Efficient Layout-Guided Image Inpainting for Mobile Use

Wenbo Li, Yi Wei, Yilin Shen, Hongxia Jin
Samsung Research America
{wenbo.li1,yi.wei1,yilin.shen,hongxia.jin}@samsung.com

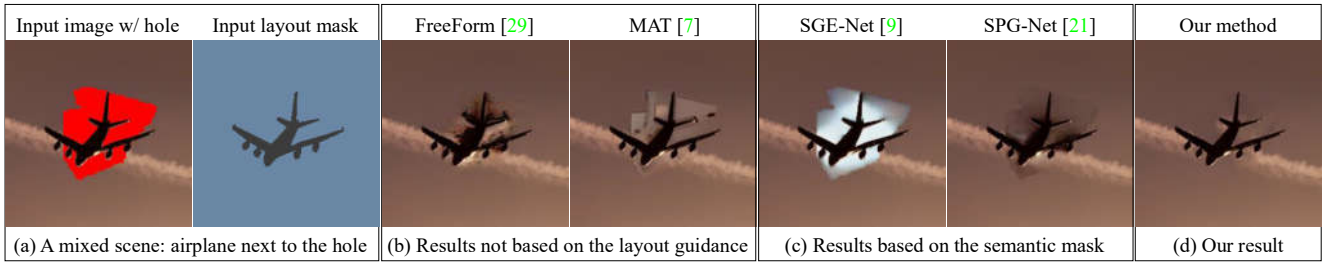


Figure 1. An illustration of a mixed scene and the influences of the layout guidance.

Abstract

The layout guidance, which specifies the pixel-wise object distribution, is beneficial to preserving the object boundaries in image inpainting while not hurting model's generalization capability. We aim to design an efficient and robust layout-guided image inpainting method for mobile use, which can achieve the robustness in presence of the mixed scenes where objects with the delicate shape reside next to the hole. Our method is made up of two sub-models, which restore the pixel-information for the hole from coarse to fine, and support each other to overcome the practical challenges encountered when making the whole method lightweight. The layout mask guides the two sub-models, which thus enables the robustness of our method in mixed scenes. We demonstrate the efficiency and robustness of our method via both the experiments and a mobile demo.

1. Introduction

In image inpainting, it remains a great challenge to preserve the boundaries of objects residing next to the hole, and we term such a scenario as the *mixed scene*, *e.g.*, Fig. 1 (a). For example, in Fig. 1 (b), the airplane boundary is contaminated during inpainting. The pixel-wise semantic labels, which compose a semantic mask, can provide the layout guidance for preserving object boundaries, so the topic of semantic-guided image inpainting attracts attention, *e.g.*, AIM 2020 Challenge on Image Extreme Inpainting [19]. However, the semantic guidance is in fact a double-edged sword. Its first disadvantage is that the pixel-information which a pretrained inpainting model can generate is limited by the semantic classes defined in the training dataset, which thus restricts the *generalization capability* of a semantic-guided method. The second disadvantage is that the bias of the appearance of a semantic class in the training dataset will further impair the *generalization capability* of a semantic-guided method. For example, in Fig. 1 (c), SGE-Net [11] preserves the airplane boundary well, but it generates the light blue sky for the hole instead of the reddish brown sky which is similar to the contexts beyond the hole. This is because the skies in the training dataset are usually light blue, such that the SGE-Net memorizes this bias in the training dataset. Compared to SGE-Net, SPG-Net [23] is a much more lightweight method, but it is also impacted by the bias memorization issue, which is demonstrated by the clear hole boundary in its result.

In order to circumvent the aforementioned two disadvantages in the generalization capability, we eliminate the semantic labels in a semantic mask to form a *layout mask*. The differences between a semantic mask and a layout mask are two-fold. First, the indices of a semantic class are fixed across different images according to the definition in the training dataset, which is not necessary for the case of a layout mask. This enables a layout-guided inpainting method which is able to generalize to novel semantic classes which have not been seen in the training dataset. In addition, the semantic-agnosticism of the layout mask liberates an inpainting method from memorizing the bias of the appearance of semantic classes. Second, a semantic mask fully covers the whole image plane, it is not necessary for a layout mask to do so. This enables multiple convenient ways of preparing the layout masks on a mobile device for image inpainting. Given a full layout mask output by a segmentation model, the hole can be filled either by the model inference, *e.g.*, the first stage of [23], or by user scribbling via a pen

associated with the mobile device. Rather than relying on a segmentation model to generate a full layout mask, users can use a pen, *e.g*., Apple Pencil, to sketch a partial layout mask which indicates the contexts for the hole. The allowance of user scribbling or sketching on the layout mask enable users to adjust the contour or shape of an object.

The goal of this paper is to design a robust and efficient layout-guided inpainting method for the mobile end which can overcome the shortcoming of the existing methods [7, 9, 11, 23, 27, 29, 31, 35] in mixed scenes. Thus, the design of our method is driven by challenges in making the model lightweight and robust in mixed scenes. One philosophy shared by the existing methods is to progressively restore the pixel-information or textures for the hole from outside to inside. This can be achieved by a neural network with a considerable depth, which leads to the large overhead. A much shallower network is required for the sake of efficiency, but reducing the network depth leads to insufficient receptive field for the network to handle large holes. Therefore, it is desirable that before an image is fed to the inpainting network, the hole can be roughly and efficiently restored by a nonparametric model, *e.g*., empirical diffusion methods [2, 3, 17], which would significantly relieve network's burdens for dealing with the void hole regions. However, the conventional diffusion methods are agnostic to content being propagated, so the propagated contents of different objects tend to be mixed for large holes, which causes artifacts. Thus, we propose a new layout-guided diffusion method which efficiently propagates the over-smoothed textures toward the hole regions.

