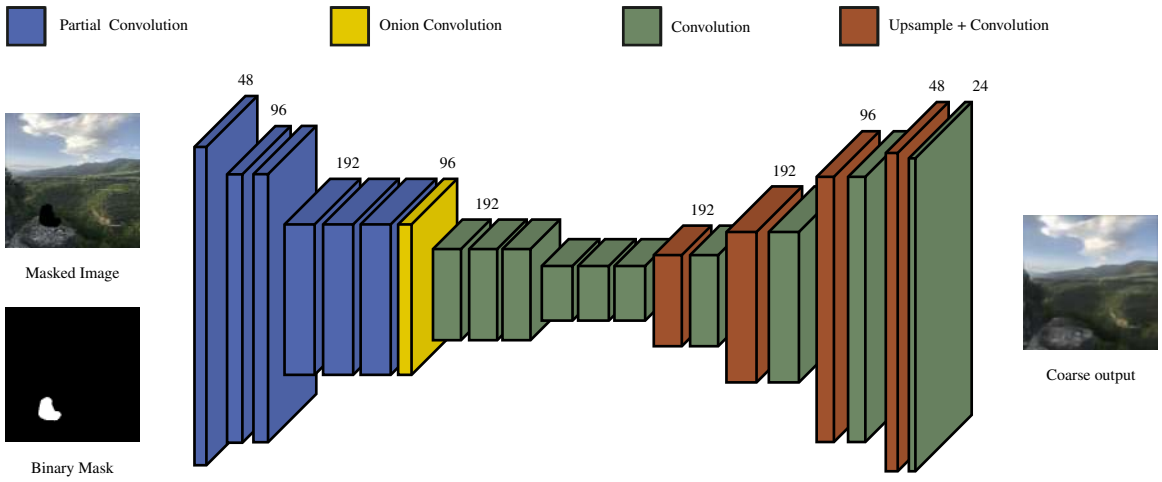


Figure 4: The architecture of our coarse network. After each convolution the corresponding activation is used. The onion convolution layer is used with parameters $k_m = 4, k_f = 2, dil = 8, k_c = 3, T = \infty$, where $T = \infty$ means that we iteratively complete the missing region until it is filled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

3.4.1 Coarse Network

Let I be an image, which is 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. 4. Six partial convolution layers with ELU (Clevert et al., 2015) activation are used at the beginning of the network to reduce the tensor sizes. Let's denote the output of the sixth partial convolution by X and the updated mask by M'. Due to boundary artifacts the partial convolution layer introduces, we do not consider the updated mask M', instead, we resize the initial missing region M to the size of X and refer to this new resized binary mask as the missing region indicator in the tensor X. Then one of the onion convolution modules is applied with parameters $k_m = 4, k_f = 2, dil = 8, k_c = 3, T = \infty$, where $T = \infty$ means that we continue the iterative process of the onion-peel patch-match until the missing region is filled (see Table 2). We have experimented with the hyper-parameters, with the position of the onion convolution module inside the network and found that 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 the output in the range $[-1,1]$. The detailed architecture of the coarse network can be found in Table 2.

3.4.2 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 a *refinement network*. The architecture of the refinement network is very similar to the refinement network used by Yu et al. (2019). The only difference is using vanilla convolutions instead of gated convolutions (this difference is discussed in our ablation study, see Section **??**).

3.5 Loss Functions

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

$$\mathscr{L} = \lambda_1 \mathscr{L}_1 + \lambda_2 \mathscr{L}_{ad} + \lambda_3 \mathscr{L}_p \ , \tag{29}$$

Table 2: Parameters of our coarse network. PConv_x denotes partial convolution layers (Liu et al., 2018). *ELU* denotes the *Exponential Linear Unit* (Clevert et al., 2015). Conv_x is a convolution block composed of a convolution and a non-linearity. ConvUp_x is a block composed of a nearest neighbor upsampling followed by a convolution and a non-linearity. All paddings are of the type "same".

Layer Name	Parameters
PConv_1	$filters = 48, ksize = 5, strides = 1, activation = ELU$
PConv_2	$filters = 96, ksize = 3, strides = 2, activation = ELU$
PConv_3	$filters = 96, ksize = 3, strides = 1, activation = ELU$
PConv_4	$filters = 192, ksize = 3, strides = 2, activation = ELU$
PConv_5	$filters = 192, ksize = 3, strides = 1, activation = ELU$
PConv_6	$filters = 192, ksize = 3, strides = 1, activation = ELU$
Onion_Conv	$d = 8, k_f = 2, k_m = 4, T = \infty, filters = 96,$ $k_c = 3, strides = 1, activation = ELU$
Conv_7	$filters = 192, ksize = 3, strides = 2, activation = ELU$
Conv_8	$filters = 192, ksize = 3, strides = 1, activation = ELU$
Conv_9	$filters = 192, ksize = 3, strides = 1, activation = ELU$
Conv_10	$filters = 192, ksize = 3, strides = 2, activation = ELU$
Conv_11	$filters = 192, ksize = 3, strides = 1, activation = ELU$
Conv_12	$filters = 192, ksize = 3, strides = 1, activation = ELU$
ConvUp_13	$filters = 192, ksize = 3, activation = ELU$
Conv_14	$filters = 192, ksize = 3, strides = 1, activation = ELU$
ConvUp_15	$filters = 192, ksize = 3, activation = ELU$
Conv_16	$filters = 192, ksize = 3, strides = 1, activation = ELU$
ConvUp_17	$filters = 96, ksize = 3, activation = ELU$
Conv_18	$filters = 96, ksize = 3, strides = 1, activation = ELU$
ConvUp_19	$filters = 48, ksize = 3, activation = ELU$
Conv_20	$filters = 24, ksize = 3, strides = 1, activation = ELU$
Conv_21	$filters = 3, ksize = 3, strides = 1$
Tanh	

where $\lambda_1, \lambda_2, \lambda_3$ are training hyperparameters. For optimizing the inpainting network G, the loss function $\mathscr{L}$ is minimized w.r.t. the generator's weights ω_G. At the same time, as there is an adversarial loss, $\mathscr{L}_{ad}$, 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.

3.5.1 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 Yu et al. (2019)):

$$\mathscr{L}_1 = ||I_c - I_{orig}||_1 + ||I_r - I_{orig}||_1 \ . \tag{30}$$

3.5.2 Adversarial Loss

Our discriminator gets the original images I_{orig} as real examples and composition images

$$I_{compos} = I_{orig} \odot (1 - M) + I_r \odot M \tag{31}$$

as fake examples. Since the discriminator belongs to the family of *PatchGAN*s, it outputs $3D$ tensors D_{real} and D_{fake}. As in (Yu et al., 2019), the *hinge loss* between the outputs of our discriminator is computed:

$$\mathscr{L}_{ad} = -\mathbb{E}[ReLU(1-D_{real}) + ReLU(1+D_{fake})] \ . \tag{32}$$

3.5.3 Perceptual Loss

We also use the *perceptual (content) loss* introduced in (Gatys et al., 2016), which minimizes the distance between the features of the original and the completed images obtained by the *VGG*-16 (Simonyan and Zisserman, 2015) network. Similar to (Liu et al., 2018), 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 perceptual loss is defined as follows:

$$\begin{aligned}\mathscr{L}_p = \sum_{i=1}^{3} \Big[&||MP_i(I_r) - MP_i(I_{orig})||_1 + \\ &||MP_i(I_{compos}) - MP_i(I_{orig})||_1 \Big] \ .\end{aligned} \tag{33}$$

Thus, the total loss $\mathscr{L}$ 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

In this section we describe the settings of our experiments with onion convolution models then perform a qualitative and a quantitative comparison with existing state-of-the-art approaches. Later a discussion on the onion convolution types is presented followed by discussions on processing high resolution images and error accumulation in the onion convolution module. Finally, an ablation study is done.

