AIM 2020 Challenge on Image Extreme Inpainting

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, Uddin S. M. Nadim, Hae Woong Jang, Soikat Hasan Ahmed, Jungmin Yoon, Yong Ju Jung, Chu-Tak Li, Zhi-Song Liu, Li-Wen Wang, Wan-Chi Siu, Daniel P.K. Lun, Maitreya Suin, Kuldeep Purohit, A. N. Rajagopalan, Pratik Narang, Murari Mandal, and Pranjal Singh Chauhan

Abstract. This paper reviews the AIM 2020 challenge on extreme image inpainting. This report focuses on proposed solutions and results for two different tracks on extreme image inpainting: classical image inpainting and semantically guided image inpainting. The goal of track 1 is to inpaint large part of the image with no supervision. Similarly, the goal of track 2 is to inpaint the image by having access to the entire semantic segmentation map of the input. The challenge had 88 and 74 participants, respectively. 11 and 6 teams competed in the final phase of the challenge, respectively. This report gauges current solutions and set a benchmark for future extreme image inpainting methods.

Keywords: Extreme Image Inpainting, Image Synthesis, Generative Modeling

1 Introduction

Image inpainting is the task of recovering regions with some level of corrupted or missing regions. Normally, these regions are the outcome of degradation, artifacts, or they were altered by human intervention such as whitening object removal or image manipulation. The goal of this task is to fill in the missing pixels of the image, so the generated pixels are harmonious and perceptually plausible looking with the rest of the image.

E. Ntavelis (entavelis@ethz.ch, ETH Zurich and CSEM SA), A. Romero, S. Bigdeli, and R. Timofte are the AIM 2020 challenge organizers, while the other authors participated in the challenge.

Appendix A contains the authors' teams and affiliations.