After the rough restoration by the layout-guided diffusion method, we refine the pixel-information of the hole using a shallow refinement network. Between the encoder and decoder in the network, we attempt to add the widely used contextual attention mechanism [30] which can estimate the relevance of contexts, sample the potentially relevant contexts from the image plane, and aggregate them to refine the pixel-information of the hole. The contextual attention should be a necessary supplement to make up for the limited receptive field of our shallow network. Yet, the contextual attention exposes two drawbacks when being placed in the refinement model. The first drawback is that the estimation of the context relevance, *i.e*., attention weight, is unreliable for scenes with various semantic classes, causing the ambiguity artifacts. The second drawback is caused by our first effort to improve the efficiency, *i.e*., layout-guided diffusion method which outputs over-smoothed pixel-information. As such, the nearby hole regions of the same object share quite similar textures, and the texture diversity becomes even lower for these regions after the context aggregation for them, which causes the repetitive grid artifacts. To overcome these two drawbacks, we propose Top-X attention which samples the top-ranked relevant contexts with resort to the guidance of the layout mask, and use an adaptive dropout mechanism to enable the diversity for the sampled contexts. Thus, the Top-X attention and the layout-guided diffusion method support each other to make their respective efforts for efficiency and robustness meaningful.

As shown in Fig. 1, our method is robust in presence of mixed scenes thanks to the effective usage of the layout mask. We demonstrate the robustness of our method in mixed scenes against the others on a dataset merged from subsets of COCO [13] and ADE20K [37]. In terms of the efficiency, our method achieves at least 83.2% smaller in parameter size and the lowest computational cost. We prove the efficiency and the practicability of our method through a mobile demo on a Snapdragon 865+ with around 2 seconds on average in latency.

2. Related Works

Semantic-guided image inpainting is a problem formulation that is most related to ours, which attracts attention recently because the semantic mask can provide the layout guidance to address the mixed scenes. The Track 2 of AIM 2020 Challenge on Image Extreme Inpainting [19] is about the semantic-guided image inpainting, of which the results found that the semantic information can both increase and decrease the performance on the inpainting task, based on how its processing was implemented in the model. Therefore, it is nontrivial to study how to make use of the layout guidance. There are only a few existing methods focusing on the semantic-guided image inpainting. For example, Song *et al*. [23] proposed a two-stage network, in which the first stage restores the hole regions of the semantic mask, and the second stage restores the hole regions of the image. Liao *et al*. [11, 12] proposed to estimate the semantic mask on the fly, and the estimated mask is injected back to influence the intermediate feature maps. An inherent disadvantage of the semantic-guided image inpainting methods is the restricted generalization capability. This is because the neural network parameters that process the semantic mask (which is represented as a multi-channel tensor) are fixed after training, and when a new semantic class occurs (a new channel needs to be added in the tensor), in order for the neural networks to process the new semantic class, the network has to be retrained.

Empirical diffusion inpainting methods [1–3, 17] propagate the information of neighboring non-hole regions to the hole according to appearances, structures or both. As mentioned in §1, these nonparametric methods are not robust for large holes.

Influences of hole regions. Hole regions of the input image are usually initialized with zeros or mean values of the whole image. This often leads to artifacts such as color discrepancy and blur. To resolve this problem, Liu *et al*. [14]

propose the partial convolution which maintains a mask within a convolution operation to zero out values within the hole and normalize the convolution result, and updates the mask gradually following a predefined rule. Then, Xie *et al*. [27] and Yu *et al*. [31] improves the partial convolution by generalizing the rule-based mask estimation and updating mechanism to the learnable ones.

Coarse to fine. Apart from methods based on the one-pass U-Net [7, 14, 15, 27, 32], there are many methods following the coarse-to-fine pipeline [18, 22, 28–31]. Most of them include two stages. Their first stage focus on restoring the coarse information such as edges [18], the edge-preserved image structure [22], the contour of foreground object [28], monochromic image [25], and the coarse textures [5, 16, 21, 29–31, 33]. Their second stage restore the fine-grained textures based on the restored coarse information at the first stage. Compared to the edges [18], the layout mask contains richer layout information, *i.e*., the layout correspondence between the hole regions and the non-hole regions. With such a layout correspondence, it is much easier for a method to locate the valid non-hole contexts for each hole region.

Large holes. There are some recent works attempting to address the large hole challenge. Suvorov *et al*. [24] introduced the fast Fourier convolutions to image inpainting for the sake of accessing image-wide receptive field at early layers. Li *et al*. [9] introduced the transformer architecture to the image inpainting to capture the long-range dependencies of contexts. Zheng *et al*. [36] proposed the cascaded modulation GAN to capture both the long-range dependency and high-level semantics of an image. Li *et al*. [10] posed the image inpainting task as a filtering problem, and proposed a novel technique named multi-level interactive siamese filtering.

Shape-guided object inpainting is another topic that is related to our layout-guided inpainting problem. [4] first completes the invisible part of a segmentation mask, and then generates the invisible image textures based on the completed segmentation mask. The difference between the problem in [4] and our problem is that [4] focuses on inpainting one object at a time, while we aim to inpaint an arbitrary number of objects at the same time. [8] studies the shape-guided video object inpainting problem. Specifically, [8] makes use of the temporal information to complete the occluded object shape, and then uses the completed object shape to recover the object flows. Finally, [8] inpaint the occluded image textures based on the recovered object flows and the completed object shape. The difference between the problem in [8] and our problem is that [8] makes use of the additional temporal information to inpaint the object, while we have no such information available. Given an object shape, [34] first predicts its semantic class, and then based on the semantic class, it generates the image textures within the object shape. [34] aims to generate a novel object which does not appear in the input image while our method attempts to fill the partially missing regions of objects in the input image.