4.1 Implementation Details

The training experiments with onion convolution modules are performed on Places2 (Zhou et al., 2017) and Celeba-HQ (Karras et al., 2017) datasets. All the models are trained on the image resolution 256×256 without any augmentations, with batch size equal to 28, and using the ADAM optimizer (Kingma and Ba, 2015) with a learning rate of 10^{-4} and parameters $\beta_1 = 0.5$, $\beta_2 = 0.999$. *NVIDIA V100* and *NVIDIA Quadro RTX 8000* GPUs were used for all the trainings. The training period for each experiment was from 10 to 14 days.

To evaluate our models, we have sampled 20000 images from *Places2* (Zhou et al., 2017) test dataset and 3000 samples from *Celeba-HQ* (Karras et al., 2017). For each image, we have created random free-form masks as it was done in (Yu et al., 2019). 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. To show the generalization ability of our method, we evaluate it also on a test set of 3000 images randomly sampled from the Imagenet (Deng et al., 2009) validation set.

As it was mentioned earlier, the onion convolution family consists of eight block types. A comprehensive study on these onion convolution layers is provided in Sec. 4.3. Although they perform almost similarly (see Table 7), we pick up the one with the highest SSIM value to compare with other state-of-the-art methods and perform discussions in the further sections.

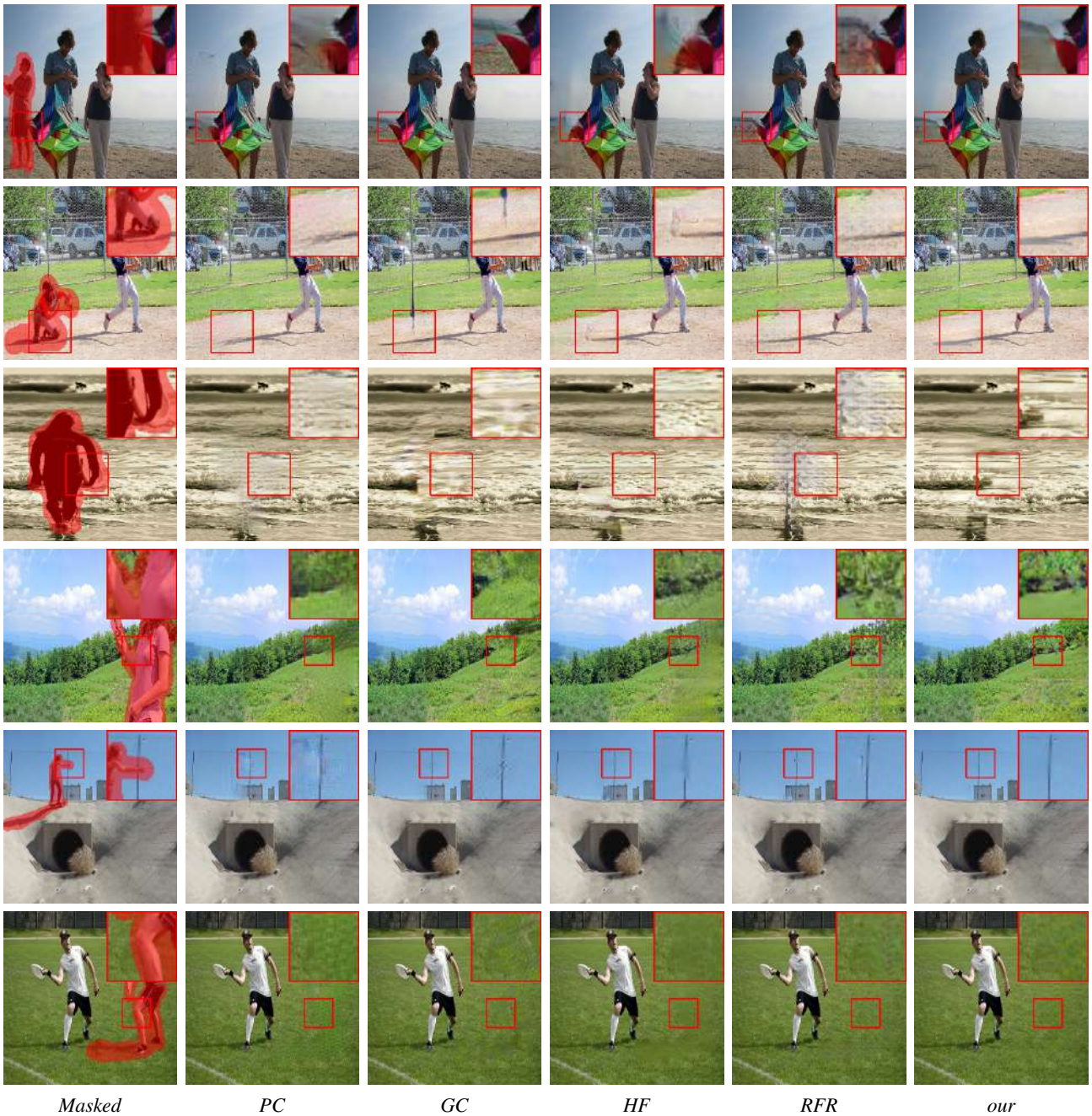


Figure 5: Comparisons of our method with PC (Liu et al., 2018), GC (Yu et al., 2019), HF (Yi et al., 2020) and RFR (Li et al., 2020a). Please see in color.

4.2 Comparison with State-of-the-Art Methods

In this subsection we compare our method with several learning-based state-of-the-art approaches: Gated Convolutions (GC) (Yu et al., 2019), Partial Convolutions (PC) (Liu et al., 2018), HiFill (HF) (Yi et al., 2020) and Recurrent Feature Reasoning (RFR) (Li et al., 2020a), as well as two representatives of patch-based traditional approaches: Patch-Match (PM) (Barnes et al., 2009) and Exemplar-Based Image Inpainting (EBII) (Criminisi et al., 2004).

For comparison on general scenes (such as Places2, Imagenet) we took the pretrained GC (Yu et al., 2019) and RFR (Li et al., 2020a) models from their official repositories. As there is no official implementation of the method PC (Liu et al., 2018), we made our own, which benefited a lot from `https://github.com/MathiasGruber/PConv-Keras`. For HF (Yi et al., 2020) we took the TensorFlow reimplementation from `https://github.com/duxingren14/Hifill-tensorflow`. The PM (Barnes et al., 2009) and EBII (Criminisi et al., 2004) algorithms were taken from the publicly available repositories `https://github.com/vacancy/PyPatchMatch` and `https://github.com/igorcmoura/inpaint-object-remover` respectively.

For comparison on face images we used Celeba-HQ (Karras et al., 2017) dataset and took the pretrained GC model (Yu et al., 2019) from its official repository. The models RFR (Li et al., 2020a), HF (Yi et al., 2020) and PC (Liu et al., 2018) were retrained on Celeba-HQ. For both qualitative (Sec. 4.2.1) and quantitative (Sec. 4.2.2) comparisons we consider our best model in terms of *SSIM*, the onion convolution with *cosine similarity and random sampling*, *Onion-RS-CosSim* (see Table 7).

