Supplementary Material: Resolution-robust Large Mask Inpainting with Fourier Convolutions

November 11, 2021

Contents

1 Evaluation

1.1 User study

There is no perfect metric to measure the quality of generated images. Thus, we alleviate a possible bias of selected metrics. We conducted a crowdsourced user study in two setups: *side-by-side* and *spot the mask*. In the *side-by-side* setup a user has to choose a more realistic inpainting out of two variants. The variants are provided by different methods for the same image and mask. An example of the crowdsourcer UI for *side-by-side* setup is presented on Figure 1. In the *spot the mask* setup users see only an inpainted image. Neither original image nor mask is provided. The user is asked to click on an image, pointing a part that is likely inpainted. If there are more than one inpainted region, a user has to point the one with more severe artifacts. An example of the crowdsourcer UI for *spot the mask* setup is presented on Figure 2.

The *side by side* setup aims mostly at comparison between different inpainting methods, while *spot the mask* also challenges the participants to distinguish real regions from inpainted ones. The quantitative results of the user study are present in Table 1.

Figure 1: Example of task for *side-by-side* setup for User Study. From left to right: first model prediction, original image with mask, second model prediction. The assessors need to select the most realistic prediction - left or right.

Figure 2: Example of task for *spot the mask* setup for User Study. From left to right: original image with mask, inpainted image. Note: assessors were only shown the right image and were asked to click on the most suspicious part.

	Narrow masks		**Wide masks**	
Method	RP ↑	Acc ↓	RP ↑	Acc ↓
LaMa-Fourier (ours)	50	34±1.7	50	54±1.7
LaMa-Dilation (ours)	48±2.5	37±1.7	46±2.4	55±1.9
CoModGAN	41±2.3	36±1.8	53±2.4	53±1.8
MADF	48±2.5	33±1.7	36±2.4	64±1.8
AOT GAN	43±2.4	39±1.9	25±2.1	77±1.6
GCPR	37±2.3	41±1.8	30±2.2	71±1.6
HiFill	20±1.9	45±1.9	22±2.1	73±1.6
DeepFill v2	38±2.4	41±1.8	37±2.3	57±1.8
EdgeConnect	31±2.2	42±1.8	22±2.0	66±1.8
Region-wise inp.	43±2.3	35±1.8	33±2.3	56±1.7
Region norm inp.	34±2.3	43±1.9	17±1.7	66±1.7

Table 1: Results of the user study on Places dataset in 512×512 resolution demonstrate that the inpainting produced by our method is more preferable and less detectable compared to most methods. While MADF comes close on narrow masks, it is uncooperative on wide ones. The CoModGAN performs better on wide masks than the LaMa-Fourier, and is worse on the narrow masks, this makes us hypothesise that methods are close in the performance on wide masks. In this case, we need more samples to estimate standard deviation. We would like to note that LaMa-Fourier (27M params) has significantly less trainable parameters than CoModGAN (109M params) and MADF (85M params). RP states for the relative preference score in comparison with LaMa-Fourier in the *side by side* setup. The score is expressed in percents. RP$=$50 means that the user cannot distinguish between a method and LaMa-Fourier. RP <50 means that inpainting results of a method are less realistic compared to LaMa-Fourier. Acc is the percent of correctly localized inpainted areas in *spot the mask* setup. Metrics are calculated separately for narrow and wide masks. The best values are marked bold. For RegionWise Inpainting, DeepFillv2, EdgeConnect, we report only the best metrics of the two models pre-trained or re-trained model. The standard deviations are obtained with bootstrap [2].

Quality control of the user study To prevent adaptation to the task, we set a limit to the maximum number of 5 pages per assessor. For *side-by-side* task each sample was labeled independently by 3 assessors, and for *spot the mask* by 5. In *side-by-side* task the assessors were shown 3 pictures: original image in the center with applied mask and two images inpainted with different methods on the left and right. Assessors were asked to select the most realistic inpainted image out of two. In *spot the mask* the assessors were only shown an inpainted image—no original image or mask is provided—and they were asked to click on the most suspicious part of the image. Final score is obtained as percent of samples on which assessors guessed the mask position correctly.

1.2 Places - Full Metrics

Detailed metrics for all models on Places are presented in Table 2. Columns titled "40-50% masked" contain metrics calculated using the most hard samples in a test set — samples with 40-50% area of an image covered by a mask. Columns "All samples" contain metrics calculated with all samples regardless of masked area. These numbers help to better understand robustness of various models and training setups.

