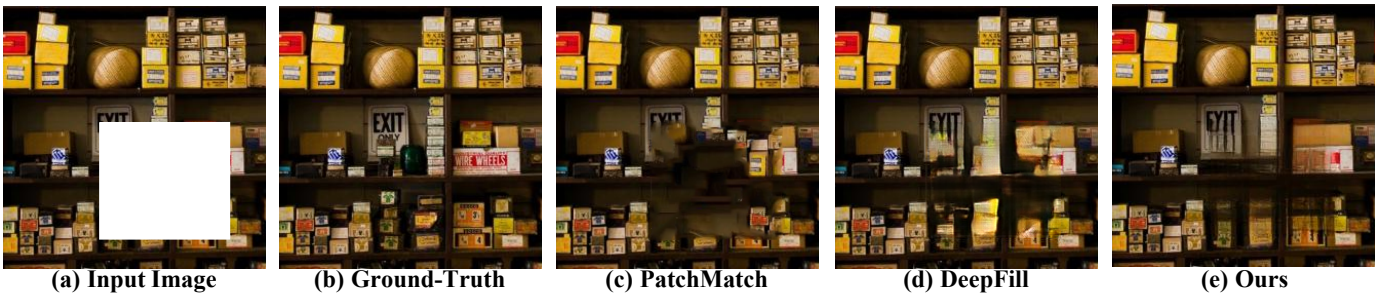
Texture Memory-Augmented Deep Patch-Based Image Inpainting

Rui Xu, Minghao Guo, Jiaqi Wang, Xiaoxiao Li, Bolei Zhou and Chen Change Loy

arXiv:2009.13240v1 [cs.CV] 28 Sep 2020



(a) Input Image (b) Ground-Truth (c) PatchMatch (d) DeepFill (e) Ours

Fig. 1: Comparing inpainting results generated by (c) a patch-based approach, (d) a deep learning-based method, and (e) the proposed Texture Memory Augmented approach. Our method is capable of recovering both the global semantic structure and local texture details, taking the best of both worlds of patch-based paradigm and deep learning-based method.

***Abstract*—Patch-based methods and deep networks have been employed to tackle image inpainting problem, with their own strengths and weaknesses. Patch-based methods are capable of restoring a missing region with high-quality texture through searching nearest neighbor patches from the unmasked regions. However, these methods bring problematic contents when recovering large missing regions. Deep networks, on the other hand, show promising results in completing large regions. Nonetheless, the results often lack faithful and sharp details that resemble the surrounding area. By bringing together the best of both paradigms, we propose a new deep inpainting framework where texture generation is guided by a texture memory of patch samples extracted from unmasked regions. The framework has a novel design that allows texture memory retrieval to be trained end-to-end with the deep inpainting network. In addition, we introduce a patch distribution loss to encourage high-quality patch synthesis. The proposed method shows superior performance both qualitatively and quantitatively on three challenging image benchmarks, i.e., Places, CelebA-HQ, and Paris Street-View datasets.**[1]

***Index Terms*—Image Completion, Generative Adversarial Network, Texture Synthesis.**

I. Introduction

Image inpainting aims at filling in missing regions of a given image with consistent and coherent contents. The challenge of image inpainting lies in two aspects, *i.e.*, reconstructing the missing global structure and synthesizing realistic local textures coherent to unmasked regions.

Image inpainting is a long-standing problem in the field of computer vision. Classic patch-based methods [1], [2] perform patch matching within an image and fill the missing region that plausibly matches the remaining image content. While textures are faithfully recovered in many cases, patch-based methods often fail to restore the global geometric structure, especially when presented with corrupted images with large missing regions. A failure case from the popular PatchMatch method [1] is shown in Fig. 1(c). A significant progress in image inpainting has been attained through the usage of deep convolutional neural networks (CNN) and generative adversarial network (GAN) [3], [4], [5]. Deep inpainting methods formulate image inpainting as a conditional image generation task. A convolutional encoder-decoder is typically trained with an adversarial loss to restore reasonable structures and textures. These methods demonstrate a promising performance in hallucinating the missing contents with coherent semantic structures. However, given the high complexity of scenes and textures in the wild, it is still challenging for a deep generative model to inpaint arbitrary images.

The goal of this work is to bring together the best of classic patch-based and deep learning based approaches, *i.e.*, leveraging deep networks to recover the global structure of the missing region and exploiting the notion of patch matching to restore the details. It is non-trivial to combine the two vastly different paradigms in a unified end-to-end model. Specifically, the learning process of classic patch-based methods is not end-to-end, requiring expensive patch-synthesis based on optimization *w.r.t.* the content and texture constraints. A plausible solution is to perform implicit patch selection through contextual attention, which has been widely used in deep inpainting literature [4], [5], [6], [7] to extract information from unmasked regions. Contextual attention module adopts a softmax function to fuse all candidate texture features for generation. In our experiments, we observe that interpolation in texture space may lead to blurry output and unnatural artifacts, as shown in Fig. 1(d).

To mitigate the aforementioned challenges, we formulate a new mechanism for patch matching, retrieval, and generation within the deep network framework. The proposed method, which we call as *Texture Memory-Augmented Deep Patch-Based*

[1]Code will be made publicly available in https://github.com/open-mmlab/mmediting.

Image Inpainting (T-MAD) has a few appealing properties. Differing to implicit patch selection in contextual attention which involves interpolation in texture space, we present a new sampling strategy that provides a more explicit guidance towards the generation of patch-level details. This is achieved through the notion of *texture memory*, which contains patches extracted from unmasked regions of the input image. With a tailored design, the patch matching and retrieval steps in texture memory is end-to-end trainable. Guided by the coarse result (which can be generated by any existing deep inpainting networks), patches with high similarity to the coarse result are retrieved from the memory, and their features are then used to guide the recovery of patch-level textures on top of the coarse global reconstruction. With this retrieve-and-guidance design, we pay more attention to the generation of patch-level details, the results are thus less susceptible to blurry and unnatural artifacts. An example is shown in Fig. 1(e), more examples can be found in the experiments section.

The main contribution of this paper is an effective image inpaiting method that unifies patch- and deep learning-based approaches. We devise an effective method that enables back-propagation for patch matching and retrieval from texture memory. Hence, these steps can be learned in an end-to-end manner within our unified framework. We also propose an effective adversarial loss at patch level to better capture patch statistics. Different from existing adversarial losses for learning the distribution of the whole dataset, we adopt patch distribution loss to model the texture distribution of a target image.

We show the advantages of retrieve-and-guidance framework on various benchmark data with random rectangle masks and free-form irregular holes. In particular, our method outperforms state-of-the-art methods CRA [7] in two different mask settings quantitatively and qualitatively. We further show that the notion of texture memory can be easily adapted into other pipelines, *e.g.*, DeepFill [4], as a useful post-processing module to improve the quality of image inpainting.

II. Related Work

Patch-Based Image Inpainting. The seminal work PatchMatch [1] and its successful variants [2], [8], [9] have been widely used in image editing and inpainting. These optimization-based methods regard image inpainting as finding the optimal patches for each masked location with manually designed constraints. These methods have shown to be robust to the variance of textures and input resolution. However, the optimization process is expensive. In addition, these methods are inherently limited by the ability of generating novel contents and they tend to fail to preserve a reasonable global structure when the missing region is large. Yang *et al* [8] adopt a deep model to reconstruct the coarse content and apply VGG network [10] to compute a perceptual loss as an extra constraint in the patch matching paradigm. Nevertheless, their method requires expensive multi-scale patch synthesis based on optimization *w.r.t.* the content and texture constraints. In addition, the learning of coarse generation and texture synthesis cannot be trained end-to-end.