4.2.1 Qualitative Comparison

Some visual results of our method and other state-of-the-arts can be found in Figures 5, 6, 7, and 8. As can be noticed from these figures, in general scenes PC (Liu et al., 2018) reconstructs semantics, but introduces some blur (e.g. rows 1 and 2 in Fig. 5), pattern (e.g. rows 3,4,6 in Fig. 5, columns 2,4 in Fig. 6) or sometimes does not preserve continuous lines (e.g. columns 1,4 in Fig. 6). While often keeping high frequencies, GC (Yu et al., 2019) sometimes introduces them excessively (e.g. rows 2,6 in Fig. 5, columns 1,5 in Fig. 7) does not keep image structures (e.g. rows 2,5 in Fig. 5, columns 1,3,4 in Fig. 6), or generates some strange artifacts (e.g. columns 2,5 in Fig. 6, column 4 in Fig. 7). HF (Yi et al., 2020) reconstructs semantics as well, but one can observe grey artifacts when reconstructing mask-near borders (e.g. column 2 in Fig. 6) or "copy-pasting" with low intensity (e.g. row 2 in Fig. 5). Moreover, it can be noticed that HF sometimes fails to keep the structure continuities (e.g. row 5 in Fig. 5, columns 1,2,3 in Fig. 6). Although RFR (Li et al., 2020a) propagates texture from the known region to the missing one (e.g. column 2 in Fig. 6) it sometimes fails to preserve structure continuities (e.g. columns 1,3 in Fig. 6) or introduce a non-realistic pattern (e.g. columns 1,2,3,5 in Fig. 7). The traditional approaches EBII (Criminisi et al., 2004) and PM (Barnes et al., 2009) sometimes greatly recover the textures and structures (e.g. column 2 in Fig. 6), however fail to generate the context in complex scenes (columns 1,3,4 in Fig. 6) or large holes (see Fig. 7). In contrast with these methods, our approach can successfully reconstruct detailed textures without introducing strange artifacts and coherently preserve structure continuities due to its property of *feature continuity propagation*. Moreover, due to its nature of iteratively filling the whole missing region from boundary to the center, our onion convolution module has a clear advantage in inpainting large missing regions (see Fig. 7).

Visual comparison on face images can be found in Fig. 8. As can be noticed, PC fails to generate the missing parts of face images. Although GC completes the faces in a semantically plausible way, sometimes it introduces distortion artifacts. RFR reconstructs face structures, however fails to generate the details such as eyes. In contrast with these methods the onion convolution model generates semantically meaningful face structures and details. Moreover, one can notice that our method recovers the eye color if one of the eyes is in the known region. We hypothesize this is due to the patch-based nature of the onion convolution operation.

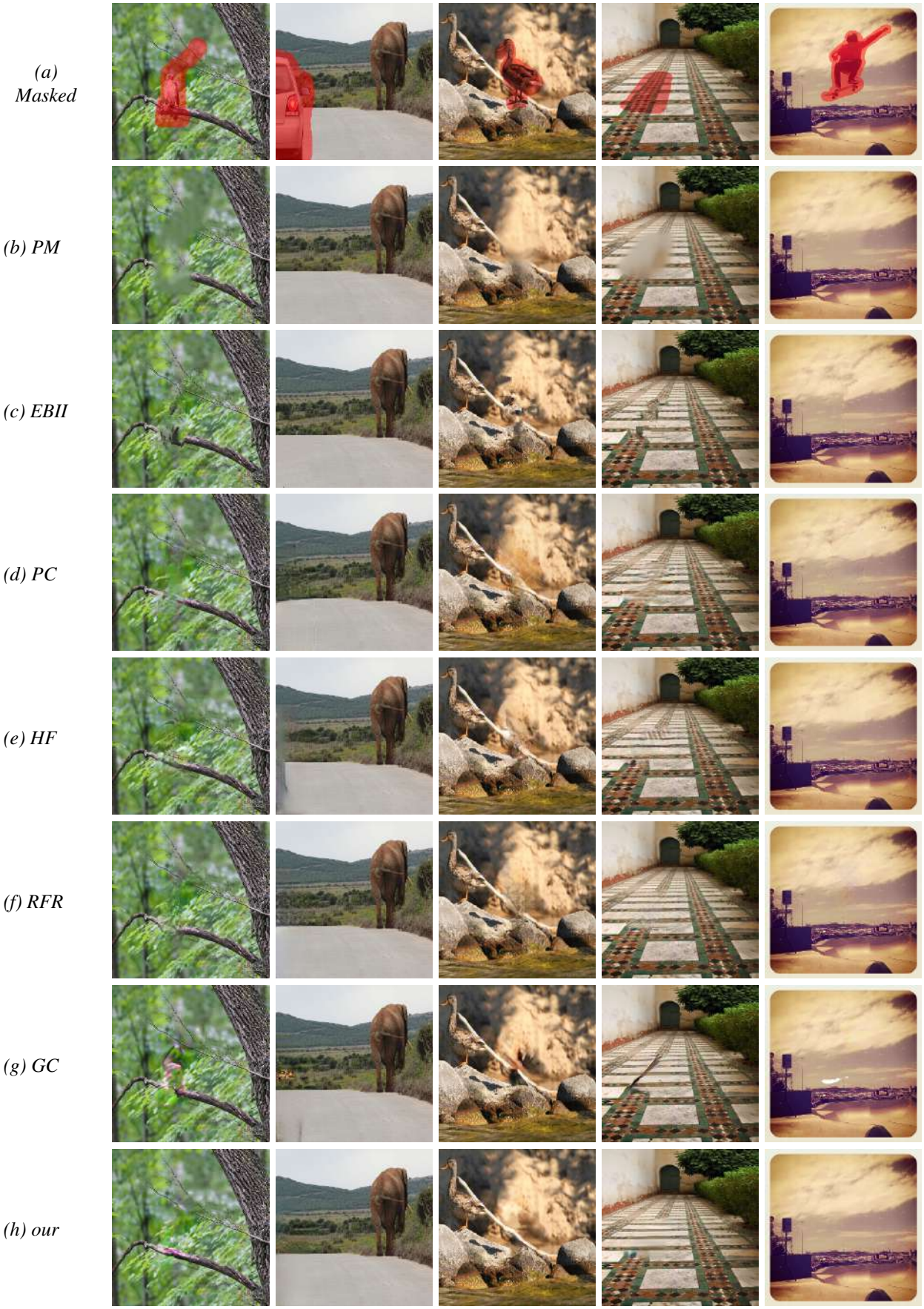


Figure 6: *(a)* Masked image. Comparisons of our method *(h)* with *(b)* PM (Barnes et al., 2009), *(c)* EBII (Criminisi et al., 2004), *(d)* PC (Liu et al., 2018), *(e)* HF (Yi et al., 2020), *(f)* RFR (Li et al., 2020a), *(g)* GC (Yu et al., 2019). Please see in color.