3. Method

Our method restores the textures for the hole from coarse to fine in two stages. In the first stage, the layout-guided diffusion method restores the coarse textures for the hole, and outputs the coarse image. In the second stage, the coarse image is refined by a refinement model which is equipped with the Top-X attention. The layout mask is used as input in both stages. The design philosophy of the two sub-models is to support each other in order to overcome the encountered practical challenges when making the whole method efficient and robust.

3.1. Layout-guided diffusion method

The reduction of the network's depth is of great benefit to the efficiency, but the resulting receptive field of the network will be overstretched for the large hole. In order to complement the shortcoming of the shallow network, we propose a layout-guided diffusion method, which can restore the coarse textures for the hole regions in an efficient nonparametric fashion. Specifically, the layout mask guides the diffusion method to propagate the textures of the non-hole regions to the hole regions of the same class.

Fig. 2 shows the pipeline of the layout-guided diffusion method, which is an iterative process with three sub-steps in each iteration. The hole regions are initialized with all zeros. There are three classes in the input image, which are colored in blue, green and orange, respectively. According to the layout mask, the hole regions belong to either the blue class or the green one. The orange class resides next to the hole, which thus leads to a mixed scene. In order to shield against the artifacts brought by mixing different classes, we split the image plane by the class distribution. In Fig. 2, we mark regions of the other classes with the brick patterns, which are fixed to zero constantly throughout the whole process.

Then, we aggregate the neighboring contexts as the content for the hole. The more distant a non-hole region is to a hole region, the lower impact it should have on this hole region. Thus, we employ a 2D Gaussian kernel of which the weights follow a 2D Gaussian distribution with the center placed at the square kernel center. For each hole region, we aggregate its contexts by performing the 2D convolution centered on it with the 2D Gaussian kernel, and we name such an operation as "Gaussian Blur", *i.e*., Sub-step ① in Fig. 2:

$$g(\mathbf{W}, \mathbf{H}) = \frac{1}{\sum_i^K \sum_j^K \mathbf{W}_{i,j}} \sum_i^K \sum_j^K (\mathbf{W} \odot \mathbf{H})_{i,j}, \qquad (1)$$

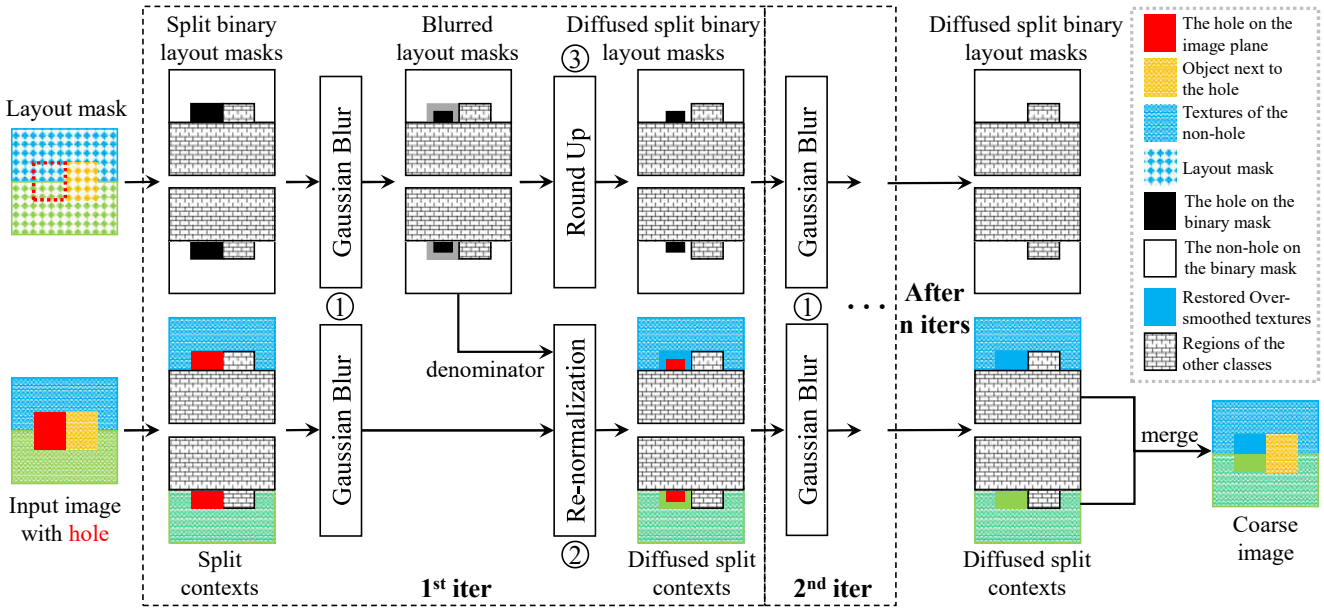


Figure 2. The iterative pipeline of the layout-guided diffusion method, which contains three steps within each iteration. Three semantic classes are marked in blue, green and orange, respectively. The white and black in the split binary layouts represent values 1 and 0, respectively. The gray in the blurred layouts represent values between 0 and 1.

where $\mathbf{W} \in \mathbb{R}_{>0}^{K \times K}$ represents a $K \times K$ Gaussian kernel, with each entry being positive. $\mathbf{H}$ represents a $K \times K$ slice of input, *i.e.*, split contexts or split binary layouts. $\odot$ denotes the Hadamard product. $g(\mathbf{W}, \mathbf{H})$ represents the Gaussian Blur operation which consists of two sub-operations, *i.e.*, the Hadamard product between $\mathbf{W}$ and $\mathbf{H}$, and the normalization through the division of the Hadamard product by the grand sum of $\mathbf{W}$. We set $K = 15$ in practice.