Deep Image Inpainting. Existing deep inpainting methods fall into two categories, single-stage and two-stage approaches. Single-stage approaches [11], [3], [12], [13] adopt an encoder-decoder network with multiple losses to recover the corrupted region directly. The networks are trained to jointly capture the structure and texture information in a single pass. As for two-stage approaches, existing models can be further divided into two streams. Methods from the first category [4], [5] adopt a coarse-to-fine framework where synthesis is gradually refined. Approaches in the second category reconstruct structural information in the first stage as a prior to guide the second stage for synthesizing detailed textures [14], [15], [6].

Although the aforementioned methods are capable of generating better global structures compared with the traditional optimization-based methods, the generated textures are still unsatisfactory for two reasons. First, they require the model to synthesize detailed textures on the entire image. The requirement inevitably adds difficulty in training a high-resolution GAN. Second, these methods typically encounter difficulty in generating complex textures for in-the-wild images since such textures of the target image may not be seen during training. To address these problems, we synthesize textures at patch level and leverage the texture memory constructed from unmasked regions.

The fact that our method retrieves plausible patches from a texture memory can be loosely regarded as a kind of attention. Attention mechanism [16], [17], [18] has been widely used in deep inpainting literature [4], [5], [6], [7] to extract information from valid regions. The key differences between our method and the popular contextual attention approach lie in the ultimate generative goal and patch sampling strategy. First, T-MAD aims at synthesizing high-quality patches with guidance of texture memory to inpaint the hole while contextual attention approaches [4], [5], [6], [7] directly inpaint at image level. Second, while contextual attention module adopts softmax function to fuse candidate texture features for generation, the proposed T-MAD uses a differentiable patch retrieval module to sample similar patches to guide the generation process and thus avoids interpolation in texture space. The differences lead to the better capability in T-MAD in generating visually pleasing details.

III. Methodology

Unlike previous deep models that perform inpainting on an entire image, our T-MAD approach focuses on the corrupted regions to synthesize high-quality patches. These patches are then tiled into the missing region accordingly. As illustrated in Fig. 2, our T-MAD contains three modules. First, as shown in Fig. 2(a), we roughly estimate the missing contents with a coarse network trained with l_1 reconstruction loss, producing a reasonable global structural prior. Next, we split the coarse result into non-overlapping patches of equal size, $\{p_s\}$, and further prepare $\tilde{p}_s$ that are extracted from the coarse result according to $\{p_s\}$ with a larger neighboring context. Meanwhile, as shown in Fig. 2(b), we maintain a texture memory containing multiple patches extracted from the unmasked region. Guided by the coarse result patches, the retrieval module selects the most similar texture patches as a prior for subsequent patch synthesis step in patch synthesis module (PSNet), as depicted in

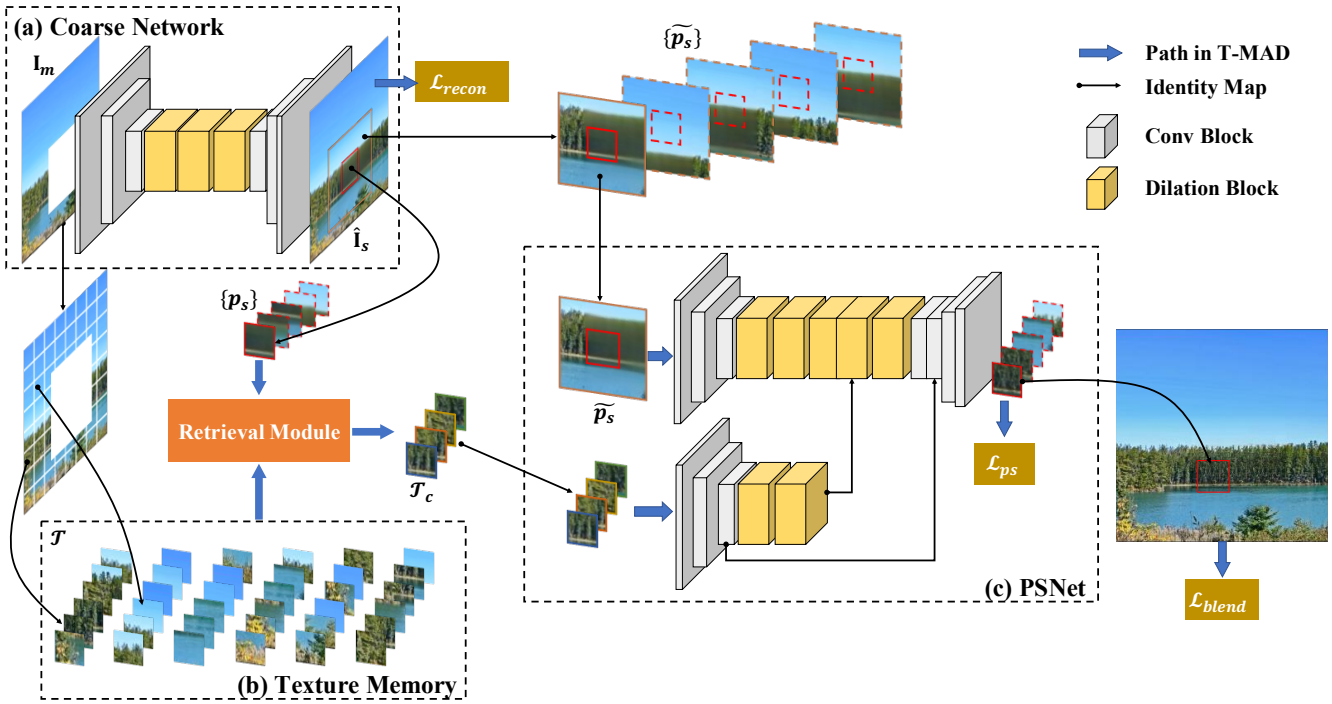


Fig. 2: Framework of the proposed T-MAD. There are three key modules: (a) A coarse network for generating the coarse global results, (b) A texture memory for maintaining texture priors from unmasked regions, and (c) A patch synthesis module (PSNet) for patch synthesis, conditioned on the coarse global results and patches retrieved from the texture memory. Patch retrieval from the texture memory is formulated as an end-to-end trainable module.

Fig. 2(c). By fusing information from both the coarse structure and the patches with native texture, the proposed patch synthesis module generates high-quality textures that are coherent with the unmasked region and meet the global structure constraint.

The section is organized as follows: Sec. III-A introduces the proposed coarse network for recovering rough structure information. The construction and retrieval procedure of the texture memory are clarified in Sec. III-B. Finally, the patch synthesis module is explained in Sec. III-C

A. Coarse Network for Structural Prior

A coarse network, as depicted in Fig. 2(a), is used to generate a coarse and global estimation, of which the content will be used to guide the subsequent patch synthesis process. One can adopt any existing deep inpainting network for this purpose. In this work, the network takes an encoder-decoder structure, accepting a masked image $\mathbf{I}_m$ and the corresponding binary indicating mask $\mathbf{M}$ as input. The coarse network produces an initial coarse result $\mathbf{I}_s$ with the same size as the input. The completed image $\hat{\mathbf{I}}_s$ can be obtained by combining $\mathbf{I}_s$ with valid pixels from the unmasked region.