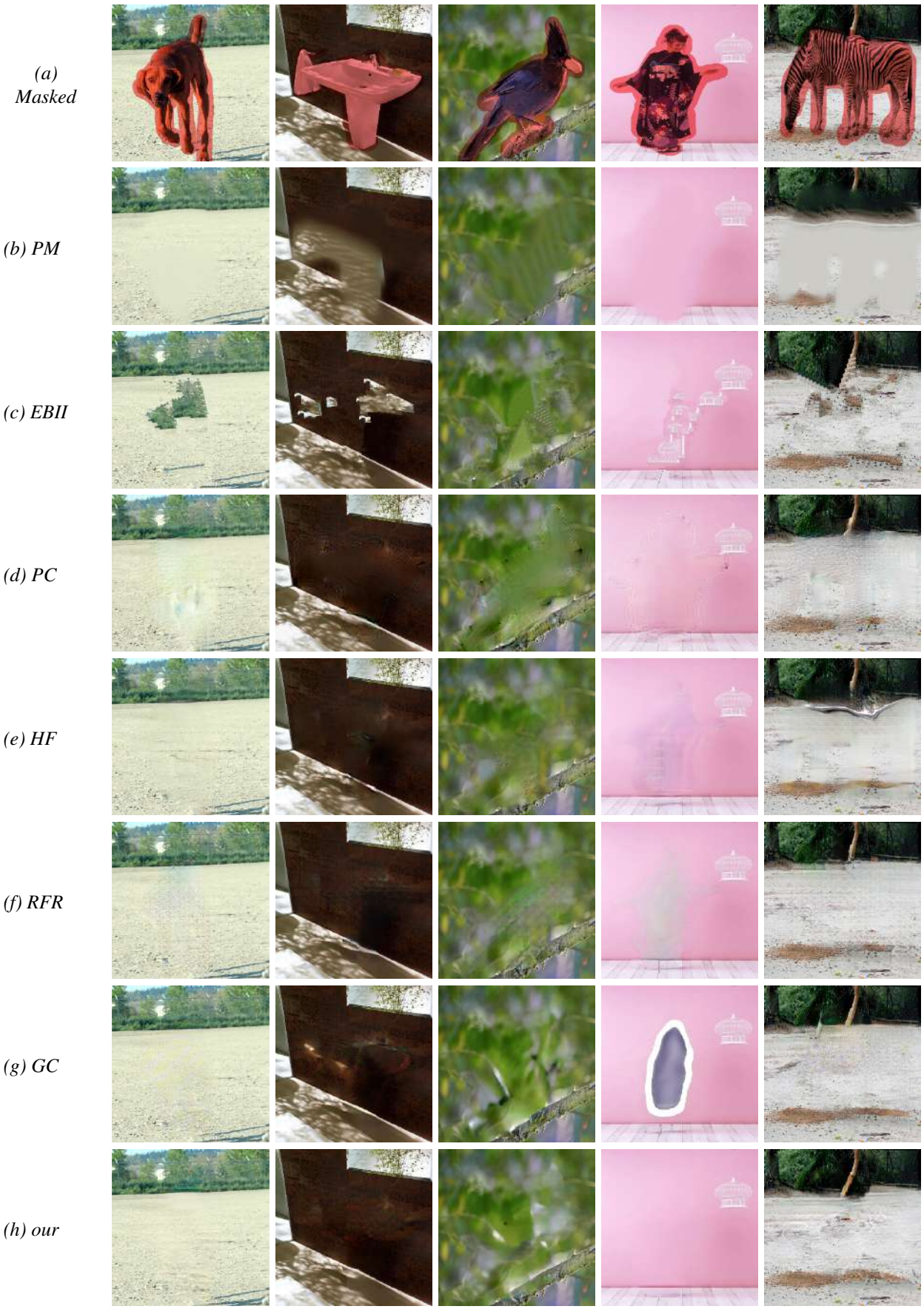


Figure 7: *(a)* Masked image. Comparisons of our method *(h)* with *(b)* PM (Barnes et al., 2009), *(c)* EBII (Criminisi et al., 2004), *(d)* PC (Liu et al., 2018), *(e)* HF (Yi et al., 2020), *(f)* RFR (Li et al., 2020a), *(g)* GC (Yu et al., 2019). Please see in color.

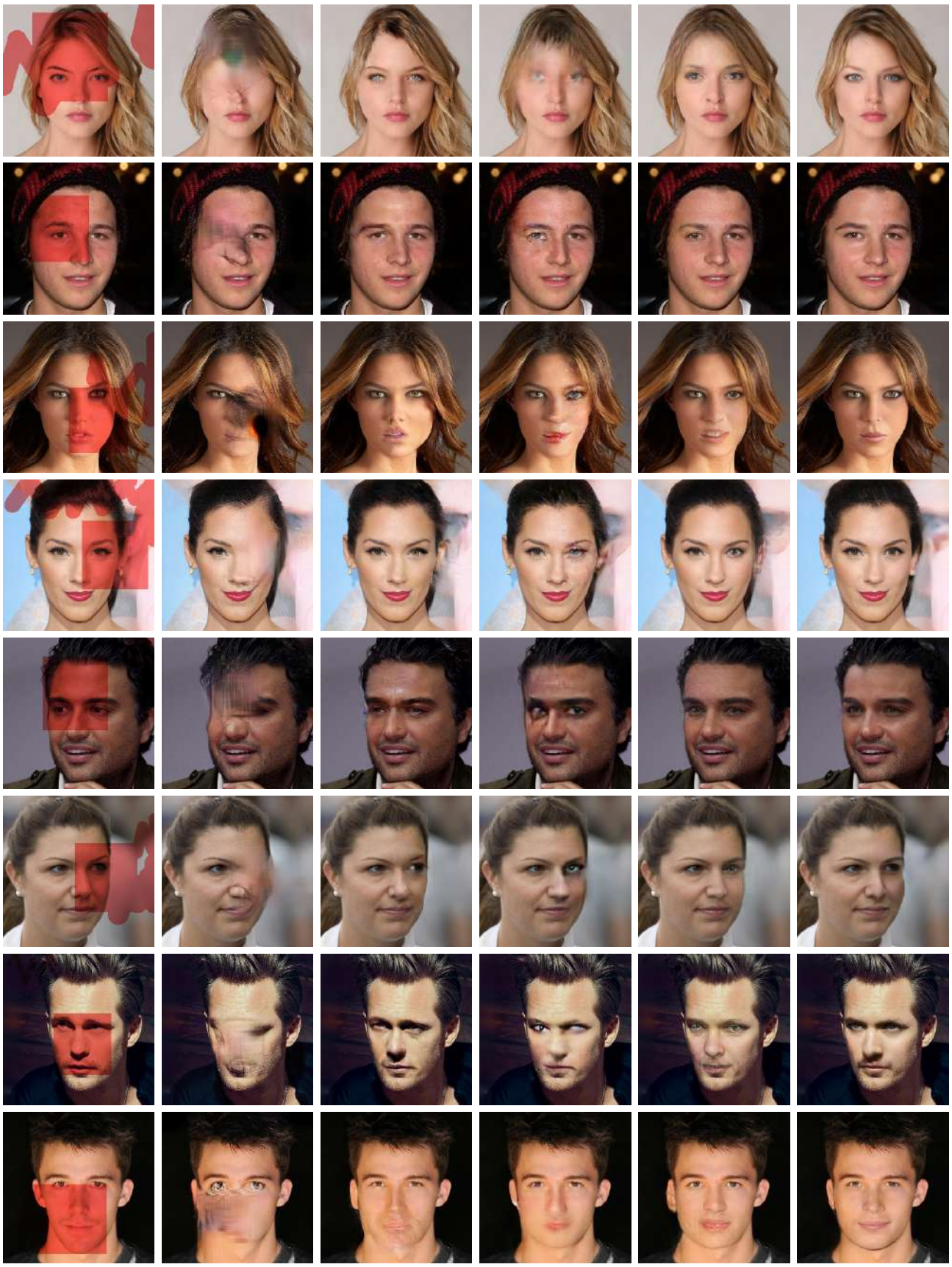


Figure 8: Comparisons of our method with PC (Liu et al., 2018), GC (Yu et al., 2019), HF (Yi et al., 2020) and RFR (Li et al., 2020a) on Celeba-HQ (Karras et al., 2017) dataset. Please see in color.

4.2.2 Quantitative Comparison

For a comprehensive comparison with other state-of-the-art methods we consider 2 *image domains* (general scenes and face images) with 3 *datasets* (Places2 (Zhou et al., 2017), Imagenet (Deng et al., 2009), and Celeba-HQ (Karras et al., 2017)) and evaluate the models with 5 *evaluation metrics*, namely *PSNR*, *SSIM* (Zhou Wang et al., 2004), *Mean Absolute Error (MAE)*, perceptual metric *LPIPS* (Zhang et al., 2018) (with linear configuration) with AlexNet (Krizhevsky et al., 2012) and VGG (Simonyan and Zisserman, 2015), and *Fréchet inception distance (FID)* (Heusel et al., 2017).

As we mentioned before, for the evaluations on Places2 and Celeba-HQ datasets we sampled 20000 and 3000 images respectively and created random free-form masks similar to (Yu et al., 2019). For the evaluation on Imagenet, we sampled 3000 images from its validation set.