	Places (512 × 512)													
	Narrow masks				**Medium masks**				**Wide masks**				**Segm. masks**	
	40-50% masked		All samples		40-50% masked		All samples		40-50% masked		All samples		All samples	
	FID ↓	LPIPS ↓	FID ↓	LPIPS ↓	FID ↓	LPIPS ↓	FID ↓	LPIPS ↓	FID ↓	LPIPS ↓	FID ↓	LPIPS ↓	FID ↓	LPIPS ↓
LaMa-Fourier	12.7	0.168	0.63	0.090	11.7	0.212	1.30	0.112	12.0	0.243	2.21	0.135	5.35	0.058
LaMa-Dilated	13.3▲5%	0.171▲2%	0.68▲8%	0.091▲1%	13.1▲12%	0.215▲2%	1.60▲22%	0.114▲1%	14.2▲18%	0.246▲1%	2.81▲27%	0.136▲1%	5.54▲3%	0.058▲1%
LaMa-Regular	12.4▼2%	0.167▼1%	0.60▼5%	0.089▼1%	12.3▲4%	0.215▲1%	1.37▲5%	0.114▲1%	17.0▲41%	0.252▲4%	3.51▲59%	0.139▲3%	5.69▲6%	0.059▲3%
LaMa-Fourier-Shallow	13.4▲6%	0.175▲4%	0.72▲13%	0.094▲4%	12.2▲4%	0.219▲3%	1.39▲7%	0.116▲3%	12.4▲3%	0.248▲2%	2.31▲5%	0.138▲2%	5.61▲5%	0.060▲4%
LaMa-Regular-Deep	12.6▼1%	0.167	0.63	0.090	12.5▲6%	0.214▲1%	1.58▲21%	0.114▲1%	13.5▲12%	0.247▲2%	2.62▲18%	0.137▲2%	5.59▲4%	0.059▲2%
LaMa-Regular (narrow train masks)	12.7	0.168	0.68▲7%	0.091▲1%	15.1▲29%	0.222▲5%	1.92▲47%	0.117▲5%	23.5▲95%	0.261▲7%	5.41▲145%	0.144▲7%	6.50▲22%	0.062▲8%
CoModGAN [11]	16.3▲28%	0.206▲23%	0.82▲30%	0.111▲23%	12.4▲6%	0.239▲13%	1.34▲3%	0.128▲14%	10.4▼14%	0.261▲7%	1.82▼18%	0.147▲9%	6.40▲20%	0.066▲14%
AOT GAN [9]	14.1▲11%	0.173▲3%	0.79▲25%	0.091▲1%	15.9▲36%	0.224▲6%	2.29▲75%	0.119▲6%	24.4▲103%	0.269▲11%	5.94▲169%	0.149▲11%	7.34▲37%	0.063▲10%
RegionWise [3]	15.5▲22%	0.191▲14%	0.90▲42%	0.102▲14%	17.0▲45%	0.234▲11%	2.42▲86%	0.125▲11%	21.3▲77%	0.269▲10%	4.75▲115%	0.149▲11%	7.58▲42%	0.066▲14%
DeepFill v2 [8]	17.9▲41%	0.197▲17%	1.06▲68%	0.104▲16%	18.3▲56%	0.244▲15%	2.68▲106%	0.130▲16%	22.1▲84%	0.278▲14%	5.20▲135%	0.155▲15%	9.17▲71%	0.068▲18%
EdgeConnect [4]	18.9▲49%	0.205▲22%	1.33▲110%	0.111▲23%	21.9▲86%	0.250▲18%	3.66▲181%	0.135▲20%	30.5▲153%	0.284▲17%	8.37▲279%	0.160▲19%	9.44▲76%	0.073▲27%
RegionWise [3] (wide train masks)	14.1▲11%	0.180▲7%	0.74▲17%	0.095▲6%	14.8▲26%	0.229▲8%	1.91▲47%	0.121▲8%	17.2▲43%	0.259▲7%	3.56▲61%	0.144▲7%	6.70▲25%	0.064▲11%
DeepFill v2 [8] (wide train masks)	19.3▲51%	0.200▲19%	1.35▲114%	0.107▲19%	18.3▲56%	0.238▲12%	2.72▲109%	0.127▲13%	19.2▲60%	0.264▲9%	4.34▲96%	0.148▲10%	7.77▲45%	0.066▲15%
EdgeConnect [4] (wide train masks)	28.9▲127%	0.264▲57%	2.78▲339%	0.141▲56%	23.2▲97%	0.259▲22%	3.91▲200%	0.140▲25%	30.0▲149%	0.284▲17%	7.94▲259%	0.160▲19%	NAN	NAN

Table 2: Detailed metrics for all models on the Places dataset.

1.3 CelebA-HQ - Full Metrics

Detailed metrics for all models on CelebA-HQ are presented in Table 3. Note that the "40-50%" columns, which contain metrics on the most difficult samples from the test sets: these are samples with more than 40% of images covered by masks. These numbers help to better understand robustness of various models and training setups.