A large receptive field is desirable for recovering missing structural information. Adopting dilated residual blocks [19], [14] is a feasible option to enlarge the receptive field. We opt to adopt eight residual blocks in our coarse network, after taking into account both the model capacity and the training cost. Meanwhile, we substitute all up-sampling modules with CARAFE [20] to allow content-aware feature reassembly for better structural reconstruction. The coarse network is supervised with l_1 reconstruction loss between the output and the original image $\mathbf{I}$

$$\mathcal{L}_{\text{recon}} = \lambda_m^s \left\|\mathbf{I}_s - \mathbf{I}\right\|_1 \odot \mathbf{M} + \lambda_v^s \left\|\mathbf{I}_s - \mathbf{I}\right\|_1 \odot (1 - \mathbf{M}), \quad (1)$$

where $\odot$ denotes an element-wise product. We empirically find that setting $\lambda_m^s = 5$ and $\lambda_v^s = 0$ brings better results for random rectangle missing regions, while $\lambda_m^s = 6$, and $\lambda_v^s = 1$ is a better choice for irregular holes. After obtaining $\hat{\mathbf{I}}_s$, we apply a $k_p \times k_p$ sliding window to cut the completed region in $\hat{\mathbf{I}}_s$ into multiple non-overlapping patches $\{p_s\}$. The patches $\{p_s\}$ that encapsulate the initial structure prior will guide the subsequent texture retrieval and patch synthesis process.

B. Retrieving Visual Prior from Texture Memory

Texture Memory. Different from existing deep learning approaches, T-MAD has a carefully designed texture memory containing texture patches extracted from the unmasked region. Together with the coarse global results, these patches will be used to guide the patch synthesis process.

Similar to the procedure in extracting coarse result patches, as shown in Fig. 2(b), a $k_p \times k_p$ sliding window is adopted to extract valid texture patches from $\mathbf{I}_m$. We perform a dense sampling to keep sufficient texture patches in our memory pool. A reasonable choice of sliding window stride is $s_{pool} = k_p/2$. Given the texture patches, we randomly select $\mathbb{N}_{pool}$ patches to construct the final texture pool $p_t \in \mathcal{T}$. As for irregular holes like [4], s_{pool} is set to $k_p/4$ and we choose texture patches constaining the least masked regions as p_t for further increasing the number of valid patches. In our implementation,

$\mathbb{N}_{pool}$ is set to 100. For the patch size in the texture memory, we recommend a size of $k_p = 32$. This setting gives a good balance between the final texture quality and the training cost. It is noteworthy that unlike previous methods [21] in computer graphics, we do not need to introduce complex distortions to the texture patches since these patches mainly serve as a condition for the following patch synthesis module (discussed in Section III-C). The module is capable of transferring the texture effectively onto the inpainted image.

Patch Retrieval. The selected patches in the texture memory are retrieved to guide the process of patch synthesis. The retrieval module consists of two components to perform *correspondence embedding* and *sampling*, respectively.

1) *Correspondence embedding* – The correspondence embedding component constructs a similarity matrix c between the coarse patches p_s^i and the texture patches in the texture memory $p_t \in \mathcal{T}$ using a non-local module [17]

$$c(p_s^i, p_t^j) = \theta(p_s^i)^\top \phi(p_t^j), \tag{2}$$

where θ and ϕ are two shallow convolutional networks. As the scale of $||\phi(x_j)||$ introduces an unreasonable norm term to $c(p_s^i, p_t^j)$, we reformulate the original equation as:

$$c(p_s^i, p_t^j) = \theta(p_s^i)^\top \frac{\phi(p_t^j)}{||\phi(p_t^j)||}. \tag{3}$$

The normalization of $\phi(p_t^j)$ removes the influence of the norm value on learning such specific similarity. The output of the non-local module is normalized by a softmax function to establish the correspondence $\mathcal{S}$ between every patch in $\{p_s\}$ and in $\mathcal{T}$.

2) *Sampling* – Previous works with memory pool [22], [23] predict softmax value to form the pool in a weighted-sum manner. However, an interpolation of different textures prevents one from generating meaningful textures as the weighted sum would result in blurry outputs. In our approach, for each patch p_s, we sample $\mathbb{N}_c$ texture patches $p_c \in \mathcal{T}_c$ from $\mathcal{T}$ as candidates. These patches are regarded as those that are most similar to the coarse patch p_s, yet contain diverse and rich priors on textures. In Fig. 2, we use an example patch with solid red bounding box to show the retrieval procedure.

Differentiable Sampling. It is known that the sampling operation prevents gradients from having a direct path in back-propagation. Inspired by Binary Network [24], we formulate a discrete weighted sum to ensure that the output in the forward path is the exact patch we need. Meanwhile, a soft value will be adopted in the back-propagation. Suppose the obtained similarity vector is $\mathcal{S} \in \mathbb{R}^{1\times\mathbb{N}_{pool}}$. To sample the i-th patch from $\mathcal{T}$, we first construct an indicating vector $\mathbf{1}_i \in \mathbb{R}^{1\times\mathbb{N}_{pool}}$, in which only the i-th index is 1 and the others are 0. We can obtain the exact i-th patch p_i^s from $\mathcal{T}$ through Eq. (4), where $\otimes$ denotes a weighted-sum operation and `detach` indicates variables that do not require gradient in the back-propagation process.

$$p_c^i = \mathbf{1} \otimes \mathcal{T} = \{(\mathbf{1} - \mathcal{S}).\texttt{detach} + \mathcal{S}\} \otimes \mathcal{T}. \tag{4}$$

The back-propagation through our sampling function is as

$$\begin{aligned} \frac{\partial p_c^i}{\partial w} &= \frac{\partial \mathbf{1}}{\partial w} \otimes \mathcal{T} \\ &= (\frac{\partial(\mathbf{1} - \mathcal{S}).\texttt{detach}}{\partial w} + \frac{\partial \mathcal{S}}{\partial w}) \otimes \mathcal{T} \\ &= (\mathbf{0} + \frac{\partial \mathcal{S}}{\partial w}) \otimes \mathcal{T} = \frac{\partial \mathcal{S}}{\partial w} \otimes \mathcal{T}, \end{aligned} \tag{5}$$

where w indicates learnable parameters in this module. Due to the `detach` operation, the back-propagation procedure regards $\mathbf{1} - \mathcal{S}$ as a constant value so that gradients can be propagated to $\mathcal{S}$ successfully. This reformulation allows the proposed T-MAD to be trained in an end-to-end manner.

Gumbel-Softmax [25] replaces non-differentiable sampling from a categorical distribution with a differentiable sampling from a novel Gumbel-Softmax distribution. However, Gumbel-Softmax can easily sample the one with highest score but cannot sample other candidates with specific score or rank.

C. Patch Synthesis Module – PSNet

In this module, we focus on generating high-quality patches guided by coarse result patch $\{p_s\}$ and the selected texture prior $\mathcal{T}_c$. The goal is not only to synthesize textures without blurry artifacts but also to match the texture distribution of the unmasked region. Importantly, $\{p_s\}$ captures global structural prior, which guides the texture synthesis module to have a reasonable global semantic structure. Through synthesizing textures on each small patch, we expect to lift the burden of training a general conditional GAN on the entire image.

Network Architecture. As shown in Fig. 2(c), an encoder-decoder network is adopted as a backbone for texture synthesis, while another simple encoder is designed to encode the texture prior $\mathcal{T}_c$ for PSNet. Two skip connections provide the backbone encoder-decoder network with pyramid texture information from the texture memory.

The inputs to the backbone encoder-decoder and texture prior encoder are different. To preserve the consistency between the synthesized texture and its neighboring patches, the backbone encoder-decoder takes the initial coarse patches $\{p_s\}$ and its neighboring patches in $\hat{I}_s$ as inputs. This can be achieved by using a larger sliding window $3k_p \times 3k_p$ to extract larger patches $\tilde{p}_s$ with stride $s_{\tilde{p}_s} = k_p$. As for the input for the texture encoder network, all patches in $\mathcal{T}_c$ will be concatenated as the input.

A notable feature of our method is that it can perform patch synthesis in parallel. For parallel computation, all $\tilde{p}_s^i$ and the corresponding $p_c^i \in \mathcal{T}_c$ are reshaped into a single batch dimension. Compared with existing deep image inpainting models [11], [4] that need to process an entire image, our approach consumes 20% less memory in training and inference by processing non-overlapped patches of missing regions in parallel.