Tables 3, 4, and 5 shows the comparison results on the datasets Places2, Imagenet, and Celeba-HQ respectively. On Places2 in terms of SSIM, MAE, and the FID score (which can measure how realistic the generated results are) our method outperforms the other six. In terms of PSNR, our method under-performs a bit RFR and HF, which can be due to the fact that these methods sometimes tend to provide blurry results or introduce a pattern (a similar observation can be made for Imagenet dataset also). In terms of LPIPS, our method under-performs GC with a negligible difference. On Imagenet dataset our method outperforms the other six in terms of SSIM, LPIPS and FID score, whereas slightly underperforms RFR and HF in terms of PSNR, and RFR in terms of MAE.

On face images, as can be noticed from Table 5, our method outperforms the other six in terms of all metrics besides PSNR with a small difference.

As the onion convolution modules are designed to iteratively fill the missing region from its boundary to the center, the models trained with these layers has an advantage to complete large missing regions in a visually plausible manner without introducing any artifacts or color inconsistencies. To show that, we perform an evaluation on the Imagenet testing set (with 3000 images) mentioned above and randomly generated masks indicated large missing regions, i.e. more than 60% of the whole image area should be filled. The results are presented in the Table 6, from which one can notice that our method outperforms the others in terms of all the metrics besides PSNR. In terms of PSNR our method slightly underperforms PC, the reason of which can be the blurry and patterny results of PC (see Fig. 7).

Table 3: The quantitative comparison of our method with PC (Liu et al., 2018) , GC (Yu et al., 2019) , HF (Yi et al., 2020), RFR (Li et al., 2020a), PM (Barnes et al., 2009), and EBII (Criminisi et al., 2004) on *Places2* 20000 testing dataset.

	PSNR ↑	*SSIM* ↑	*MAE* ↓	*LPIPS (AlexNet)* ↓	*LPIPS (Vgg)* ↓	*FID* ↓
PC	19.83	0.719	0.052	0.240	0.276	24.21
GC	19.79	0.782	0.042	**0.171**	**0.182**	9.84
HF	20.32	0.775	0.043	0.192	0.202	13.18
RFR	**20.67**	0.741	0.049	0.178	0.249	15.83
PM	17.844	0.735	0.066	0.288	0.261	14.58
EBII	18.387	0.749	0.051	0.206	0.202	16.60
Onion-RS-CosSim	20.111	**0.786**	**0.041**	0.173	0.186	**7.67**

4.3 A Study on the Family of Onion Convolutions

As we have mentioned earlier, the onion convolution family members differ from each other by methods of

- modeling the *categorical distribution probabilities* $\{P_{p\hat{p}} \mid \overline{M^t}_{\hat{p}} = 1\}$ and
- obtaining the replacement for $patch^{k_f}_{X^t}(p)$ by either taking the expectation or random sampling from the patches $\{patch^{k_f}_{X^t}(\hat{p}) \mid \overline{M^t}_{\hat{p}} = 1\}$ with above-mentioned probabilities.

Table 4: The quantitative comparison of our method with PC (Liu et al., 2018) , GC (Yu et al., 2019) , HF (Yi et al., 2020), RFR (Li et al., 2020a), PM (Barnes et al., 2009), and EBII (Criminisi et al., 2004) on *Imagenet* 3000 testing dataset.

	PSNR ↑	*SSIM* ↑	*MAE* ↓	*LPIPS (AlexNet)* ↓	*LPIPS (Vgg)* ↓	*FID* ↓
PC	19.868	0.701	0.054	0.254	0.281	50.86
GC	19.456	0.756	0.046	0.194	0.198	31.54
HF	20.261	0.763	0.043	0.205	0.207	32.88
RFR	**20.761**	0.767	**0.04**	0.189	0.201	32.46
PM	17.982	0.729	0.064	0.292	0.262	39.09
EBII	18.35	0.733	0.052	0.218	0.21	37.6
Onion-RS-CosSim	20.204	**0.771**	0.041	**0.178**	**0.189**	**26.32**

Table 5: The quantitative comparison of our method with PC (Liu et al., 2018) , GC (Yu et al., 2019) , HF (Yi et al., 2020), RFR (Li et al., 2020a), PM (Barnes et al., 2009), and EBII (Criminisi et al., 2004) on *Celeba-HQ* 3000 testing dataset.

	PSNR ↑	*SSIM* ↑	*MAE* ↓	*LPIPS (AlexNet)* ↓	*LPIPS (Vgg)* ↓	*FID* ↓
PC	20.809	0.774	0.047	0.172	0.208	29.97
GC	24.231	0.844	0.025	0.098	0.13	6.44
HF	23.675	0.805	0.033	0.132	0.167	10.68
RFR	**24.481**	0.837	0.027	0.099	0.139	12.55
PM	17.653	0.752	0.069	0.266	0.269	36.58
EBII	18.531	0.761	0.049	0.215	0.217	93.60
Onion-RS-CosSim	24.193	**0.851**	**0.024**	**0.091**	**0.12**	**5.39**

Table 6: The quantitative comparison of our method with PC (Liu et al., 2018) , GC (Yu et al., 2019) , HF (Yi et al., 2020), RFR (Li et al., 2020a), PM (Barnes et al., 2009), and EBII (Criminisi et al., 2004) on *Imagenet* 3000 testing dataset for large ($> 60\%$ of the image area) missing regions.

	PSNR ↑	*SSIM* ↑	*MAE* ↓	*LPIPS (AlexNet)* ↓	*LPIPS (Vgg)* ↓	*FID* ↓
PC	**15.436**	0.497	0.111	0.438	0.472	106.31
GC	14.436	0.517	0.112	0.388	0.404	77.99
HF	14.912	0.507	0.113	0.427	0.431	95.39
RFR	15.175	0.535	0.104	0.389	0.412	87.40
PM	13.346	0.505	0.144	0.523	0.488	76.48
EBII	13.826	0.483	0.123	0.427	0.42	98.19
Onion-RS-CosSim	15.186	**0.545**	**0.103**	**0.381**	**0.405**	**67.28**

By these two points *eight types* of onion convolutions are obtained and presented in Table 1.

As it has been discussed in Section 3, the categorical distribution probabilities are modeled by the *softmax* function applied on the similarities $s_{p\hat{p}}$ between patches $patch^{k_m}_{X^t}(p)$ and $patch^{k_m}_{X^t}(\hat{p})$. We consider four ways of defining these similarities $s_{p\hat{p}}$ and refer them shortly as mentioned in Table 1: *BestMatchSim, OppositeSim, InverseSim* and *CosSim*.

Qualitative and quantitative comparisons of the image inpainting methods with different types of onion convolutions can be found in Fig. 9 and Table 7 respectively.

It can be noticed from Table 7 that there is no explicit leader between the onion convolution types. The same situation is with Fig. 9.

However, some observations can be made:

- When looking at Table 7, in comparison with *CosSim*, distance-based similarity choosing methods *OppositeSim* and *InverseSim* perform better in the case of taking the *expectation (EV)* and perform disadvantageous in the case of *random sampling (RS)*.

  We have a hypothesis that the reason for such observation is that *cosine similarities are not sensitive to brightness changes*. This causes some patches with the same structure but different intensities to be

treated as similar patches, which may lead to messy results in the case of taking the expected value.

- *Taking the expectation (EV)* performs better on textures (see Fig. 9 columns 1,2,4), whereas the *random sampling (RS)* performs better on scenes with complex structures (see Fig. 9 columns 3,5).

  Due to our hypothesis, the reason is that in the case of texture patches the amount of similar patches is much more than in the case of structural patches. It causes the expected value to be a better estimator of the missing patch in the case of inpainting textures than in the case of inpainting complex structures.

Table 7: The quantitative comparison between the members of the onion convolution family.