The context aggregation for a specific hole region should be exclusive to only the non-hole regions. In (1), the Hadamard product operation complies with this rule because of the zero values in hole regions. However, the normalization operation violates this rule because the grand sum considers all regions indiscriminately. Therefore, we need to re-normalize the results of Gaussian Blur by eliminating influences from other hole regions.

Thus, we design a re-normalization module, *i.e.*, Sub-step ② in Fig. 2, and we use the blurred layouts output by Gaussian Blur as the denominator for the re-normalization. Let $\mathbf{H}^S$ denote a slice of split binary layout, and we define an index set as $\Omega = \{i, j | \mathbf{H}^S_{i,j} = 1\}$. The blurred layout can thus be computed by inserting $\mathbf{H}^S$ into (1), removing the expression in the Hadamard product operation involving $\mathbf{H}^S_{i,j} = 0$, and reducing the expression in the Hadamard product operation involving $\mathbf{H}^S_{i,j} = 1$:

$$g(\mathbf{W}, \mathbf{H}^S) = \frac{1}{\sum_i^K \sum_j^K \mathbf{W}_{i,j}} \sum_{i,j \in \Omega} \mathbf{W}_{i,j}. \tag{2}$$

Then, the re-normalized aggregation result h' for a specific hole region can be computed by having (1) divide by (2).

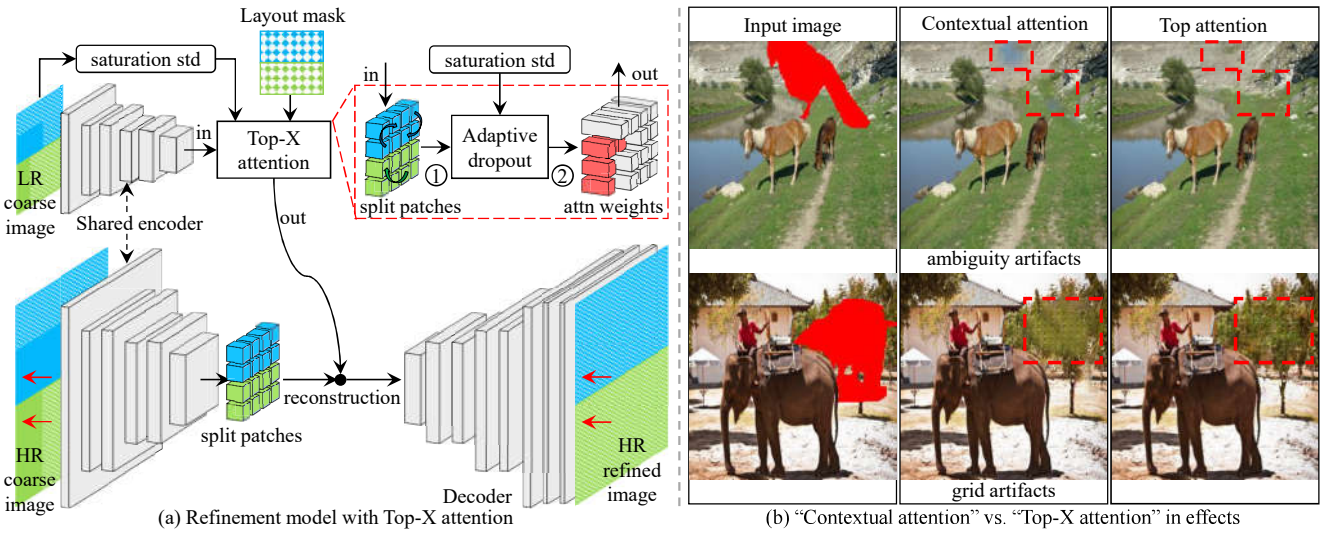
$$h' = \frac{1}{\sum_{i,j \in \Omega} \mathbf{W}_{i,j}} \sum_i^K \sum_j^K (\mathbf{W} \odot \mathbf{H})_{i,j}. \tag{3}$$

As shown in Fig. 2, these aggregation results constitute the diffused split contexts. In order for the new layout masks to correspond to the diffused split contexts, we round up the blurred regions[1] with values between 0 and 1 in the blurred layout mask to 1, *i.e.*, Sub-step ③ in Fig. 2. The diffused split contexts and diffused split binary layouts are fed to the next iteration as inputs. After several iterations, all coarse textures will be restored in the diffused split contexts which can be merged to form the coarse image.

3.2. Refinement model with Top-X attention

Given a coarse image restored by the layout-guided diffusion method as input, we use a refinement model to enrich its texture details. As shown in Fig. 3 (a), the refinement model consists of three major components, *i.e.*, *(i)* a shared encoder for encoding both low-resolution (LR) and high-resolution (HR) coarse images, *(ii)* Top-X attention module for aggregating the contexts for each region, and *(iii)* a decoder for reconstructing textures of the non-hole regions and restoring those of the hole regions, and outputs the HR refined image. Inspired by [35], in order to save the computational cost, we perform the attention estimation at low

[1]The blurred regions in the blurred layout masks correspond to the newly diffused area in the diffused split contexts.