Patch Distribution Loss. To leverage the texture prior and match the texture distribution across observed patches, we propose a patch-level adversarial loss.

In our task, besides the ground-truth patches p_{gt}, the synthesized patches should also subject to a similar distribution with candidate texture pool $\mathcal{T}_c$. Following the notion of adding

relativistic comparison between fake results and real samples in RAGAN [26], we introduce the comparison at patch level into adversarial training. The detailed formulation is shown in Eq. (6) and Eq. (7)

$$\begin{aligned} L_D = &\mathbb{E}_{x_r \sim \mathbb{P}(\mathcal{T}_c, p_{gt})}[f_1(C(x_r) - \mathbb{E}_{x_f \sim Q} C(x_f))] \\ &+ \mathbb{E}_{x_f \sim Q}[f_2(C(x_f) - \mathbb{E}_{x_r \sim \mathbb{P}(\mathcal{T}_c, p_{gt})} C(x_r))], \end{aligned} \tag{6}$$

$$\begin{aligned} L_G = &\mathbb{E}_{x_r \sim \mathbb{P}(\mathcal{T})}[g_1(C(x_r) - \mathbb{E}_{x_f \sim Q} C(x_f))] \\ &+ \mathbb{E}_{x_f \sim Q}[g_2(C(x_f) - \mathbb{E}_{x_r \sim \mathbb{P}(\mathcal{T}_c, p_{gt})} C(x_r))], \end{aligned} \tag{7}$$

where f_1, f_2, g_1, g_2 are scalar-to-scalar functions, $\mathbb{P}(\cdot)$ indicates the distribution of certain data, $C(x)$ is the non-transformed discriminator output and $\mathcal{T}$ is our texture memory containing diverse texture patches from unmasked regions. Here, we provide the discriminator D with $(\mathcal{T}_c, p_{gt})$ to encourage the discriminator to predict the difference between x_f and the most similar textures. Meanwhile, in the generator G, a compensation term is added by computing the distance between $\mathbb{P}(\mathcal{T})$ and the mean distribution of $\{x_f\}$ in case of failing to find the suitable patches $\{p_c\}$. In addition, this compensation term encourages the mean style of the synthesized patches to be coherent with unmasked regions.

To encourage the patch synthesis module to generate patches following the constraints encoded in the coarse patches $\{p_s\}$, we also adopt a l_1 loss $\mathcal{L}_{l_1}$ as another supervision. Moreover, a perceptual loss $\mathcal{L}_{percep}$ is also adopted to further improve the perceptual quality. Unlike previous works [3], [14] that employ features after pooling layers, we adopt features before ReLU function to reduce artifacts. We use features from 'conv_5', 'conv_9' and 'conv_15' before ReLU in our perceptual loss. The total loss for patch synthesis is

$$\mathcal{L}_{\text{ps}} = \lambda^{pd}_{\text{gan}} \mathcal{L}^{pd}_{\text{gan}} + \lambda_{l_1} \mathcal{L}_{l_1} + \lambda_{\text{percep}} \mathcal{L}_{\text{percep}}. \tag{8}$$

The proposed patch-level adversarial loss is different from PatchGAN [27], [28] or patch discriminator [29], [30], which is widely adopted in recent deep inpainting methods [14], [4], [29]. These approaches encourage the discriminator to determine the quality of a synthesized image by classifying if each $N \times N$ patch in the image is real or fake. The scores of all patches are averaged to get the final score of the image as the loss. The main differences between PatchGAN and our approach are: 1) we model the distribution at patch level rather than image level, and 2) the goal of our patch synthesis module is to match the distribution of unmasked regions while PatchGAN still focuses on synthesizing an image *w.r.t.* the entire dataset distribution.

Blending Loss. To generate the final image $\hat{\mathbf{I}}_p$, the synthesized patches will be allocated back to the original position of the input to fill in the missing region. To remove the boundary artifacts and preserve the consistency among neighboring patches, we apply a total variation loss. A global patch discriminator [27] on $\hat{\mathbf{I}}_p$ is also helpful in this step. The blending loss is as

$$\mathcal{L}_{\text{blend}} = \lambda^{gl}_{\text{gan}} \mathcal{L}^{gl}_{\text{gan}} + \lambda_{\text{tv}} \mathcal{L}_{\text{tv}}. \tag{9}$$

The complete loss function of T-MAD contains the three terms provided in Eq. (1), Eq. (8) and Eq. (9), corresponding to the loss of coarse network, patch synthesis and the final tiling process

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{recon}} + \mathcal{L}_{\text{ps}} + \mathcal{L}_{\text{blend}}. \tag{10}$$

TABLE I: The number of basic blocks in different modules. 'input-conv' and 'out-conv' contain a 1×1 convolution layer for channel transformation. 'PSNet-FeatEnc' denotes the encoder in PSNet for extracting features from $\mathcal{T}_c$. The dilated residual block is denoted as 'res-block'.

	Coarse Net	PSNet-Backbone	PSNet-FeatEnc
input-conv	1	1	1
down-block	3	2	2
res-block	8	5	2
up-block	3	3	0
out-conv	1	1	0

TABLE II: Hyper-parameters in our T-MAD framework for random rectangle holes and free-form holes.

		Hyper-Parameters
$\mathcal{L}_{\text{recon}}$	λ_{valid}	1.
	λ_{hole}	6.
$\mathcal{L}_{\text{ps}}$	λ_{l_1}	1.
	$\lambda^{pd}_{\text{gan}}$	0.05
	λ_{percep}	0.02
$\mathcal{L}_{\text{blend}}$	λ_{tv}	0.02
	$\lambda^{blend}_{\text{gan}}$	0.02

IV. Experiments

Implementation Details. We follow the design of convolution blocks in PGGAN [31] to prepare our encoders and decoders as it proves to be effective on image generation. For all encoders in our T-MAD, a general downsampling block with 'conv3×3, conv3×3, Downsample' is adopted and we halve the image resolution using nearest neighbor filtering. As for decoders in our framework, we apply 'Upsample, conv3×3, conv3×3' as the basic building unit, named as upsampling block. The content-aware reassembly method, CARAFE [20], is chosen as our 'Upsample' method. In Tab. I, we show the detailed number of the basic blocks in different modules. Note that we just apply a standard residual block [14] in PSNet without any dilation operation. This is because PSNet aims at synthesizing high-quality textures at patch level and large receptive field is not necessary. In addition, we use leaky ReLU with leakiness 0.2 in all layers of the whole networks, except for the last layer in decoder that uses linear activation. We do not apply batch normalization or instance normalization in our T-MAD. In patch retrieval module, only two convolution layers with 'conv3×3, conv1×1' are adopted for efficient feature extraction. In our experiments, we find that applying spectral normalization [32], [33] in PSNet makes the training procedure more stable. To further stabilize the adversarial training, we apply gradient penalty [34] in the two discriminators of PSNet. Adam optimizer is adopted with a learning rate of 0.0001.