	PSNR ↑	*SSIM* ↑	*MAE* ↓	*LPIPS (AlexNet)* ↓	*LPIPS (Vgg)* ↓
Onion-EV-BestMatchSim	19.869	0.780	0.042	0.174	0.187
Onion-EV-OppositeSim	**20.216**	0.781	0.041	0.174	0.187
Onion-EV-InverseSim	19.964	0.785	0.041	0.172	**0.184**
Onion-EV-CosSim	20.105	0.781	0.041	0.173	0.187
Onion-RS-BestMatchSim	19.806	0.784	0.042	0.172	**0.184**
Onion-RS-OppositeSim	20.046	0.779	0.041	**0.171**	0.188
Onion-RS-InverseSim	20.203	0.784	**0.040**	0.175	0.187
Onion-RS-CosSim	20.111	**0.786**	0.041	0.173	0.186

4.4 Study on Error Propagation in Onion Convolution

As we mentioned earlier the onion convoluition block was designed to satisfy the criterion of *preserving structure continuities*. Based on the qualitative analyses discussed in Sec. 4.2.1 (see also Figures 5, 6, 7, 8) one can conclude that the onion convolution layer possesses an ability of preserving feature continuities. On the other hand the iterative nature of onion-peel patch-match may cause an error propagation in the case of large masks (i.e. large number of iterations) and introduce structure distortions in the generated region. To study this phenomenon we consider a simple example of a straight line recovery (see Fig. **??**). As can be noticed from Fig. **??**, our method is able to successfully reconstruct the discontinuity of the line in the case of quite large regions. However in parallel with the growth of the missing region the onion-peel patch-match should do more iterations to be able to fill the whole region. This causes an error propagation starting from some iterations which leads to disconnections of the line in the generated part. However one still can notice that the algorithm tries to *prolong* the two sides of the line due to the patch-based nature of onion convolutions. While keeping growing the missing region, one can observe that the disconnection between the two ends of the line is also growing. The line continuity dependency on the mask area for this example can be found in the graph shown in Fig. 10.

A potential improvement of onion convolutions in this direction can be done by introducing some priorities for the order of filling the patches, similarly as suggested in Criminisi et al. (2004).

5 Conclusion

We present a *family of onion convolution modules* for image inpainting problem. This concept is a result of combining patch-based techniques with attention mechanisms. As a hybrid method, the family of onion convolutions inherits the *long-range pixel dependency capturing* ability from the attention mechanisms and the *feature continuity preserving* ability from the patch-based techniques. This allows modeling of high-level semantics as well as propagating textures and structures from the known region to the unknown. We show that our method outperforms existing state-of-the-art approaches of image inpainting, both quantitatively and qualitatively. It is worth noting that our onion convolutions can be adopted to various architectures and learning techniques.

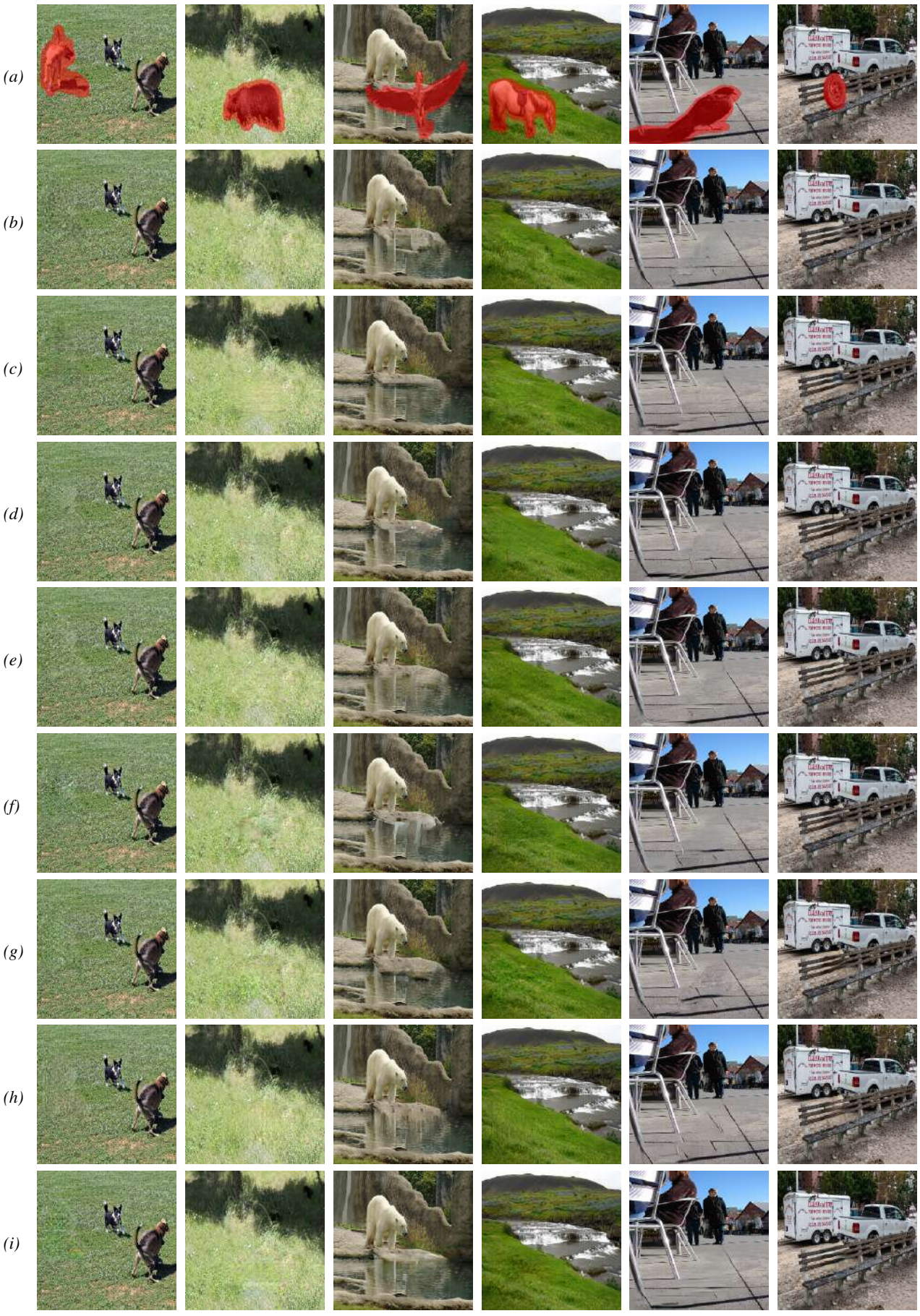


Figure 9: *(a)* Masked image, *(b)* Onion-EV-BestMatchSim, *(c)* Onion-EV-OppositeSim, *(d)* Onion-EV-InverseSim, *(e)* Onion-EV-CosSim, *(f)* Onion-RS-BestMatchSim (Navasardyan and Ohanyan, 2020), *(g)* Onion-RS-OppositeSim, *(h)* Onion-RS-InverseSim, *(i)* Onion-RS-CosSim .

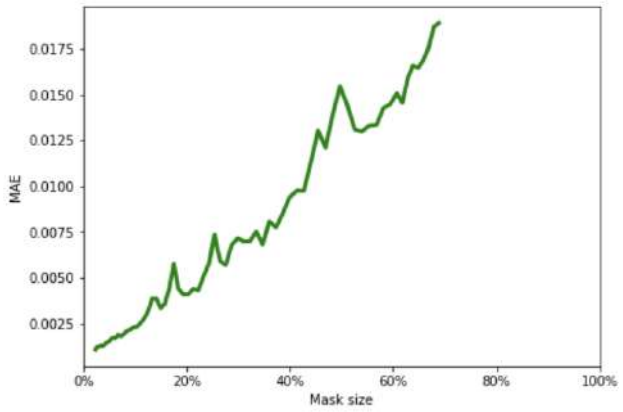


Figure 10: Dependency of the line continuity (in terms of MAE) on the mask area.