(a) Refinement model with Top-X attention

(b) "Contextual attention" vs. "Top-X attention" in effects

Figure 3. ***(a)*** Refinement model with Top-X attention. The red dotted box encloses the detailed dissection of Top-X attention module. We mark the restored hole regions using red arrows, which can be observed via the over-smoothed textures (restored by the layout-guided diffusion method) on high-resolution (HR) coarse image and the normal textures on HR refined image. The two steps of Top-X attention module are marked with the circled numbers: ① marks the computation of layout-guided attention weights (reflected by the curved arrows pointing from and to the same object layout), and ② marks the adaptive dropout of the computed attention weights based on the saturation standard deviation computed from the input coarse image. The dropout regions of attention weights are highlighted in red. ***(b)*** "Contextual attention" vs. "Top-X attention" in effects. The first column shows images with holes (red). The red dotted boxes enclose the artifacts.

resolution. The shared encoder and the decoder are normal convolutional network of which the architecture details are presented in the supplementary.

We revisit the contextual attention mechanism [30] before diving into the introduction of Top-X attention mechanism. In order to exploit the useful global contexts which are far away from the holes, Yu *et al*. proposed the contextual attention mechanism, which aggregates and projects the information of the non-hole regions to hole regions according to the estimated patch similarities, *i.e*., attention weights. The role of contextual attention is crucial to us, because the global contexts sampled by the contextual attention can make up for the insufficiency of the receptive field size due to the shallow depth of the refinement model.

Let h_i and h_j denote the features at location i and j from the input feature maps, respectively. Let $s_{i,j}$ denote the attention weight of location i paid to location j, which can be computed as $s_{i,j} = \exp\left(\alpha \cdot \cos(h_i, h_j)\right) / \sum_k^N \exp\left(\alpha \cdot \cos(h_i, h_k)\right)$, where $\cos(\cdot)$ represent the cosine similarity function. α is a hyperparameter that enlarges the range of cosine similarity, and increases the attention paid to the relevant locations. In practice, α is set to 10.

The contextual attention has *two drawbacks*. First, it cannot exclude the influences from contexts of other classes, because the estimated attention weights are not accurate enough. This causes the ambiguity artifacts for scenes with various classes as shown in Fig. 3 (b). The second drawback is triggered by our layout-guided diffusion method which outputs the over-smoothed textures. So the nearby hole regions of the same class share quite similar textures. Thus, the contextual attention will give these regions similar attention weights over the similar sets of contexts, which causes the repetitive grid artifacts in Fig. 3 (b).

Top-X attention mechanism is proposed to overcome these two drawbacks. Suggested by its name, "Top-X", this mechanism randomly samples "X" number of top-ranked relevant contexts to aggregate for each region. As shown in Fig. 3 (b), the Top-X attention can resolve the ambiguity artifacts and grid artifacts effectively. As shown in Fig. 3 (a), Top-X attention includes two steps. The first step computes the layout-guided attention weight as

$$s^{\mathrm{X}}_{i,j} = \frac{\delta(o_i = o_j)\exp\left(\alpha \cdot \cos(h_i, h_j)\right)}{\sum_k^N \delta(o_i = o_k)\exp\left(\alpha \cdot \cos(h_i, h_k)\right)}, \tag{4}$$

where o_i denotes the index of the class to which the i^{th} patch belongs. $\delta(\cdot)$ is a delta function that outputs 1 when the condition is true, and 0 otherwise. By (4), only the patches of the same class, which are usually top-ranked, are considered to be the valid contexts. In this way, we can exclude the negative influences from the contexts belonging to the other classes. If a partial layout mask is provided, the class index for the white area without the layout guidance will be 0, and the class indices for areas with the layout guidance will be positive integers.

In order to resolve the grid artifacts in Fig. 3 (b), the second step of the Top-X attention is to randomly sample

Table 1. The quantitative experiments and efficiency comparison. ↑ (↓) means the higher (lower), the better. The best performances are highlighted in **bold**. Block named "Fair" ("Unfair") indicates methods with the same (different) inputs as ours. The asterisk means that a method only releases the executable file or API of its pretrained model or its training code is not useable. The "Unfair" block can show broader context of this comparison, and show more insights about the influences of the input semantic mask on other methods. THOP [38] can only measure the computational costs of deep learning models, so it cannot measure those for PatchMatch [1]. THOP [38] also failed to measure the computational costs of MAT [9] due to its unusual implementation fashion. We measure the latency and memory on a Nvidia Titan X GPU. The latency and memory of PatchMatch is unavailable because it runs on CPU.

	Method	FID ↓	SSIM ↑	Params (M) ↓	MACs (G) ↓	Latency (s) ↓	Memory (G) ↓
Ablation	Ours	**11.79**	**0.9587**	**0.48**	**8.20**	0.04	**0.62**
	Ours w/o layout-guided diffusion method	15.78	0.9555	-	-	-	-
	Ours w/o Top-X attention	12.98	0.9586	-	-	-	-
Fair	SPG-Net w/ mask [23]	16.66	0.9193	96.34	71.33	0.03	0.96
	LBAM w/ mask [27]	12.17	0.9215	-	-	-	-
	FreeForm w/ mask [31]	13.23	0.9213	-	-	-	-
	EdgeConnect w/ mask [18]	16.19	0.9554	-	-	-	-
	PatchMatch w/ mask [1]	13.47	0.9449	-	-	-	-
Unfair	SGE-Net [11]	23.20	0.7986	73.61	212.34	0.08	1.57
	LBAM [27]	13.60	0.9205	68.30	22.08	0.02	0.88
	FreeForm [31]	14.24	0.9212	16.17	99.08	0.06	0.76
	EdgeConnect [18]	17.39	0.9552	21.54	122.56	0.04	**0.62**
	PatchMatch [1]	20.92	0.9285	N/A	N/A	N/A	N/A
	DFNet* [7]	12.03	0.9568	32.88	9.71	**0.01**	0.74
	IterFdbk* [35]	12.60	0.9063	19.46	116.68	0.07	0.63
	CRA* [29]	21.57	0.8579	2.84	41.47	0.05	0.64
	MAT* [9]	14.16	0.7969	FAILED	FAILED	0.09	1.11