	CelebA-HQ (256 × 256)											
	Narrow masks				**Medium masks**				**Wide masks**			
	40-50% masked		All samples		40-50% masked		All samples		40-50% masked		All samples	
	FID ↓	LPIPS ↓	FID ↓	LPIPS ↓	FID ↓	LPIPS ↓	FID ↓	LPIPS ↓	FID ↓	LPIPS ↓	FID ↓	LPIPS ↓
LaMa-Fourier	22.7	0.132	7.26	0.085	34.1	0.145	6.13	0.080	27.8	0.168	6.96	0.098
LaMa-Dilated	25.1▲10%	0.145▲10%	8.75▲21%	0.095▲12%	38.8▲14%	0.159▲10%	7.02▲14%	0.087▲9%	29.6▲7%	0.176▲5%	7.62▲9%	0.105▲7%
LaMa-Regular	22.5▼1%	0.139▲5%	7.21▼1%	0.089▲5%	34.9▲2%	0.151▲4%	6.41▲4%	0.084▲5%	29.4▲6%	0.177▲6%	7.31▲5%	0.104▲5%
LaMa-Fourier-Shallow	25.0▲10%	0.143▲8%	7.96▲10%	0.092▲9%	35.9▲5%	0.153▲5%	6.56▲7%	0.085▲6%	30.0▲8%	0.175▲4%	7.31▲5%	0.103▲4%
LaMa-Dilated-Shallow	24.3▲7%	0.148▲12%	7.86▲8%	0.096▲13%	36.4▲7%	0.155▲7%	6.50▲6%	0.086▲8%	28.7▲3%	0.177▲5%	7.14▲3%	0.104▲6%
LaMa-Regular-Deep	22.8▲1%	0.137▲3%	7.50▲3%	0.089▲5%	35.2▲3%	0.149▲3%	6.53▲6%	0.083▲3%	28.9▲4%	0.173▲3%	7.34▲5%	0.102▲4%
LaMa-Regular (narrow train masks)	23.4▲3%	0.139▲5%	7.46▲3%	0.090▲6%	34.8▲2%	0.152▲5%	6.51▲6%	0.084▲5%	29.6▲7%	0.177▲5%	7.42▲7%	0.103▲5%
CoModGAN [11]	35.9▲58%	0.139▲5%	16.8▲131%	0.079▼7%	48.4▲42%	0.169▲16%	19.4▲216%	0.092▲15%	64.4▲132%	0.191▲13%	24.4▲250%	0.102▲4%
AOT GAN [9]	21.0▼7%	0.127▼4%	6.67▼8%	0.081▼4%	39.1▲15%	0.162▲11%	7.28▲19%	0.089▲11%	40.4▲46%	0.204▲21%	10.3▲48%	0.118▲20%
RegionWise [3]	32.5▲43%	0.188▲42%	11.1▲53%	0.124▲46%	40.4▲18%	0.179▲24%	7.52▲23%	0.101▲25%	33.9▲22%	0.205▲22%	8.54▲23%	0.121▲23%
DeepFill v2 [8]	37.0▲63%	0.201▲52%	12.5▲73%	0.130▲53%	45.3▲33%	0.189▲30%	9.05▲48%	0.105▲31%	43.0▲55%	0.214▲28%	11.2▲61%	0.126▲28%
EdgeConnect [4]	29.2▲29%	0.156▲18%	9.61▲32%	0.099▲17%	40.5▲19%	0.174▲20%	7.56▲23%	0.095▲19%	34.7▲25%	0.205▲22%	9.02▲30%	0.120▲22%
RegionWise [3] (wide train masks)	47.5▲109%	0.246▲86%	17.9▲147%	0.164▲94%	50.9▲49%	0.220▲51%	10.3▲67%	0.124▲55%	42.6▲54%	0.233▲39%	11.2▲61%	0.140▲42%
DeepFill v2 [8] (wide train masks)	30.4▲34%	0.169▲28%	9.99▲38%	0.108▲27%	40.3▲18%	0.173▲19%	7.65▲25%	0.095▲19%	34.6▲24%	0.196▲16%	8.95▲29%	0.115▲17%
EdgeConnect [4] (wide train masks)	55.5▲144%	0.248▲88%	18.3▲152%	0.152▲79%	40.2▲18%	0.174▲20%	7.79▲27%	0.097▲22%	32.7▲18%	0.196▲17%	8.43▲21%	0.116▲18%

Table 3: Detailed metrics for all models on CelebA-HQ dataset. Columns titled "40-50% masked" contain metrics calculated using the most hard samples in a test set — samples with 40-50% area of an image covered by a mask. Columns "All samples" contain metrics calculated with all samples regardless of masked area.

2 Masks

2.1 Random Mask Generation Algorithm

```
from np.random import uniform

def gen_large_mask(img_h, img_w, n):
    """ img_h:    int, an image height
        img_w:    int, an image width
        marg:     int, a margin for a box starting coordinate
        p_irr:    float, 0 <= p_irr <= 1, a probability of a polygonal chain mask

        min_n_irr: int, min number of segments
        max_n_irr: int, max number of segments
        max_l_irr: max length of a segment in polygonal chain
        max_w_irr: max width of a segment in polygonal chain

        min_n_box: int, min bound for the number of box primitives
        min_n_box: int, max bound for the number of box primitives
        min_s_box: int, min length of a box side
        max_s_box: int, max length of a box side"""

    mask = ones(img_h, img_w)

    if np.random.uniform(0,1) < p_irr: # generate polygonal chain
        n = uniform(minn_irr, maxn_irr) # sample number of segments

        for _ in range(n):
            y = uniform(0, img_h) # sample a starting point
            x = uniform(0, img_w)

            a = uniform(0, 360) # sample angle
            l = uniform(10, max_l_irr) # sample segment length
            w = uniform(5, max_w_irr) # sample a segment width

            # draw segment starting from (x,y) to (x_,y_) using brush of width w
            x_ = x + l * sin(a)
            y_ = y + l * cos(a)

            gen_segment_mask(mask, start=(x, y), end=(x_, y_), brush_width=w)
            x, y = x_, y_
    else: # generate Box masks
        n = uniform(min_n_box, min_n_box) # sample number of rectangles

        for _ in range(n):
            h = uniform(min_s_box, max_s_box) # sample box shape
            w = uniform(min_s_box, max_s_box)

            x_0 = uniform(marg, img_w - marg + w) # sample upper-left coordinates of box
            y_0 = uniform(marg, img_h - marg - h)

            gen_box_mask(mask, size=(img_w, img_h), masked=(x_0, y_0, w, h))
    return mask
```

Listing 1: The mask generation algorithm.

2.2 Segmentation Mask Generation Algorithm

In addition to random irregular masks we used segmentation-based masks, to ensure that our conclusions made with synthetic irregular masks are also valid for real-world objects, shapes and sizes. The **Segm** mask set aims on modeling a real-world application of object removal, e.g. in a photo editor. We used two datasets constructed in a similar way — one for validation and model selection purposes and another for final evaluation — but with disjoint sets of images.