References

Abadi M, Agarwal A, Barham P, Brevdo E, Chen Z, Citro C, Corrado GS, 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 (2015) TensorFlow: Large-scale machine learning on heterogeneous systems. URL `https://www.tensorflow.org/`, software available from tensorflow.org

Ashikhmin M (2001) Synthesizing natural textures. In: Proceedings of the 2001 Symposium on Interactive 3D Graphics, Association for Computing Machinery, New York, NY, USA, I3D '01, p 217–226, DOI 10.1145/364338.364405, URL `https://doi.org/10.1145/364338.364405`

Bahdanau D, Cho K, Bengio Y (2015) Neural machine translation by jointly learning to align and translate. CoRR abs/1409.0473

Barnes C, Shechtman E, Finkelstein A, Goldman DB (2009) PatchMatch: A randomized correspondence algorithm for structural image editing. ACM Transactions on Graphics (Proc SIGGRAPH) 28(3)

Bello I, Zoph B, Le Q, Vaswani A, Shlens J (2019) Attention augmented convolutional networks. In: 2019 IEEE/CVF International Conference on Computer Vision (ICCV), pp 3285–3294

Bertalmío M, Sapiro G, Caselles V, Ballester C (2000) Image inpainting. In: SIGGRAPH '00

Bertalmio M, Vese L, Sapiro G, Osher S (2003) Simultaneous structure and texture image inpainting. IEEE Transactions on Image Processing 12(8):882–889

Boykov Y, Veksler O, Zabih R (2001) Fast approximate energy minimization via graph cuts. IEEE Transactions on Pattern Analysis and Machine Intelligence 23(11):1222–1239

Canny J (1986) A computational approach to edge detection. IEEE Trans Pattern Anal Mach Intell 8(6):679–698, DOI 10.1109/TPAMI.1986.4767851, URL `https://doi.org/10.1109/TPAMI.1986.4767851`

Chan TF, Shen J (2000) Non-texture inpainting by curvature-driven diffusions (cdd). J Visual Comm Image Rep 12:436–449

Chen LC, Zhu Y, Papandreou G, Schroff F, Adam H (2018) Encoder-decoder with atrous separable convolution for semantic image segmentation. In: Ferrari V, Hebert M, Sminchisescu C, Weiss Y (eds) ECCV (7), Springer, Lecture Notes in Computer Science, vol 11211, pp 833–851, URL `http://dblp.uni-trier.de/db/conf/eccv/eccv2018-7.html#ChenZPSA18`

Chen TQ, Schmidt M (2016) Fast patch-based style transfer of arbitrary style. arXiv preprint arXiv:161204337

Clevert DA, Unterthiner T, Hochreiter S (2015) Fast and accurate deep network learning by exponential linear units (elus). CoRR abs/1511.07289

Cordonnier JB, Loukas A, Jaggi M (2020) On the relationship between self-attention and convolutional layers. In: International Conference on Learning Representations, URL https://openreview.net/forum?id=HJlnC1rKPB

Criminisi A, Pérez P, Toyama K (2003) Object removal by exemplar-based inpainting. vol 2, pp 721–728, DOI 10.1109/CVPR.2003.1211538

Criminisi A, Perez P, Toyama K (2004) Region filling and object removal by exemplar-based image inpainting. Trans Img Proc 13(9):1200–1212, DOI 10.1109/TIP.2004.833105, URL https://doi.org/10.1109/TIP.2004.833105

Deng J, Dong W, Socher R, Li LJ, Li K, Fei-Fei L (2009) Imagenet: A large-scale hierarchical image database. In: 2009 IEEE conference on computer vision and pattern recognition, Ieee, pp 248–255

Efros A, Leung T (1999) Texture synthesis by non-parametric sampling. In: In International Conference on Computer Vision, pp 1033–1038

Efros AA, Freeman WT (2001) Image quilting for texture synthesis and transfer. In: Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques, Association for Computing Machinery, New York, NY, USA, SIGGRAPH '01, p 341–346, DOI 10.1145/383259.383296, URL https://doi.org/10.1145/383259.383296

Fu J, Liu J, Tian H, Li Y, Bao Y, Fang Z, Lu H (2019) Dual attention network for scene segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp 3146–3154

Garber DD (1981) Computational models for texture analysis and texture synthesis. PhD thesis, USA, aAI0551115

Gatys LA, Ecker AS, Bethge M (2016) Image style transfer using convolutional neural networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 2414–2423

Goodfellow IJ, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A, Bengio Y (2014) Generative adversarial nets. In: Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2, MIT Press, Cambridge, MA, USA, NIPS'14, p 2672–2680

Gregor K, Danihelka I, Graves A, Rezende D, Wierstra D (2015) Draw: A recurrent neural network for image generation. In: Bach F, Blei D (eds) Proceedings of the 32nd International Conference on Machine Learning, PMLR, Lille, France, Proceedings of Machine Learning Research, vol 37, pp 1462–1471, URL http://proceedings.mlr.press/v37/gregor15.html

Harrison P (2001) A non-hierarchical procedure for re-synthesis of complex textures. In: Skala V (ed) Proceedings of The 9-th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision'2001, University of West Bohemia, pp 190 – 197, URL http://wscg.zcu.cz/wscg2001/wscg2001.htm, international Conferences in Central Europe on Computer Graphics, Visualization and Computer Vision 2001, WSCG 2001 ; Conference date: 05-02-2001 Through 09-02-2001

Hertzmann A, Jacobs CE, Oliver N, Curless B, Salesin DH (2001) Image analogies. In: Proceedings of the 28th annual conference on Computer graphics and interactive techniques, pp 327–340

Heusel M, Ramsauer H, Unterthiner T, Nessler B, Hochreiter S (2017) Gans trained by a two time-scale update rule converge to a local nash equilibrium. Advances in neural information processing systems 30

Hu J, Shen L, Sun G (2018) Squeeze-and-excitation networks. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp 7132–7141

Huang JB, Kang SB, Ahuja N, Kopf J (2014) Image completion using planar structure guidance. ACM Trans Graph 33(4), DOI 10.1145/2601097.2601205, URL https://doi.org/10.1145/2601097.2601205

Hung J, Chun-Hong H, Yi-Chun L, Tang N, Ta-Jen C (2008) Exemplar-based image inpainting base on structure construction. Journal of Software 3, DOI 10.4304/jsw.3.8.57-64

Iizuka S, Simo-Serra E, Ishikawa H (2017) Globally and locally consistent image completion. ACM Transactions on Graphics 36:1–14, DOI 10.1145/3072959.3073659

Karras T, Aila T, Laine S, Lehtinen J (2017) Progressive growing of gans for improved quality, stability, and variation. arXiv preprint arXiv:171010196

Kingma DP, Ba J (2015) Adam: A method for stochastic optimization. CoRR abs/1412.6980

Kingma DP, Welling M (2013) Auto-encoding variational bayes. arXiv preprint arXiv:13126114

Köhler R, Schuler C, Schölkopf B, Harmeling S (2014) Mask-specific inpainting with deep neural networks. In: German Conference on Pattern Recognition, Springer International Publishing, Cham, pp 523–534

Krizhevsky A, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems 25:1097–1105

Kwatra V, Schödl A, Essa I, Turk G, Bobick A (2003) Graphcut textures: Image and video synthesis using graph cuts. ACM Trans Graph 22:277–286, DOI 10.1145/1201775.882264

Levin, Zomet, Weiss (2003) Learning how to inpaint from global image statistics. In: Proceedings Ninth IEEE International Conference on Computer Vision, pp 305–312 vol.1

Li C, Wand M (2016) Combining markov random fields and convolutional neural networks for image synthesis. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp 2479–2486

Li J, Wang N, Zhang L, Du B, Tao D (2020a) Recurrent feature reasoning for image inpainting. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)