AIM webpage: `http://www.vision.ee.ethz.ch/aim20/`

Github webpage: `https://github.com/vglsd/AIM2020-Image-Inpainting-Challenge`

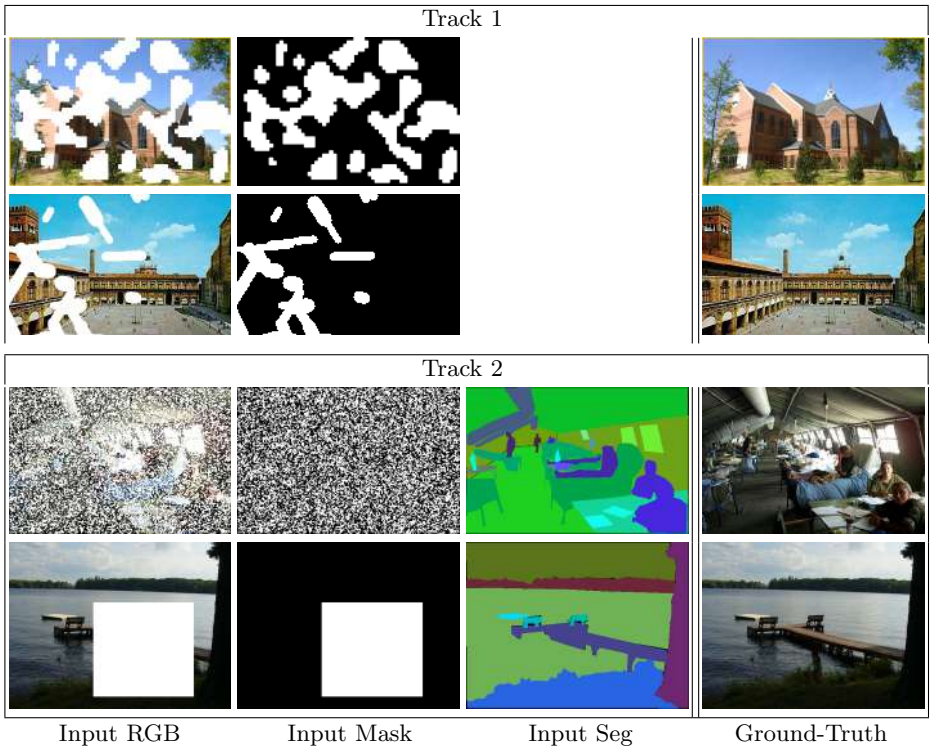

Fig. 1: Visual example of the extreme degraded input images and ground truth images used in the challenge. In both tracks, we automatically degraded each image to simulate different levels of extreme degradation. In Track 1 the aim is to produce realistic images using the Input RGB and Input Mask, and no additional information. Track 2 uses in addition a semantic segmentation map (Input Seg) as condition information for the generation of well defined objects.

Since the introduction of Generative Adversarial Networks (GANs) [10], recent efforts on image inpaiting have produced impressive results for replacing objects [32,33], retouching landscapes [24,4], or altering the content of a scene [11,22], either by given weak supervision (semantic segmentation guiding in addition to the binary mask to reconstruct) or no supervision (only using binary reconstruction mask). Interestingly, it is common to assume that the inpainted region is small (*e.g.* squared bounding boxes in general) with respect to the entire image, which leads to expected high perceptual scores.

In our AIM 2020 Challenge on Image Extreme Inpainting , we set the first extreme image inpainting challenge that aims at generating photo-realistic and perceptually appealing inpainted regions. The contestants are called to design a solution that is able to complete images of various sizes, spanning from low to high resolution, where the number of missing pixels is significant with respect

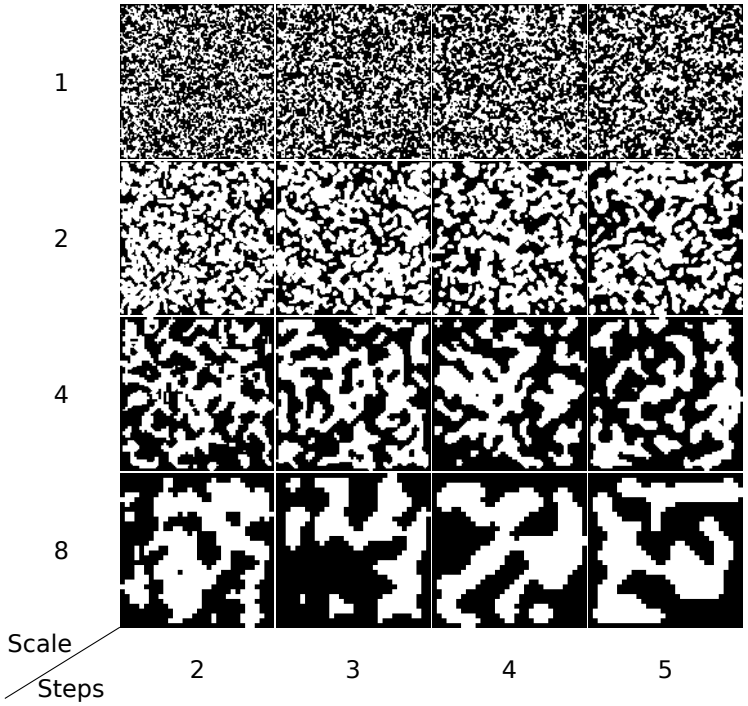

Fig. 2: We use *Cellural Automata* to generate the mask of an image. We apply a median filter at each step of the automaton to change its states. We start with a down-scaled mask size and re-scale after the automaton has reached its final state to create *islands* of different sizes.

to the entire image. Moreover, the masked regions can have a variety of shapes and sizes calling for an intricate approach to the problem. The objective of this challenge is to stimulate and propose a benchmark for further research in this direction.

This challenge is one of the AIM 2020 associated challenges on: scene relighting and illumination estimation [7], image extreme inpainting [23], learned image signal processing pipeline [13], rendering realistic bokeh [14], real image super-resolution [28], efficient super-resolution [35], video temporal super-resolution [26] and video extreme super-resolution [9].