Segmentation-based validation and test sets were constructed using a segmentation-based mask generator. This mask generator extracts silhouettes of foreground objects using Detectron2 [6] from Places test_large images, and randomly superimposes one of them onto 1,000 images sampled and curated from Places val_large so as to include mostly structural, man-made shapes in their background scenes. We constructed the validation subset similarly—using object silhouettes extracted

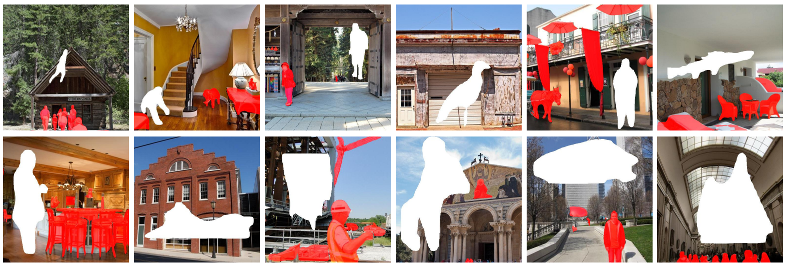

Figure 3: Examples from the **Segm** test set for **Places**. First row: examples with the 0-10% masked area range. Second row: examples with the 10-30% masked area range. (Existing object regions at Step 2 are marked red. These regions are further used to superimpose an object silhouette in the background region at Step 5. White area: target mask hole to inpaint. Note that red markings are shown here only for a visualization purpose and do not appear in the actual ⟨image, mask⟩ pairs.)

from test_large and images sampled from val_large, ensuring the test set and the validation set are strictly disjoint.

10-30% masked area range: In total 2,000 ⟨image, mask⟩ pairs were created. Here, we sampled 500 512×512 crop images for a 10-20% masked area range, and 500 512×512 crop images for a 20-30% masked area range. Then, each of the crop images was coupled with two random masks with hole sizes within 10-15% and 15-20% over the 10-20% range, or 20-25% and 25-30% over the 20-30% range.

0-10% masked area range: Another group of 2,000 ⟨image, mask⟩ pairs was created. Here, we reused the same 1,000 crop images that had been created at the 10-30% range, where each crop image was coupled with two random masks within 0-5% and 5-10%.

The detailed process of constructing the Segm test set is described as follows:

1. **Prepare original images of structural scenes:** We choose images from 157 curated scene categories[1] from Places val_large, which more likely have structural, man-made complex shapes in their background scenes

2. **Mark existing object regions on the images at Step 1:** We apply Detectron2 object detector (ex. red regions shown in Figure 3) and filter masks by foreground categories.

3. **Create a pool of foreground object silhouettes:** We apply Detectron2 object detector to images from Places test_large and filter masks by foreground categories.

4. **Choose target images at 10-20% (or 20-30%):** First, we randomly sample hundreds of images from those prepared at Step 1, which can fit a hole in the size of 10-20% (or 20-30%) avoiding existing objects marked at Step 2. Then, we manually filter out a few inappropriate[2] images. Finally, we randomly choose the final 500 images from the rest

[1] airplane_cabin, airport_terminal, alcove, alley, amphitheater, amusement_park, apartment_building/outdoor, aqueduct, arcade, arch, archive, art_gallery, artists_loft, assembly_line, atrium/public, attic, auditorium, bakery/shop, balcony/exterior, balcony/interior, ballroom, banquet_hall, barndoor, basement, basketball_court/indoor, bathroom, bazaar/indoor, bazaar/outdoor, beach_house, bedchamber, bedroom, berth, boardwalk, boathouse, bookstore, booth/indoor, bow_window/indoor, bowling_alley, bridge, building_facade, bus_interior, bus_station/indoor, cabin/outdoor, campus, canal/urban, candy_store, carrousel, castle, chalet, childs_room, church/indoor, church/outdoor, closet, conference_center, conference_room, construction_site, corridor, cottage, courthouse, courtyard, delicatessen, department_store, diner/outdoor, dining_hall, dining_room, doorway/outdoor, dorm_room, downtown, driveway, elevator/door, elevator_lobby, elevator_shaft, embassy, entrance_hall, escalator/indoor, fastfood_restaurant, fire_escape, fire_station, food_court, galley, garage/outdoor, gas_station, gazebo/exterior, general_store/indoor, general_store/outdoor, greenhouse/outdoor, gymnasium/indoor, hangar/outdoor, hardware_store, home_office, home_theater, hospital, hotel/outdoor, hotel_room, house, hunting_lodge/outdoor, industrial_area, inn/outdoor, jacuzzi/indoor, jail_cell, kasbah, kitchen, laundromat, library/indoor, library/outdoor, lighthouse, living_room, loading_dock, lobby, lock_chamber, mansion, manufactured_home, mausoleum, medina, mezzanine, mosque/outdoor, movie_theater/indoor, museum/outdoor, nursery, oast_house, office, office_building, office_cubicles, pagoda, palace, pantry, parking_garage/indoor, parking_garage/outdoor, pavilion, pet_shop, porch, reception, recreation_room, restaurant_patio, rope_bridge, ruin, sauna, schoolhouse, server_room, shed, shopfront, shopping_mall/indoor, shower, skyscraper, staircase, storage_room, subway_station/platform, synagogue/outdoor, television_room, temple/asia, throne_room, tower, train_station/platform, utility_room, waiting_room, wet_bar, youth_hostel

[2] Including none or very little portion of structural shapes (ex. image is mostly covered with the sky, sea, or woods)/ Huge human portrait covering the whole image/ Capture of another photo (ex. from a magazine)/ Thick outer frames superimposed/ Text caption visibly superimposed/ CG rendered image/ No meaningful content available within (ex. only cloudy textures given)/ Quality issues (ex. dark, over-exposed, blurry, etc. at extreme level)

Figure 4: Comparison of 256×256 mask statistics produced by different random mask generators with different settings. *DeepFillv2* correspond to the statistics of the training masks that are produced by DeepFillv2 generator. *LaMa Training* correspond to our training mask generator. *Test Narrow*, *Medium* and *Wide* correspond to the statistics of 256×256 CelebA test sets. Masked area is an average number of masked pixels per image. Width is calculated as a distance to the closest known pixel, averaged over all masked pixels in an image.