To process patches in an efficient way within the existing deep learning framework for training, for each image, a fixed number $\mathbb{N}_{p_s}$ of p_s are extracted and concatenated in an extra dimension. Once the number of actual masked patches is

TABLE III: The evaluation results of PatchMatch [1], GL [11], PICNet [35], Edge [14], DeepFill [4] and CRA [7] over Places [36] validation set. ↓ the lower the better, ↑ the higher the better. †As we cannot obtain the original trained weights of GL, we re-implemented this method according to their paper [11]

	random rectangle mask					random irregular mask				
	l_1 ↓	PSNR↑	SSIM↑	TV ↓	FID↓	l_1 ↓	PSNR↑	SSIM↑	TV ↓	FID↓
PatchMatch	14.795	15.038	0.819	**10.93**	11.630	11.276	17.387	0.839	**13.02**	10.751
GL†	13.806	15.659	0.821	12.04	10.379	10.269	17.403	0.855	13.50	8.295
PICNet	12.722	16.068	0.801	12.64	9.638	9.477	18.097	0.860	13.35	8.097
Edge	11.105	16.690	0.858	11.30	8.176	9.368	18.249	0.869	13.44	8.097
DeepFill	10.829	16.843	0.859	11.35	8.148	9.372	18.230	0.871	13.42	8.079
CRA	10.830	16.839	0.861	11.40	8.150	**9.260**	18.226	0.870	13.33	8.071
Our T-MAD	**10.334**	**17.203**	**0.867**	11.04	**8.131**	**9.261**	**18.351**	**0.873**	13.27	**8.058**

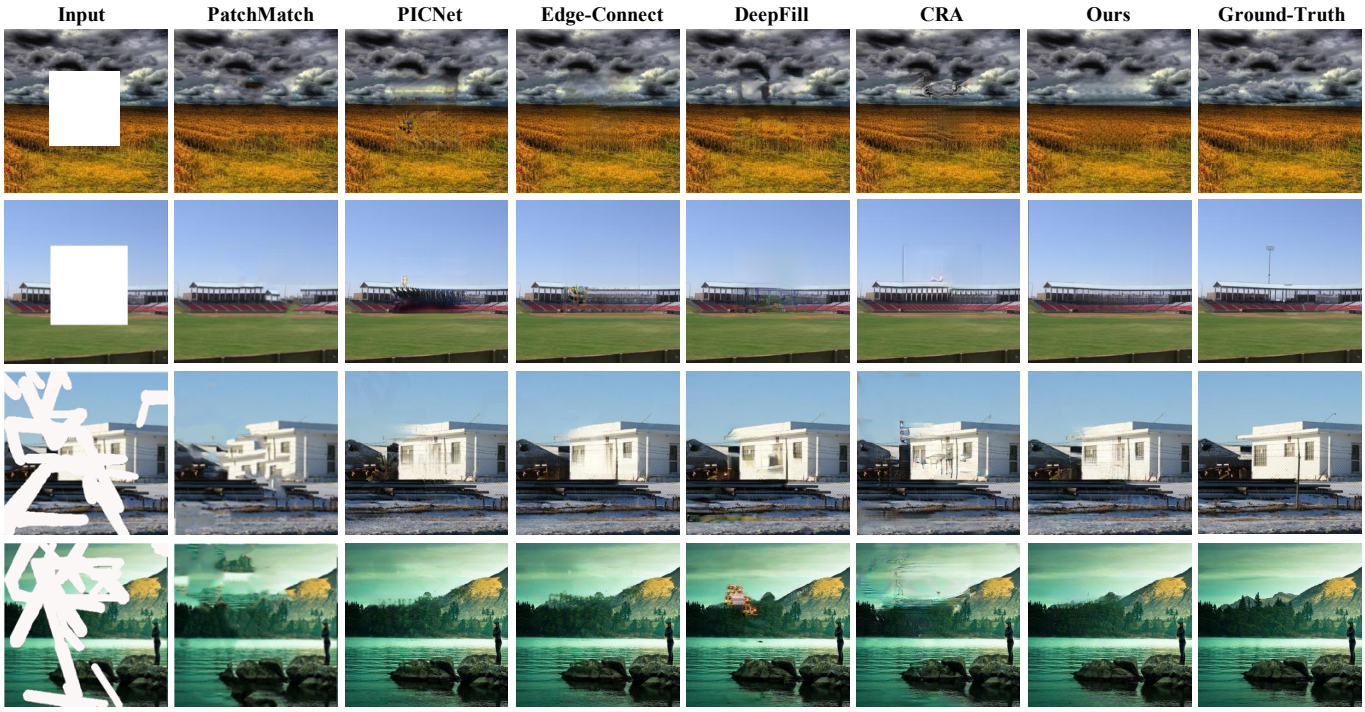


Fig. 3: The qualitative comparison with existing models. From left to right: Corrupted input image, results of PatchMatch [1], PICNet [35], Edge-Connect [14], DeepFill [4], CRA [7], our T-MAD and ground-truth. (**Best viewed with zoom-in**)

smaller than $\mathbb{N}_{p_s}$, we will randomly sample some valid patches from the unmasked region as complementary patches. As for testing, we can just choose the masked patches to process in patch synthesis stage. The coarse network is pre-trained for four epochs to obtain a reasonable initialization for the following joint training procedure. Note that the batch size for our PSNet is equal to the number of images in a batch multiplied with the number of hole patches in each image. We use four images in each batch and each image has different holes. The whole model is trained on four Titan V GPUs for three days. In training, we use images of resolution 256×256 with the largest hole size 128×128 in random positions. As for testing, given 512×512 input in V100, our method runs at 8 fps on average. Cosine restart scheduler is also adopted as our training scheduler. For each dataset mentioned below, we follow the official partition of the training and validation subsets.

As for the patch distribution loss described in Sec. III-C, we have the freedom of choosing different f and g for the general formulation like [26]. In this paper, we apply classical non-saturated GAN loss in [37]. The hyper-parameters of losses are presented in Tab. II.

A. Main Results

We evaluate the proposed T-MAD approach on three standard benchmarks Places [36], CelebA-HQ [31] and Paris Street-View [38]. Among the three datasets, Places is the most challenging benchmark with more than 400 natural scenes of diverse objects and textures. CelebA-HQ and Paris Street-View comprise highly-structured face and building images, respectively. Following the previous study [5], we use random rectangle masks with the same settings for evaluation. In addition, we also report results on Places with free-form irregular holes [4] to further demonstrate the effectiveness of T-MAD for handling various kinds of masks.

Baselines. We compare our methods with traditional optimization-based methods and contemporary deep inpainting methods:

- PatchMatch [1]: fills in the hole with the most optimal pathces from unmasked regions.

Fig. 4: Comparison of our model with PatchMatch [1], PIC [35], Edge-Connect [14], and DeepFill [4] in Paris Street-View[38] validation set. (**Best viewed with zoom-in**)

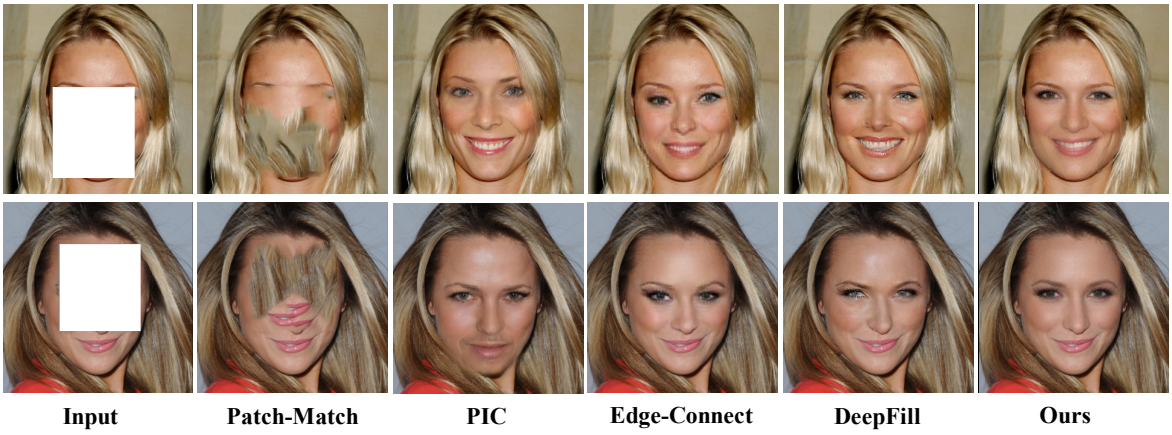


Fig. 5: Comparison of our model with PatchMatch [1], PIC [35], Edge-Connect [14], and DeepFill [4] in CelebA-HQ validation set. (**Best viewed with zoom-in**)

- GL [11]: introduces two discriminators to keep global and local consistency in the results.
- PICNet [35]: introduce random noises for generating diverse results with deep generative network.
- Edge [14]: adopts edge information to help reconstruct global structure.
- DeepFill [4]: combines gated convolution and contextual attention module for high-quality results. To extract patches from unmasked regions, the contextual attention module adopts softmax function to fuse all of the features from valid regions.
- CRA [7]: improves DeepFill [4] with multi-scale contextual attention mechanism using shared attention scores.