2 AIM 2020 Challenge

The goals of the AIM 2020 Challenge on Image Extreme Inpainting are: *(i)* to advance towards more challenging conditions for image inpainting methods, *(ii)* to provide a common benchmark, protocol and dataset for image inpainting

*Participant reported runtime in CPU time.

Table 1: Technical details of the participants. All teams that participated in Challenge 2 also participated in Challenge 1, and the runtime of using the semantic information is under mild assumptions the same

Team	Runtime (s/img)	GPU	Framework
Rainbow	0.2	TITAN RTX	Pytorch
Yonsei-MVPLab	1.58	RTX 2080Ti	Pytorch
BossGao	10	Tesla V100	Tensorflow
ArtIst	0.32	GTX 1080Ti	Tensorflow
DLUT	36.39*	Tesla V100	Pytorch
AI-Inpainting	3.54	RTX 2080Ti	Pytorch
qwq	2.5	GTX 1080Ti	Pytorch
CVIP Inpainting	0.57	RTX 2080Ti	Pytorch
DeepInpaintingT1	6.4	RTX 2080Ti	Pytorch
IPCV_IITM	1.2	Titan X	Tensorflow
MultiCog	0.44	K40	Keras

Table 2: Challenge results for Track 1: Source domain on the final test set

Team	FID↓	LPIPS↓	PSNR↑	SSIM↑	MAE↑
Rainbow	30.69	0.10 ± 0.07	26.71 ± 7.43	0.88 ± 0.09	0.03 ± 0.02
Yonsei-MVPLab	30.71	0.11 ± 0.09	27.25 ± 8.09	0.89 ± 0.09	0.02 ± 0.02
BossGao	31.23	0.11 ± 0.08	26.59 ± 8.37	0.88 ± 0.1	0.03 ± 0.02
ArtIst	33.29	0.12 ± 0.08	26.64 ± 8.55	0.87 ± 0.1	0.03 ± 0.02
DLUT	40.46	0.13 ± 0.09	26.15 ± 8.47	0.87 ± 0.1	0.03 ± 0.02
AiriaBeijingTeam	40.63	0.13 ± 0.08	26.1 ± 6.91	0.87 ± 0.09	0.03 ± 0.02
qwq	41.03	0.19 ± 0.13	25.95 ± 5.86	0.86 ± 0.11	0.03 ± 0.02
CVIP Inpainting Team	44.29	0.16 ± 0.11	26.2 ± 7.59	0.87 ± 0.1	0.03 ± 0.02
DeepInpaintingT1	48.40	0.18 ± 0.11	26.64 ± 7.6	0.87 ± 0.09	0.03 ± 0.02
IPCV_IITM	93.95	0.30 ± 0.27	20.98 ± 7.61	0.68 ± 0.29	0.08 ± 0.08
MultiCog	117.52	0.53 ± 0.18	17.58 ± 2.53	0.62 ± 0.14	0.09 ± 0.04

methods, and *(iii)* to establish current state-of-the-art under extreme conditions. The aim is to obtain a network design / solution capable of producing high quality results with the best perceptual quality and similarity to the reference ground truth.

2.1 Description

In the classical sense, the task of image inpainting aims to reconstruct *damaged* or *missing* areas of an image. The ideal reconstruction provided by an inpainting operation should recreate the original pixels. However, in many cases all the information regarding a present entity in an image, *i.e.* an object, is vanished behind the missing pixels. It is impossible for a model to perfectly reconstruct the missing information without external guidance. Motivated by this problem we created two tracks for the AIM 2020 Extreme Inpainting Challenge: (1) the