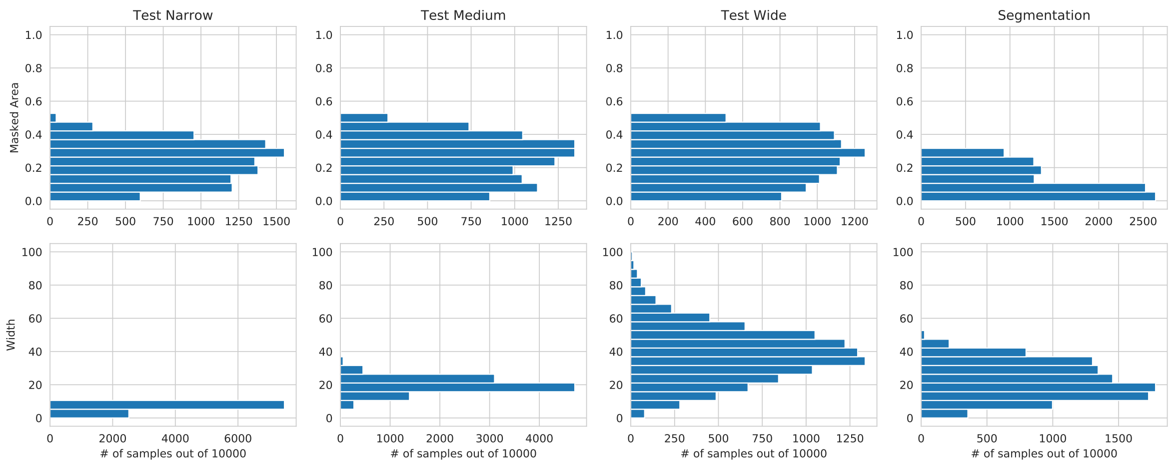

Figure 5: Comparison of 512×512 mask statistics produced by different random mask generators with different settings. *Test Narrow*, *Medium* and *Wide* correspond to the statistics of Places 512×512 test sets with random irregular masks. **Segmentation** reflect statistics of Places 512×512 test set with segmentation-based masks. Masked area is an average number of masked pixels per image. Width is calculated as a distance to the closest known pixel, averaged over all masked pixels in an image.

5. **Create $\langle$image, mask$\rangle$ pairs:** For each of the 10-20% and 20-30% area ranges, we randomly crop 512×512 regions out of each image from Step 4. For 0-10% masked area range, we reuse same images as for 10-20% and 20-30%. For each crop, the mask generator superimposes an object silhouette taken from the pool prepared at Step 3 onto the background region, by avoiding existing object regions marked at Step 2.

2.3 Masks Settings and Statistics

Table 4 contains settings of random irregular mask generator that we use to train and evaluate our models. We use "256-Train" settings during training. The configuration "256-Narrow" is only used in ablation study to show importance of wide and diverse mask generation during training (Table 4 in paper).

Figure 4 contains descriptive statistics of the masks produced by different mask generation algorithms and different settings. Our masks are much more aggressive and diverse compared to those of

		p_irr	Irregular Masks				Box-shaped masks				
			min_n_irr	max_n_irr	max_l_irr	max_w_irr	min_n_box	max_n_box	min_s_box	max_s_box	marg
256	Narrow*	1	4	50	40	10	-	-	-	-	-
	Medium	0.77	4	5	100	50	1	5	10	50	0
	Wide	0.77	1	5	200	100	1	3	30	150	10
	Train	0.5	1	5	200	100	1	4	30	150	10
512	Narrow	1	4	70	100	20	-	-	-	-	-
	Medium	0.77	4	10	200	100	1	5	30	150	0
	Wide	0.77	1	5	450	250	1	4	30	300	10

Table 4: Parameters for random mask generation algorithm. Our models are trained with ”256-Train” settings. *”256-Narrow” roughly correspond to the settings used in DeepFillv2 and EdgeConnect repositories.

DeepFillv2. To obtain each chart, we generated 10000 samples and measured percentage of masked area and mask width. Masked area corresponds to the ratio of masked pixels to total image area. Width correponds to the average distance from each masked pixel to its closest known neighbor (calculated using Euclidean Distance Transform).

3 Dataset splits

3.1 Places

Training To train most of our models, we use all high resolution images (approximately 512×512) from Places-Standard[3].

Validation To conduct in-training evaluation, to track overfitting and to choose the best checkpoint, we prepared a validation set consisting of 2000 image-mask pairs. Images for validation set were randomly sampled from high resolution validation subset of Places [4]. Masks for validation set were prepared using segmentation-based mask generation algorithm.

Test To conduct final evaluation, we prepared four test sets—three with irregular random masks of different widths (narrow, medium, thick) and one with segmentation-based masks. Test sets with random masks contain 30000 image-mask pairs and segmentation-based set contains 4000 pairs. All images were randomly sampled from high resolution test part of Places [5].

3.2 CelebA-HQ

We use the train-val split used in DeepFill [6].