Quantitative Results. Following previous studies, we employ two kinds of metrics to measure distortion and perceptual quality, respectively. For distortion measurement, l_1 error, Peak Signal-to-Noise Ratio (PSNR), structural similarity index (SSIM) and total variation loss (TV loss) are adopted. As for the perceptual measurement, we use Fréchet Inception Distance (FID) [39] to show the Wasserstein-2 distance between two distributions. The evaluation results on Places validation set are reported in Tab. III. The proposed T-MAD achieves competitive results compared with previous works. The better performance in PSNR and SSIM demonstrates that the patch-based generation can also contribute effectively to structural recovery. The lower FID score suggests our method is more effective in synthesizing realistic texture on such challenging dataset. It is noted that PatchMatch is doing better on TV metric because the method directly copies raw image patches. Although raw image patches bring a lower total validation loss, the overall structure in PatchMatch's results are not satisfactory.

Qualitative Results. Qualitative results are shown in Fig. 3. It is observed that the traditional patch-based method PatchMatch [1] fills in the missing region with native texture patches but neglects the global structure. This is in concordance with its lower total variation loss but poorer performance on other metrics in Tab. III. Existing deep methods [14], [4], [7] perform better structural recovery but the generated textures still suffer from artifacts. On the contrary, our approach generates fine textures faithful to the unmasked regions, while obeying the global structure of the scene (*e.g.*, the highly structured building in the second case). In addition, DeepFill [4] and CRA [7] both apply contextual attention mechanism to borrow information from valid regions, which is similar with our texture memory in

TABLE IV: Ablation study on the number of candidates in the texture pool $\mathbb{N}_c$. 'W-Sum' denotes fusing $\{p_c\}$ by simple weighted-sum with correspondance map $\mathcal{S}$. ↓ the lower the better, ↑ the higher the better

	l_1 error↓	PSNR↑	SSIM↑	TV↓	FID↓
$\mathbb{N}_c = 1$	10.341	17.119	0.852	11.52	8.151
$\mathbb{N}_c = 2$	10.336	17.130	0.859	11.46	8.147
$\mathbb{N}_c = 4$	**10.334**	17.203	**0.867**	**11.23**	8.131
$\mathbb{N}_c = 6$	10.335	**17.210**	0.864	11.24	**8.130**
W-Sum	10.602	16.872	0.853	11.31	8.602

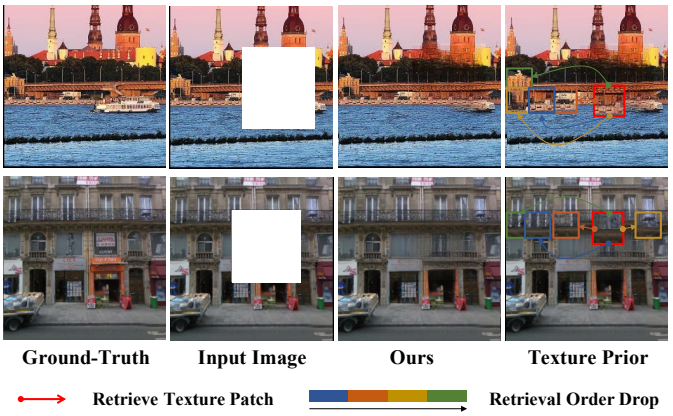


Fig. 6: We visualize the retrieval process of texture memory in the last column of this figure. The red color represents inpainted patches and other different colors represent the retrieval order of each p_c^i. For better visualization, we upsample the cropped patches

some extent. However, they use a softmax function to fuse all of the features from unmasked regions and directly generate results at image level, which causes unnatural artifacts. Thanks to the texture memory and patch synthesis, our T-MAD can easily handle more general cases in such large-scale dataset [36].

Our method also achieves outstanding perceptual quality in comparison to other methods on highly structured street views (Fig. 4) and faces (Fig. 5). Some interesting cases are observed. For corrupted street views, benefiting from the texture memory, our approach learns to borrow similar textures from unmasked appearance of a building, leading to high-quality textures coherent with the original image. In masked faces, unlike PatchMatch, our model hallucinates reasonable and high-quality contents for the missing parts. The success is partially attributed to the structural guidance offered by the coarse network. The texture memory, interestingly, also functions very well despite the absence of highly-identical patches from unmasked regions. The memory, in these examples, offers useful style information of the unmasked regions like complexion and facial texture for the completion.

B. Ablation Study

Effectiveness of Texture Memory. Figure 6 presents two actual examples to show which texture patches $p_c^i \in \mathcal{T}_c$ are retrieved from the texture memory $\mathcal{T}$. Our approach tends to select patches with similar color or material from the texture memory $\mathcal{T}$. In the first example of Fig. 6, T-MAD retains a

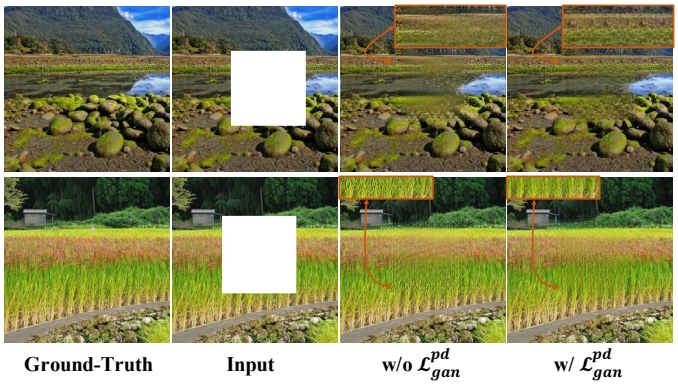


Fig. 7: Ablation study on the Patch Distribution loss. Important regions are upsampled for detailed comparison (**Best viewed with zoom-in**)

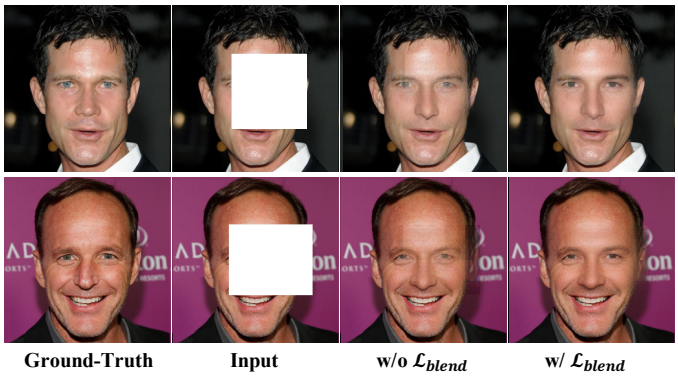


Fig. 8: Ablation study on blending loss. (**Best viewed with zoom-in**)

highly faithful texture recurrence with a reasonable semantic structure. It is interesting to see how texture synthesis can benefit from the statistical information provided by the texture memory $\mathcal{T}$.