the top-ranked contexts for each hole region, such that we can avoid the grid artifacts by improving the diversity in the context aggregation for different regions. Thus, we propose the *adaptive dropout*. In the red dotted box of Fig. 3 (a), we illustrate that the (red) attention weights for hole regions are processed by the adaptive dropout module, and the number of such attention weights becomes fewer (shorter in length) than those attention weights for the non-hole regions. The dropout rate is determined adaptively according to the appearance similarities among hole regions. Intuitively, the higher appearance similarities are, the higher dropout rate is desired to ensure the diversity in the context aggregation. In practice, we observe that the standard deviation of the color saturation reflects the appearance dissimilarity properly, so we design a formula to estimate the dropout rate for an image based on its standard deviation of the saturation:

$$rate = \min\Big(\max\big((-std + \beta) \times \gamma, 0\big), 0.9\Big), \qquad (5)$$

where β and γ denote the bias and slope, respectively, and we empirically set $\beta = 80$ and $\gamma = 0.008$. The $\min$ and $\max$ functions clip the dropout rate to the range $[0, 0.9]$. We train the refinement model using the L1 loss. The training details can be found in the supplementary.

3.3. Efficiency discussion

The reasons why our method is lightweight and efficient are three-fold. First, our layout-guided diffusion method is nonparametric, and its three sub-steps (Gaussian Blur, Renormalization, and Round Up) are all very efficient. Second, thanks to the coarse image restored by the diffusion method, our refinement model can be very shallow because the network's burdens are significantly relieved. Third, we employ the shared encoder for our refinement model and perform the attention estimation at low resolution.

4. Experiments

Dataset. We process the semantic mask annotations to simulate the layout mask. There are no large-scale ($>$ 50K) inpainting dataset with the semantic mask annotations, so we collect the outdoor images from both COCO [13] and ADE20K [37] datasets, and create a dataset with nearly 53.4K training images and 1K test images. Similarly to [35], we prepare the hole regions by randomly overlaying $16,000$ real object instance masks onto the image plane, and the object masks are harvested from the instance segmentation annotations of COCO dataset. The reason why we do not use the dataset of AIM 2020 Challenge on Image Extreme Inpainting is because their online evaluation entry has been closed for years.

Evaluation metrics. We adopt two widely used quantitative evaluation metrics: Frechet Inception Distance (FID) [6] and Structural Similarity Index (SSIM) [26]. FID measures the authenticity of the restored textures. We use SSIM to measure how well the object boundaries are preserved in the inpainting result, so we calculate the SSIM on the image gradient level which can better reflect the edge sharpness than the RGB images do. As for measuring the storage and computational cost, we use THOP [38] to measure the parameter size for the storage cost, and measure the multiply-accumulate operations (MACs) for the computational cost.

Figure 4. Qualitative comparison in mixed scenes. The top table indicates the display order. Textures enclosed by the red contours in the first column are to be restored. Due to the space limit, the results of other methods and more results of these displayed methods are presented in the supplementary material.

All reported quantitative results are obtained from the resolution of 256×256.

4.1. Ablation study

For ablation study, we create two baselines by disabling each key component of our method: *(i)* To disable the layout-guided diffusion method, we follow the way of the majority of works to prepare the input to the model, *i.e.*, initializing the image hole regions with mean values, and concatenating the input image with a binary mask indicating the hole area. *(ii)* We replace the Top-X attention mechanism with the contextual attention mechanism.

Layout-guided diffusion method. As shown in Table 1, removing the diffusion method causes significant performance drop both quantitatively and qualitatively. This shows the importance of the efficient nonparametric diffusion method to our method, which makes significant contributions to both the robustness and efficiency. In addition, it

also supports our aforementioned claim that the reduction of network depth may cause the insufficiency of the receptive field size, which hurts the inpainting result. Note that we do not show the results of the diffusion method due to the space limit. Please refer to the supplementary for results.

Top-X attention. After we replace the Top-X attention with the contextual attention, we observe the obvious downgrade for FID and steady SSIM. In terms of FID, it means that the Top-X attention mechanism plays an important role in generating meaningful macro textures. The steady SSIM means that our diffusion method can effectively undertake the duty of preserving the delicate shapes of objects during inpainting.

4.2. Comparison with other inpainting methods

We compare our method with ten existing methods [1, 7, 9, 11, 18, 23, 27, 29, 31, 35]. Note that we do not compare our method with those participated in the AIM 2020 Challenge on Image Extreme Inpainting [19] because those methods do not release their code. Table 1 and Fig. 4 show their quantitative, qualitative and efficiency comparisons, respectively. They show that though our method yields much lower both computational and storage overhead, it still achieves the best image texture and structure quality. Among the eleven compared methods, there are seven of them [1, 9, 11, 18, 23, 27, 31] with the code available. Except [11][2] and [9][3], we modify the other five methods [1, 18, 23, 27, 31] to take exactly the same input as ours. To modify the deep learning methods [18, 23, 27, 31], we concatenate their original input with the semantic mask and adjust the input dimension of the first layer, and retrain them on our dataset. PatchMatch [1] is a conventional diffusion based method, so we use the layout mask to constrain its patch matching such that only the patches of the same semantic class can be used to fill the corresponding area within the hole. As for methods without code [7, 29, 35], we still show their results output by their executable file or API of their pretrained models, in order to provide readers with broader context of the comparison.