Training We use full training subset except 2000 images, which were held out for validation.

Validation To conduct in-training validation, to control overfitting and to select the best checkpoint, we extract 2000 images from the training set. For each image in validation subset, we generate three random masks.

Test To conduct final evaluation, we used full ”val” subset according to DeepFill split (see footnote). Mask sets were prepared using random irregular generator with three different settings—to produce narrow, medium and wide masks.

[3] Places Standard Train Large http://data.csail.mit.edu/places/places365/train_large_places365standard.tar
[4] Places Standard Validation Large http://data.csail.mit.edu/places/places365/val_large.tar
[5] Places Standard Test Large http://data.csail.mit.edu/places/places365/test_large.tar
[6] https://drive.google.com/drive/folders/1lpluFXyWDxTY6wcjixQGWX8jxUUMlyBW

4 Big LaMa 51M Examples

4.1 Big LaMa 51M positive examples

Please refer to Figure 6 and the anonymous URL in the caption for more positive examples.

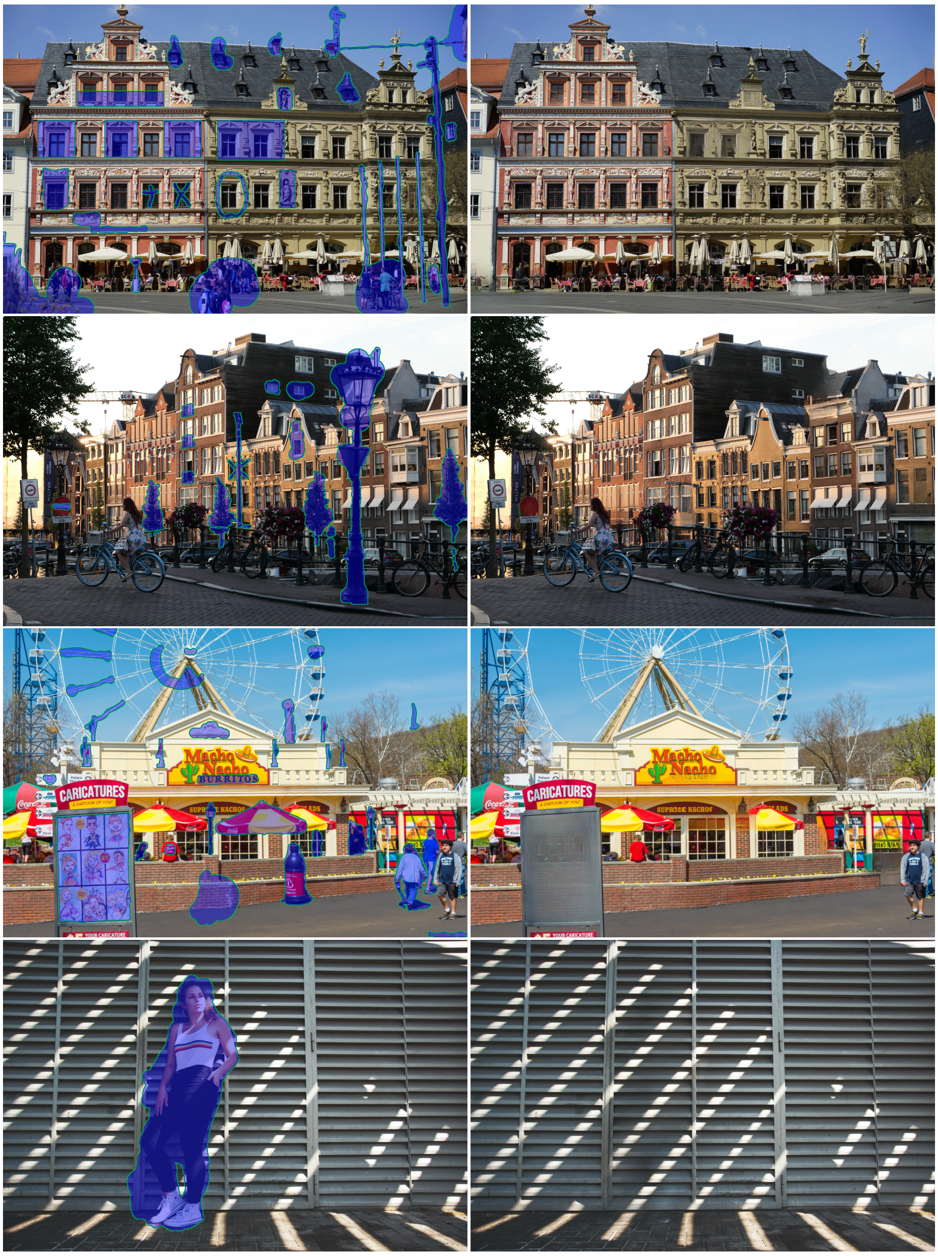

Figure 6: Big LaMa 51M positive examples. More examples can be found at the anonymous link `https://bit.ly/3k0gaIK`.

4.2 Big LaMa 51M negative examples: Distortions, Bokeh, Perspective

Please refer to Figure 7 and the anonymous URL in the caption for more failure cases.

Figure 7: Big LaMa 51M negative examples: perspective distortion, complex backgrounds. More examples can be found at the anonymous link `https://bit.ly/3k0gaIK`.

4.3 Big LaMa 51M domain transfer examples

Please refer to Figure 8, 11 and the anonymous URL in the caption for more cases of successful generalization to unseen domains.