The number of candidate texture patches $\mathbb{N}_c$ is a hyper-parameter in the texture retrieval module (Sec. III-B). We present the effect of different $\mathbb{N}_c$ in Tab. IV. The 'W-Sum' denotes fusing the texture pool with a simple weighted sum given the correspondence map $\mathcal{S}$. We can clearly see that the weighted sum of texture pool harms the performance in both structure and texture. Selecting fewer texture patches slightly impairs the FID measurement, indicating lower quality of the generated textures. We empirically found that increasing the number of texture patches beyond four does not bring more improvement as more repeated patches do not necessarily bring more statistical information of the unmasked region.

Effectiveness of Patch Distribution Loss. An improved adversarial loss at patch level is introduced in Sec. III-C. Figure 7 demonstrates the effectiveness of our Patch Distribution loss. We take the model without the relativistic components in Eq. (6) and Eq. (7) as the baseline for this ablation study. We fix other modules and train only PSNet for a fair comparison. As shown in Fig. 7, even though $\mathcal{T}$ contains useful texture patches, the model without patch distribution loss cannot learn to adopt the texture prior well.

Fig. 9: Influence of mask size on inpainting methods. The first row depicts the input with mask showing in white. DeepFill [4] (second row) generates undesired artifacts when the area of missing region increases. In contrast, our method (last row) is still capable of synthesizing realistic textures (**Best viewed with zoom-in**)

TABLE V: Quantitative results for our PSNet as a post-processing module in DeepFill [4] framework. ↓ the lower the better, ↑ the higher the better

	l_1 error ↓	PSNR ↑	SSIM ↑	FID ↓
DeepFill	10.829	16.843	0.854	8.148
DeepFill+PSNet	**10.690**	**16.967**	**0.862**	**8.142**

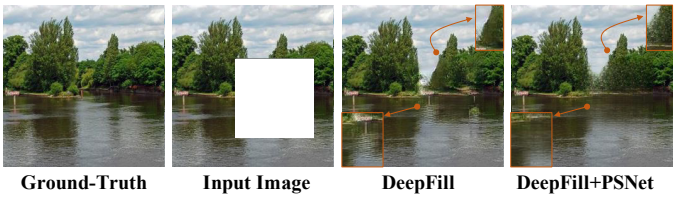


Fig. 10: Effectiveness for applying our PSNet as post-processing module for DeepFill. Important regions have been upsampled for detailed comparison

Effectiveness of Blending Loss. To reduce boundary artifacts and preserve the local consistency between neighboring patches, a blending loss $\mathcal{L}_{blend}$ is adopted in our approach (Eq. (9)). We show its importance in our case study on CelebA-HQ. As shown in Fig. 8, inpainting without the blending loss is susceptible to complex light conditions as well as reflective property of human faces.

Influence of Mask Size. In Fig. 9, we gradually increase the size of input mask to investigate the effects of mask size on inpainting methods. DeepFill [4] also adopts memory mechanism with Contextual Attention module. However, as the area of valid region reduces, the method cannot obtain useful information from unmasked regions while our T-MAD can make full use of a few selected patches to synthesize more realistic textures. Furthermore, the interpolation of texture representation in Contextual Attention module also causes undesired artifacts. On the contrary, our method performs better thanks to the more robust patch sampling (see Sec. III-B) in texture memory.

C. Applications

DeepFill + PSNet. As PSNet can also take inpainted images as the input, our patch synthesis module can be easily incorporated into recent inpainting models as an useful post-processing module. Taking DeepFill [4] method as an example, we adopt the results of DeepFill as input and fine-tune our PSNet with only 10,000 iterations. As shown in Tab. V, our PSNet improves the quantitative results with just a minor increase in computational cost. Figure 10 further demonstrates the effectiveness of PSNet as a post-processing module. With the help of PSNet, DeepFill can better preserve local consistency in both structure and texture. Importantly, the more realistic textures in water and trees verify the importance of applying texture memory in image inpainting.

Object Removal. Figure 11 shows the examples of applying our method to object removal. Users can brush in arbitrary shape and remove unwanted objects with our approach. The results suggest the generalizability of our approach in dealing with irregular missing regions.

High-Resolution Result. To further show the effectiveness of T-MAD with high-resolution input, we present some challenging results in Fig. 12. Even if with high-resolution input (512×768), our methods can generate high-quality texutres matching the original texture distribution.

D. Failure Cases

Some failure cases are shown in Fig. 13 for better understanding the limitation of our method. For most cases, the generated textures are reasonable but the global semantic structure are not fully recovered. It is because the coarse network fails to

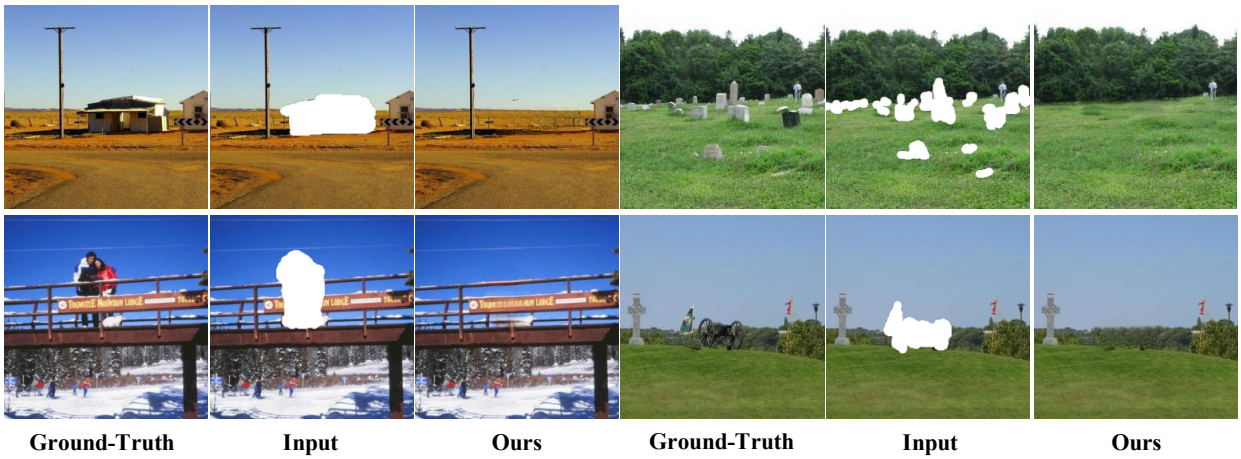


Fig. 11: Application for object removal. In each example, from left to right: original image, masked image with white irregular holes and our results (**Best viewed with zoom-in**)

Fig. 12: Additional results for high-resolution input. The images shown in this figure are taken from Places val set with their original resolution. (**Best viewed with zoom-in**)

provide a faithful coarse result due to the complex scenes in the wild. In another failure case that is shown at the bottom-right of Fig. 13, our method fails to hallucinate the whole body of the athlete due to the lack of semantic context.

V. Conclusion

We have proposed a novel method that bridges the classic notion of patch-based inpainting and deep learning-based image completion. Our method uniquely employs a texture memory that comes with an end-to-end trainable texture retrieval module to guide an improved texture generation in a deep inpainting framework. We also introduce a patch distribution loss to enhance texture synthesis at patch level. Better qualitative and quantitative results against both patch-based and contemporary deep learning-based methods are shown. We evisage that the proposed texture memory is not only applicable to image inpainting, but could also benefit other low-level vision tasks such as image super-resolution.

References

[1] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman, "Patchmatch: A randomized correspondence algorithm for structural image editing," in *ACM Transactions on Graphics (ToG)*, vol. 28, no. 3. ACM, 2009, p. 24.

[2] K. He and J. Sun, "Statistics of patch offsets for image completion," in *European Conference on Computer Vision*, 2012.