In Table 1 and Fig. 4, we use the postfix "w/ mask" to mark six methods which share the same input as ours. In Table 1, by comparing "FreeForm" against "FreeForm w/ mask", and "LBAM" against "LBAM w/ mask", we find that the semantic mask brings obvious benefits to the image quality in terms of FID, but it cannot significantly improve preserving the object boundaries according to the similar SSIMs before and after the input augmentation. As shown in Fig. 4 (a), even without the help of the layout mask, MAT [9] can preserve the boundary of the simple object, *i.e.*, the station platform. As for the complicated objects such as airplane in Fig. 4 (c), MAT yields obvious artifacts. We argue that this is because MAT is trained on the huge Places2 dataset, and has learned to preserve the boundary for the simple objects. SGE-Net [11] estimates the semantic mask, and applies the semantic mask via SPADE [20] to exert the structural influences. However, its results in Fig. 4 (especially the airplane example) show that SPADE helps SGE-Net to memorize data (the sky color) during the training, which supports our discussion in §1 regarding the generalization issue of using the semantic information. By comparing "PatchMatch" against "PatchMatch w/ mask" in Table 1 and Fig. 4, the layout mask significantly improves the inpainting quality of PatchMatch, especially in preserving the boundary clearness and generating reasonable textures for objects with simple textures, *e.g.*, Fig. 4 (c). However, "PatchMatch w/ mask" generates over-smoothed and artificial textures for objects with sophisticated textures, *e.g.*, station platform in Fig. 4 (a) and tree in Fig. 4 (b). All above shows the value of our method which enables an effective way of using the layout guidance to achieve better robustness and efficiency.

[2] [11] requires the input to have exactly three dimensions in order to be fed into a pretrained ResNet.

[3] Training code of [9] is not useable because it casts mysterious errors.

Figure 5. Generalization capability study. The semantic class "moon" and "mountain" are absent in the training dataset.

4.3. Generalization capability study

We perform the qualitative generalization capability study. As shown in Fig. 5, we aim to inpaint the occluded moon and mountain respectively, of which the semantic classes are absent in the training dataset. Fig. 5 shows that our method can successfully handle the out-of-domain cases thanks to the generalization capability brought by the layout mask.

5. Conclusion

We propose an efficient and robust layout-guided image inpainting model which consists of a layout-guided diffusion method and refinement model with Top-X attention. These two sub-models perform the inpainting from coarse to fine, and support each other to overcome challenges encountered when making the method efficient and robust. Our method achieves outstanding robustness in presence of the mixed scenes while maintaining high efficiency.

References

[1] Connelly Barnes, Eli Shechtman, Adam Finkelstein, and Dan B. Goldman. Patchmatch: a randomized correspondence algorithm for structural image editing. *TOG*, 2009. 2, 6, 8

[2] Marcelo Bertalmío, Guillermo Sapiro, Vicent Caselles, and Coloma Ballester. Image inpainting. In *SIGGRAPH*, 2000. 2

[3] Antonio Criminisi, Patrick Pérez, and Kentaro Toyama. Region filling and object removal by exemplar-based image inpainting. *TIP*, 2004. 2

[4] Kiana Ehsani, Roozbeh Mottaghi, and Ali Farhadi. Segan: Segmenting and generating the invisible. In *CVPR*, 2018. 3

[5] Xiefan Guo, Hongyu Yang, and Di Huang. Image inpainting via conditional texture and structure dual generation. In *ICCV*, 2021. 3

[6] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *NIPS*, 2017. 6

[7] Xin Hong, Pengfei Xiong, Renhe Ji, and Haoqiang Fan. Deep fusion network for image completion. In *ACMMM*, 2019. 2, 3, 6, 8

[8] Lei Ke, Yu-Wing Tai, and Chi-Keung Tang. Occlusion-aware video object inpainting. In *ICCV*, 2021. 3

[9] Wenbo Li, Zhe Lin, Kun Zhou, Lu Qi, Yi Wang, and Jiaya Jia. MAT: mask-aware transformer for large hole image inpainting. In *CVPR*, 2022. 2, 3, 6, 8

[10] Xiaoguang Li, Qing Guo, Di Lin, Ping Li, Wei Feng, and Song Wang. MISF: multi-level interactive siamese filtering for high-fidelity image inpainting. In *CVPR*, 2022. 3

[11] Liang Liao, Jing Xiao, Zheng Wang, Chia-Wen Lin, and Shin'ichi Satoh. Guidance and evaluation: Semantic-aware image inpainting for mixed scenes. In *ECCV*, 2020. 1, 2, 6, 8

[12] Liang Liao, Jing Xiao, Zheng Wang, Chia-Wen Lin, and Shin'ichi Satoh. Image inpainting guided by coherence priors of semantics and textures. In *CVPR*, 2021. 2

[13] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. In *ECCV*, 2014. 2, 6

[14] Guilin Liu, Fitsum A. Reda, Kevin J. Shih, Ting-Chun Wang, Andrew Tao, and Bryan Catanzaro. Image inpainting for irregular holes using partial convolutions. In *ECCV*, 2018. 2, 3

[15] Hongyu Liu, Bin Jiang, Yibing Song, Wei Huang, and Chao Yang. Rethinking image inpainting via a mutual encoder-decoder with feature equalizations. In *ECCV*, 2020. 3