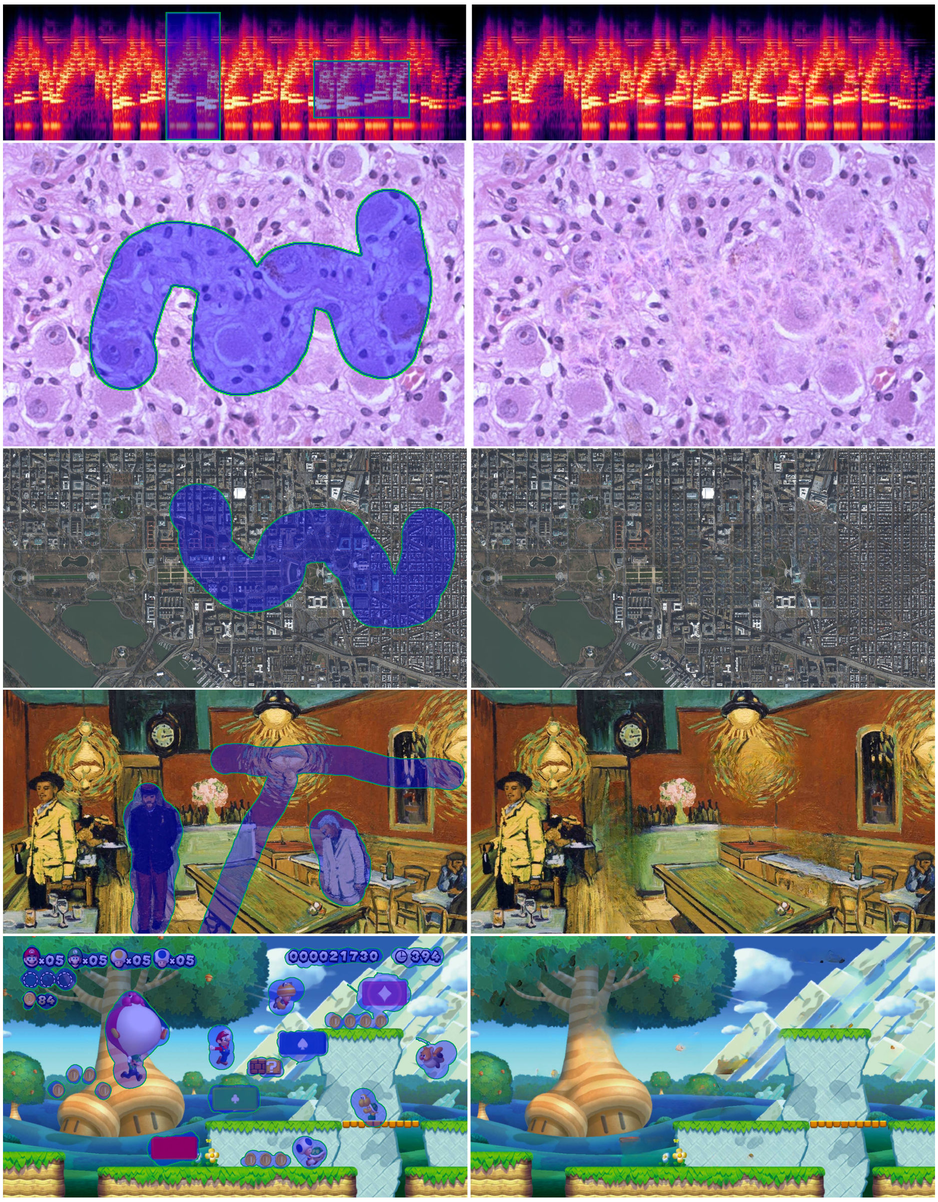

Figure 8: Big LaMa 51M examples, domain generalization: music spectrogram, hystology image, bird-eye view, Van Gogh painting, computer game. The method was trained on Places Challenge dataset and never see such kind of data, still is able to generate reasonable inpaintings.

5 Discriminator

```
NLayerDiscriminator(
  (model0): Sequential(
    (0): Conv2d(3, 64, kernel_size=(4, 4), stride=(2, 2), padding=(2, 2))
    (1): LeakyReLU(negative_slope=0.2, inplace=True)
  )
  (model1): Sequential(
    (0): Conv2d(64, 128, kernel_size=(4, 4), stride=(2, 2), padding=(2, 2))
    (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): LeakyReLU(negative_slope=0.2, inplace=True)
  )
```

Figure 9: Big LaMa 51M examples, more examples of domain generalization: outpainting, MRI. The method was trained on Places Challenge dataset and never saw such kind of data, yet it is able to generate reasonable inpaintings.

```
  (model2): Sequential(
    (0): Conv2d(128, 256, kernel_size=(4, 4), stride=(2, 2), padding=(2, 2))
    (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): LeakyReLU(negative_slope=0.2, inplace=True)
  )
  (model3): Sequential(
    (0): Conv2d(256, 512, kernel_size=(4, 4), stride=(2, 2), padding=(2, 2))
    (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): LeakyReLU(negative_slope=0.2, inplace=True)
  )
  (model4): Sequential(
    (0): Conv2d(512, 512, kernel_size=(4, 4), stride=(1, 1), padding=(2, 2))
    (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): LeakyReLU(negative_slope=0.2, inplace=True)
  )
  (model5): Sequential(
    (0): Conv2d(512, 1, kernel_size=(4, 4), stride=(1, 1), padding=(2, 2))
  )
)
```

Listing 2: We used the following discriminator architecture for all LaMa models.

6 Perceptual Losses Comparison Details

In this section, we describe the networks that were used as the feature extractors for perceptual losses in the ablation study. The ResNet-based perceptual losses exploit the encoder part of PSPNet model [10] as a feature extractor[7].