[3] G. Liu, F. A. Reda, K. J. Shih, T.-C. Wang, A. Tao, and B. Catanzaro, "Image inpainting for irregular holes using partial convolutions," in *European Conference on Computer Vision*, 2018.

[4] J. Yu, Z. Lin, J. Yang, X. Shen, X. Lu, and T. S. Huang, "Free-form image inpainting with gated convolution," in *IEEE International Conference on Computer Vision*, 2019.

[5] ——, "Generative image inpainting with contextual attention," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[6] Y. Ren, X. Yu, R. Zhang, T. H. Li, S. Liu, and G. Li, "Structureflow: Image inpainting via structure-aware appearance flow," in *IEEE International Conference on Computer Vision*, 2019.

[7] Z. Yi, Q. Tang, S. Azizi, D. Jang, and Z. Xu, "Contextual residual aggregation for ultra high-resolution image inpainting," in *IEEE International Conference on Computer Vision*, 2020.

[8] C. Yang, X. Lu, Z. Lin, E. Shechtman, O. Wang, and H. Li, "High-resolution image inpainting using multi-scale neural patch synthesis," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 6721–6729.

[9] O. Lotan and M. Irani, "Needle-match: Reliable patch matching under high uncertainty," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[10] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *International Conference on Learning Representations*, 2015.

Fig. 13: Failure cases of T-MAD

[11] S. Iizuka, E. Simo-Serra, and H. Ishikawa, "Globally and locally consistent image completion," *ACM Transactions on Graphics (ToG)*, vol. 36, no. 4, p. 107, 2017.

[12] Y. Zeng, J. Fu, H. Chao, and B. Guo, "Learning pyramid-context encoder network for high-quality image inpainting," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[13] X. Hong, P. Xiong, R. Ji, and H. Fan, "Deep fusion network for image completion," in *Proceedings of the 27th ACM International Conference on Multimedia*, 2019.

[14] K. Nazeri, E. Ng, T. Joseph, F. Qureshi, and M. Ebrahimi, "Edgeconnect: Generative image inpainting with adversarial edge learning," in *IEEE International Conference on Computer Vision Workshop*, 2019.

[15] W. Xiong, J. Yu, Z. Lin, J. Yang, X. Lu, C. Barnes, and J. Luo, "Foreground-aware image inpainting," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[16] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, 2017.

[17] X. Wang, R. Girshick, A. Gupta, and K. He, "Non-local neural networks," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[18] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[19] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 4, pp. 834–848, 2017.

[20] J. Wang, K. Chen, R. Xu, Z. Liu, C. C. Loy, and D. Lin, "Carafe: Content-aware reassembly of features," in *IEEE International Conference on Computer Vision*, 2019.

[21] J.-B. Huang, S. B. Kang, N. Ahuja, and J. Kopf, "Temporally coherent completion of dynamic video," *ACM Transactions on Graphics (TOG)*, vol. 35, no. 6, p. 196, 2016.

[22] P. Voigtlaender, Y. Chai, F. Schroff, H. Adam, B. Leibe, and L.-C. Chen, "Feelvos: Fast end-to-end embedding learning for video object segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[23] C.-Y. Wu, C. Feichtenhofer, H. Fan, K. He, P. Krahenbuhl, and R. Girshick, "Long-term feature banks for detailed video understanding," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 284–293.

[24] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks: Training deep neural networks with weight-straining deep neural networks with weights and activations constrained to+ 1 or-1," *arXiv preprint arXiv:1602.02830*, 2016.

[25] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," 2016.

[26] A. Jolicoeur-Martineau, "The relativistic discriminator: a key element missing from standard gan," *International Conference on Learning Representations*, 2018.

[27] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *IEEE International Conference on Computer Vision*, 2017.

[28] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[29] U. Demir and G. Unal, "Patch-based image inpainting with generative adversarial networks," *arXiv preprint arXiv:1803.07422*, 2018.

[30] T. R. Shaham, T. Dekel, and T. Michaeli, "Singan: Learning a generative model from a single natural image," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019.

[31] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of gans for improved quality, stability, and variation," in *International Conference on Learning Representations*, 2018.

[32] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, "Spectral normalization for generative adversarial networks," in *International Conference on Learning Representations*, 2018.

[33] A. Brock, J. Donahue, and K. Simonyan, "Large scale GAN training for high fidelity natural image synthesis," in *International Conference on Learning Representations*, 2019.

[34] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of wasserstein gans," in *Advances in Neural Information Processing Systems*, 2017.

[35] C. Zheng, T.-J. Cham, and J. Cai, "Pluralistic image completion," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[36] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba, "Places: A 10 million image database for scene recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 6, pp. 1452–1464, 2017.

[37] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems*, 2014.

[38] C. Doersch, S. Singh, A. Gupta, J. Sivic, and A. A. Efros, "What makes paris look like paris?" *ACM Transactions on Graphics (SIGGRAPH)*, 2012.

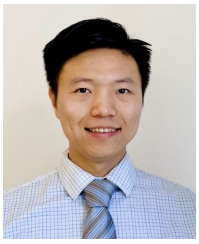
[39] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a local nash equilibrium," in *Advances in Neural Information Processing Systems*, 2017, pp. 6626–6637.

Rui Xu received the BEng degree in electronic engineering from Tsinghua University, in 2018. He is working toward the PhD degree in Multimedia Laboratory (MMLab) at the Chinese University of Hong Kong (CUHK). His research interests include computer vision and deep learning, especially for several topics in low-level vision like inpainting and image generation. He is also a core developer of MMEditing and OpenMMLab project.

Minghao Guo is a second-year graduate student in Chinese University of Hong Kong. He obtained his BEng major in Automation from Tsinghua University. His research interest lies in computer graphics and computer vision, specifically focusing on data-driven methods, physics-based simulation and inverse problems.

Jiaqi Wang is a fourth-year PhD candidate in Multimedia Laboratory (MMLab) at the Chinese University of Hong Kong. He received the BEng degree from Sun Yat-Sen University in 2017 and began to pursue his PhD degree funded by Hong Kong PhD Fellowship Scheme (HKPFS) in the same year. His research interests focus on object detection and instance segmentation. He has 5 papers accepted to CVPR/ICCV/ECCV. He is also a core developer of MMDetection. His team won the 1st place entry in COCO Object Detection Challenge in 2018 and 2019, respectively.

Xiaoxiao Li received his BEng degree in Department of Computer Science and Technology from Tsinghua University in 2014. He received the PhD degree in 2018 in Information Engineering, Chinese University of Hong Kong (CUHK). He is currently a senior researcher at Pinghu Technology Co., Ltd. His research interests include computer vision and machine learning, especially image segmentation and deep learning.



Bolei Zhou received the MPhil degree in information engineering from the Chinese University of Hong Kong and the PhD degree in computer science from Massachusetts Institute of Technology University, in 2018. He is an assistant professor in the Chinese University of Hong Kong. His research interests include computer vision and machine learning, with a focus on enabling machines to sense and reason about the environment through learning more interpretable and structural representations. He is an award recipient of the Facebook Fellowship, the Microsoft Research Asia Fellowship, and the MIT Greater China Fellowship.

Chen Change Loy (Senior Member, IEEE) received the PhD degree in computer science from the Queen Mary University of London, in 2010. He is an associate professor with the School of Computer Science and Engineering, Nanyang Technological University. Prior to joining NTU, he served as a research assistant professor with the Department of Information Engineering, The Chinese University of Hong Kong, from 2013 to 2018. His research interests include computer vision and deep learning. He serves as an associate editor of the IEEE Transactions on Pattern Analysis and Machine Intelligence and the International Journal of Computer Vision. He also serves/served as an Area Chair of CVPR 2021, CVPR 2019, ECCV 2018, AAAI 2021 and BMVC 2018-2020.