[16] Hongyu Liu, Ziyu Wan, Wei Huang, Yibing Song, Xintong Han, and Jing Liao. PD-GAN: probabilistic diverse GAN for image inpainting. In *CVPR*, 2021. 3

[17] Olivier Le Meur, Josselin Gautier, and Christine Guillemot. Examplar-based inpainting based on local geometry. In Benoît Macq and Peter Schelkens, editors, *ICIP*, 2011. 2

[18] Kamyar Nazeri, Eric Ng, Tony Joseph, Faisal Z. Qureshi, and Mehran Ebrahimi. Edgeconnect: Generative image inpainting with adversarial edge learning. In *ICCVW*, 2019. 3, 6, 8

[19] Evangelos Ntavelis, Andrés Romero, Siavash Bigdeli, Radu Timofte, Zheng Hui, Xiumei Wang, Xinbo Gao, Chajin Shin, Taeoh Kim, Hanbin Son, Sangyoun Lee, Chao Li, Fu Li, Dongliang He, Shilei Wen, Errui Ding, Mengmeng Bai, Shuchen Li, Yu Zeng, Zhe Lin, Jimei Yang, Jianming Zhang, Eli Shechtman, Huchuan Lu, Weijian Zeng, Haopeng Ni, Yiyang Cai, Chenghua Li, Dejia Xu, Haoning Wu, Yu Han, S. M. Nadim Uddin, Hae Woong Jang, Soikat Hasan Ahmed, Jungmin Yoon, Yong Ju Jung, Chu-Tak Li, Zhi-Song Liu, Li-Wen Wang, Wan-Chi Siu, Daniel Pak-Kong Lun, Maitreya Suin, Kuldeep Purohit, A. N. Rajagopalan, Pratik Narang, Murari Mandal, and Pranjal Singh Chauhan. AIM 2020 challenge on image extreme inpainting. In *ECCVW*, 2020. 1, 2, 8

[20] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In *CVPR*, 2019. 8

[21] Jialun Peng, Dong Liu, Songcen Xu, and Houqiang Li. Generating diverse structure for image inpainting with hierarchical VQ-VAE. In *CVPR*, 2021. 3

[22] Yurui Ren, Xiaoming Yu, Ruonan Zhang, Thomas H. Li, Shan Liu, and Ge Li. Structureflow: Image inpainting via structure-aware appearance flow. In *ICCV*, 2019. 3

[23] Yuhang Song, Chao Yang, Yeji Shen, Peng Wang, Qin Huang, and C.-C. Jay Kuo. Spg-net: Segmentation prediction and guidance network for image inpainting. In *BMVC*, 2018. 1, 2, 6, 8

[24] Roman Suvorov, Elizaveta Logacheva, Anton Mashikhin, Anastasia Remizova, Arsenii Ashukha, Aleksei Silvestrov, Naejin Kong, Harshith Goka, Kiwoong Park, and Victor Lempitsky. Resolution-robust large mask inpainting with fourier convolutions. In *WACV*, 2022. 3

[25] Tengfei Wang, Hao Ouyang, and Qifeng Chen. Image inpainting with external-internal learning and monochromic bottleneck. In *CVPR*, 2021. 3

[26] Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *TIP*, 2004. 6

[27] Chaohao Xie, Shaohui Liu, Chao Li, Ming-Ming Cheng, Wangmeng Zuo, Xiao Liu, Shilei Wen, and Errui Ding. Image inpainting with learnable bidirectional attention maps. In *ICCV*, 2019. 2, 3, 6, 8

[28] Wei Xiong, Jiahui Yu, Zhe Lin, Jimei Yang, Xin Lu, Connelly Barnes, and Jiebo Luo. Foreground-aware image inpainting. In *CVPR*, 2019. 3

[29] Zili Yi, Qiang Tang, Shekoofeh Azizi, Daesik Jang, and Zhan Xu. Contextual residual aggregation for ultra high-resolution image inpainting. In *CVPR*, 2020. 2, 3, 6, 8

[30] Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas S. Huang. Generative image inpainting with contextual attention. In *CVPR*, 2018. 2, 3, 5

[31] Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas S. Huang. Free-form image inpainting with gated convolution. In *ICCV*, 2019. 2, 3, 6, 8

[32] Yanhong Zeng, Jianlong Fu, Hongyang Chao, and Baining Guo. Learning pyramid-context encoder network for high-quality image inpainting. In *CVPR*, 2019. 3

[33] Yu Zeng, Zhe Lin, Huchuan Lu, and Vishal M. Patel. Cr-fill: Generative image inpainting with auxiliary contextual reconstruction. In *ICCV*, 2021. 3

[34] Yu Zeng, Zhe Lin, and Vishal M. Patel. Shape-guided object inpainting. *CoRR*, abs/2204.07845, 2022. 3

[35] Yu Zeng, Zhe Lin, Jimei Yang, Jianming Zhang, Eli Shechtman, and Huchuan Lu. High-resolution image inpainting with iterative confidence feedback and guided upsampling. In *ECCV*, 2020. 2, 4, 6, 8

[36] Haitian Zheng, Zhe Lin, Jingwan Lu, Scott Cohen, Eli Shechtman, Connelly Barnes, Jianming Zhang, Ning Xu, Sohrab Amirghodsi, and Jiebo Luo. Image inpainting with cascaded modulation GAN and object-aware training. In Shai Avidan, Gabriel J. Brostow, Moustapha Cissé, Giovanni Maria Farinella, and Tal Hassner, editors, *ECCV*, 2022. 3

[37] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ADE20K dataset. In *CVPR*, 2017. 2, 6

[38] Ligeng Zhu. Thop: Pytorch-opcounter. 6