We used the following variants of the base network:

(a) The ResNet50 with regular convolutions, that was pretrained on the classification ImageNet dataset.

(b) The (a) model that is dilated post-hoc [1, 7]—the dilation of 2 is applied to the convolutions of the third residual block, and the dilation of 4 is applied to the the convolutions of the fourth block, while weights remain the same.

(c) The (b) model that is equipped with a decoder network, and is trained on a segmentation problem on ADE20K dataset.

We evaluated the perceptual losses based in networks from steps $a - c$ in the ablation study. In all cases we used outputs of all four residual blocks as the features for the perceptual loss. For the classification-based perceptual loss, we used VGG-19 model [5][8]. In VGG network, perceptual loss uses all activations from the first thirteen ReLUs.

We performed the selection of the perceptual loss weight α using the coordinate-wise beam-search strategy separately for each variant. For final weights see Table 5.

	Model	Pretext Problem	Dilation	Weight
$\mathcal{L}_{HRFPL}$	RN50	Segm.	+	30
$\mathcal{L}_{\text{Clf}PL}$	RN50	Clf.	+	1
	RN50	Clf.	-	1
	VGG19	Clf.	-	0.1

Table 5: The best weights for each perceptual loss variant. The RN states for ResNet50 arihitecture. ClfPL Regular states for (a) network, ClfPL Dilated states (b) network, HRFPL—a high receptive field perceptual losses—states for (c) model.

[7] https://github.com/CSAILVision/semantic-segmentation-pytorch
[8] https://pytorch.org/vision/stable/models.html#torchvision.models.vgg19

7 LaMa-Dilated Details

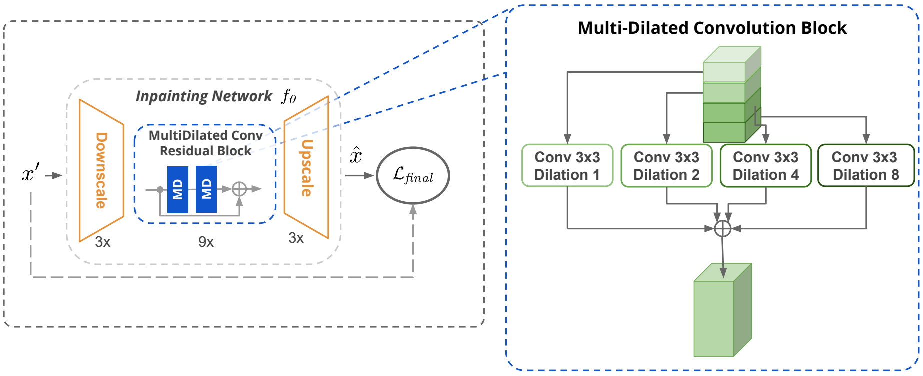

Figure 10: The architecture of LaMa Dilated network. The model is almost the same as LaMa Regular, but regular convolutions in all residual blocks are substituted with **MultiDilated Convolution Blocks**. Specifically, the input of each convolution block is split to four equal parts channel-wise. Then, the regular convolution layer with appropriate padding and the chosen dilation size is applied for each part separately. Finally, results of all four blocks are summed up.

8 Inference time comparison

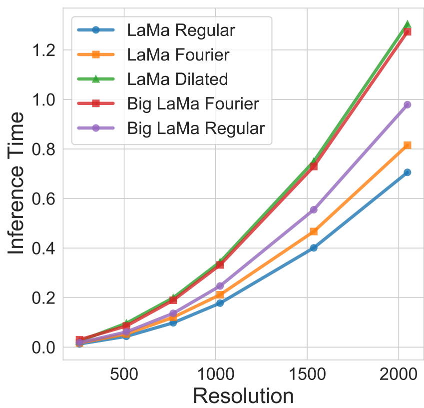

Figure 11: Inference time in sec/image of various inpainting techniques depending on resolution. The results obtained on Nvidia 1080Ti, with batch size of 100 that fully loads GPU for all methods. The results are averaged over 100 runs.

References

[1] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. In Yoshua Bengio and Yann LeCun, editors, *Proc. ICLR*, 2015.

[2] Bradley Efron and Robert J Tibshirani. *An introduction to the bootstrap*. CRC press, 1994.

[3] Yuqing Ma, Xianglong Liu, Shihao Bai, Lei Wang, Aishan Liu, Dacheng Tao, and Edwin Hancock. Region-wise generative adversarial image inpainting for large missing areas. *arXiv preprint arXiv:1909.12507*, 2019.

[4] Kamyar Nazeri, Eric Ng, Tony Joseph, Faisal Z Qureshi, and Mehran Ebrahimi. Edgeconnect: Generative image inpainting with adversarial edge learning. *arXiv preprint arXiv:1901.00212*, 2019.

[5] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[6] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. `https://github.com/facebookresearch/detectron2`, 2019.

[7] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015.

[8] Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas S Huang. Free-form image inpainting with gated convolution. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4471–4480, 2019.

[9] Yanhong Zeng, Jianlong Fu, Hongyang Chao, and Baining Guo. Aggregated contextual transformations for high-resolution image inpainting. In *Arxiv*, pages –, 2020.

[10] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2881–2890, 2017.

[11] Shengyu Zhao, Jonathan Cui, Yilun Sheng, Yue Dong, Xiao Liang, Eric I Chang, and Yan Xu. Large scale image completion via co-modulated generative adversarial networks. In *International Conference on Learning Representations (ICLR)*, 2021.