Li Y, Lu H (2020) Natural image matting via guided contextual attention. arXiv preprint arXiv:200104069

Li Y, Xu Q, Lu H (2020b) Hierarchical opacity propagation for image matting. arXiv preprint arXiv:200403249

Liao J, Yao Y, Yuan L, Hua G, Kang SB (2017) Visual attribute transfer through deep image analogy. ACM Trans Graph 36(4), DOI 10.1145/3072959.3073683, URL https://doi.org/10.1145/3072959.3073683

Liu G, Reda FA, Shih KJ, Wang TC, Tao A, Catanzaro B (2018) Image inpainting for irregular holes using partial convolutions. In: ECCV

Luong T, Pham H, Manning CD (2015) Effective approaches to attention-based neural machine translation. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Lisbon, Portugal, pp 1412–1421, DOI 10.18653/v1/D15-1166, URL https://www.aclweb.org/anthology/D15-1166

Miyato T, Kataoka T, Koyama M, Yoshida Y (2018) Spectral normalization for generative adversarial networks. In: International Conference on Learning Representations, URL https://openreview.net/forum?id=B1QRgziT-

Navasardyan S, Ohanyan M (2020) Image inpainting with onion convolutions. In: Proceedings of the Asian Conference on Computer Vision

Nazeri K, Ng E, Joseph T, Qureshi F, Ebrahimi M (2019) Edgeconnect: Structure guided image inpainting using edge prediction. In: The IEEE International Conference on Computer Vision (ICCV) Workshops

Oh SW, Lee S, Lee JY, Kim SJ (2019) Onion-peel networks for deep video completion. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp 4403–4412

Park DY, Lee KH (2019) Arbitrary style transfer with style-attentional networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp 5880–5888

Pathak D, Krähenbühl P, Donahue J, Darrell T, Efros AA (2016) Context encoders: Feature learning by inpainting. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp 2536–2544

Ramachandran P, Parmar N, Vaswani A, Bello I, Levskaya A, Shlens J (2019) Stand-alone self-attention in vision models. In: NeurIPS

Rane SD, Sapiro G, Bertalmio M (2003) Structure and texture filling-in of missing image blocks in wireless transmission and compression applications. Trans Img Proc 12(3):296–303, DOI 10.1109/TIP.2002.804264, URL https://doi.org/10.1109/TIP.2002.804264

Ren JSJ, Xu L, Yan Q, Sun W (2015) Shepard convolutional neural networks. In: NIPS

Shen J, Chan T (2002) Mathematical models for local nontexture inpaintings. SIAM Journal on Applied Mathematics 62:1019–1043, DOI 10.1137/S0036139900368844

Simonyan K, Zisserman A (2015) Very deep convolutional networks for large-scale image recognition. In: Bengio Y, LeCun Y (eds) 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings, URL http://arxiv.org/abs/1409.1556

Song Y, Yang C, Lin Z, Liu X, Huang Q, Li H, Jay Kuo CC (2018a) Contextual-based image inpainting: Infer, match, and translate. In: Proceedings of the European Conference on Computer Vision (ECCV), pp 3–19

Song Y, Yang C, Shen Y, Wang P, Huang Q, Kuo CJ (2018b) Spg-net: Segmentation prediction and guidance network for image inpainting. CoRR abs/1805.03356, URL http://arxiv.org/abs/1805.03356, 1805.03356

Suin M, Purohit K, Rajagopalan A (2020) Spatially-attentive patch-hierarchical network for adaptive motion deblurring. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp 3606–3615

Sun J, Yuan L, Jia J, Shum HY (2005) Image completion with structure propagation. ACM Trans Graph 24(3):861–868, DOI 10.1145/1073204.1073274, URL https://doi.org/10.1145/1073204.1073274

Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser Lu, Polosukhin I (2017) Attention is all you need. In: Advances in Neural Information Processing Systems 30, Curran Associates, Inc., pp 5998–6008, URL http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf

Wei LY, Levoy M (2000) Fast texture synthesis using tree-structured vector quantization. In: Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques, ACM Press/Addison-Wesley Publishing Co., USA, SIGGRAPH '00, p 479–488, DOI 10.1145/344779.345009, URL https://doi.org/10.1145/344779.345009

Williams RJ (1992) Simple statistical gradient-following algorithms for connectionist reinforcement learning. Mach Learn 8(3–4):229–256, DOI 10.1007/BF00992696, URL https://doi.org/10.1007/BF00992696

Xie C, Liu S, Li C, Cheng M, Zuo W, Liu X, Wen S, Ding E (2019) Image inpainting with learnable bidirectional attention maps. In: 2019 IEEE/CVF International Conference on Computer Vision (ICCV), pp 8857–8866

Xie J, Xu L, Chen E (2012) Image denoising and inpainting with deep neural networks. In: Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1, Curran Associates Inc., Red Hook, NY, USA, NIPS'12, p 341–349

Xiong W, Yu J, Lin Z, Yang J, Lu X, Barnes C, Luo J (2019) Foreground-aware image inpainting. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)

Xu K, Ba J, Kiros R, Cho K, Courville A, Salakhudinov R, Zemel R, Bengio Y (2015) Show, attend and tell: Neural image caption generation with visual attention. In: International conference on machine learning, pp 2048–2057

Xu Z, Sun J (2010) Image inpainting by patch propagation using patch sparsity. IEEE transactions on image processing 19(5):1153–1165

Yan Z, Li X, Li M, Zuo W, Shan S (2018) Shift-net: Image inpainting via deep feature rearrangement. In: The European Conference on Computer Vision (ECCV)

Yang C, Lu X, Lin Z, Shechtman E, Wang O, Li H (2016) High-resolution image inpainting using multi-scale neural patch synthesis. arXiv preprint arXiv:161109969v2

Yao Y, Ren J, Xie X, Liu W, Liu YJ, Wang J (2019) Attention-aware multi-stroke style transfer. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp 1467–1475

Yeh RA, Chen C, Lim TY, Schwing AG, Hasegawa-Johnson M, Do MN (2017) Semantic image inpainting with deep generative models. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp 6882–6890

Yi Z, Tang Q, Azizi S, Jang D, Xu Z (2020) Contextual residual aggregation for ultra high-resolution image inpainting. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp 7508–7517

Yu J, Lin Z, Yang J, Shen X, Lu X, Huang TS (2018) Generative image inpainting with contextual attention. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp 5505–5514

Yu J, Lin Z, Yang J, Shen X, Lu X, Huang T (2019) Free-form image inpainting with gated convolution. In: 2019 IEEE/CVF International Conference on Computer Vision (ICCV), pp 4470–4479

Zamir SW, Arora A, Khan S, Hayat M, Khan FS, Yang MH, Shao L (2021) Multi-stage progressive image restoration. In: CVPR

Zhang H, Goodfellow I, Metaxas D, Odena A (2019) Self-attention generative adversarial networks. In: Proceedings of the 36th International Conference on Machine Learning, PMLR, Long Beach, California, USA, Proceedings of Machine Learning Research, vol 97, pp 7354–7363, URL http://proceedings.mlr.press/v97/zhang19d.html

Zhang R, Isola P, Efros AA, Shechtman E, Wang O (2018) The unreasonable effectiveness of deep features as a perceptual metric. In: CVPR

Zheng C, Cham TJ, Cai J (2019) Pluralistic image completion. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) pp 1438–1447

Zhou B, Lapedriza A, Khosla A, Oliva A, Torralba A (2017) Places: A 10 million image database for scene recognition. IEEE transactions on pattern analysis and machine intelligence 40(6):1452–1464

Zhou Wang, Bovik AC, Sheikh HR, Simoncelli EP (2004) Image quality assessment: from error visibility to structural similarity. IEEE Transactions on Image Processing 13(4):600–612