Table 3: Challenge results for Track 2: Target domain on the final test set

Team	FID↓	LPIPS↓	PSNR↑	SSIM↑	MAE↑
Rainbow	32.60	0.11 ± 0.08	26.77 ± 7.82	0.88 ± 0.1	0.03 ± 0.02
ArtIst	36.00	0.13 ± 0.08	27.11 ± 7.93	0.88 ± 0.09	0.03 ± 0.02
DLUT	43.22	0.14 ± 0.10	25.8 ± 8.1	0.86 ± 0.1	0.03 ± 0.03
qwq	43.38	0.19 ± 0.12	24.74 ± 5.43	0.85 ± 0.1	0.03 ± 0.03
DeepInpaintingT1	44.57	0.18 ± 0.11	26.94 ± 7.09	0.88 ± 0.09	0.03 ± 0.02
AI-Inpainting	45.63	0.15 ± 0.10	25.79 ± 6.68	0.86 ± 0.1	0.03 ± 0.02

Table 4: Results per mask type for three top solutions of Track 1

Team	Mask Type	LPIPS↓	PSNR↑	SSIM↑	MAE↑
Rainbow	Box	0.15	22.49	0.82	0.04
	Cellural Automata	0.12	25.72	0.89	0.03
	Free-Form	0.05	32.44	0.95	0.01
Yonsei-MVPLab	Box	0.17	23.45	0.83	0.03
	Cellural Automata	0.13	25.40	0.88	0.03
	Free-Form	0.04	33.44	0.95	0.01
BossGao	Box	0.15	22.09	0.81	0.04
	Cellural Automata	0.13	25.31	0.87	0.03
	Free-Form	0.05	32.94	0.95	0.01
Masked Images	Box	0.30	11.47	0.66	0.14
	Cellural Automata	0.66	8.21	0.23	0.23
	Free-Form	0.25	16.02	0.70	0.07

classical image inpainting track, where no additional information is used, and (2) the semantically guide image inpainting track, where the pixel-level semantic labels of the whole image are provided to drive the generation of the missing pixels.

2.2 Dataset

We use a partition of ADE20k dataset [37] for both tracks of the challenge. The ADE20k is a dataset with a large diversity of contents. A public training and validation set is provided, with each image being fully annotated. The resolution of the images provided in the dataset is highly diverse, and for our challenge, the inpainting must be achieved in the full scale of the image.

We aimed to provide images for both tasks that are drawn from the same distribution. Moreover, for the semantically guided challenge, we decided not to include all the classes and bypass the problem of the distribution's long tail. To achieve this, we selected the union of (1) the most occurring semantic classes per image and (2) the most occurring semantic classes per pixel. Ultimately, we ended up with 51 semantic classes and we filtered the images so that at least 90% of the pixels in a particular image belong to these 51 semantic classes. The resulting training set was 10,330 to be used for both tracks.

Fig. 3: Qualitative comparison over the top three methods in Track 1. Rainbow and Yonsei-MVPlab team models can successfully interpret the local scene textures and propagate information to the missing regions when the inpainted region is not too extreme. Overall, the results from the rainbow team has fewer visual artifacts. Zoom in for better visual comparisons.

In order to create the validation and test set of the challenge, we followed the same procedure as in the training set. We processed the validation set of ADE20k and divided the filtered images into validation and test subset. Each set was divided into three equal numbered parts: (1) images used solely for track 1, (2) images used solely for track 2, and (3) images used for both tracks.

We enriched our test set by mining pictures from the COCO-Stuff dataset [5]. We manually picked 200 images depicting scenes similar to the ones produced by our filtering process on ADE20k. Then, we defined a matching between the class labels of ADE20k and COCO-Stuff and translated the semantic maps accordingly. The additional COCO-Stuff images were divided into three groups similar to ADE20k images.

2.3 Masking the images

In order to push the boundaries of image inpainting, for this competition we are using three different types of masks: (*i*) Rectangular masks with width and height between 30-70% of each dimension, (*ii*) brush strokes randomly drawn [33], and (*iii*) our own method generated masks based on cellural automata.

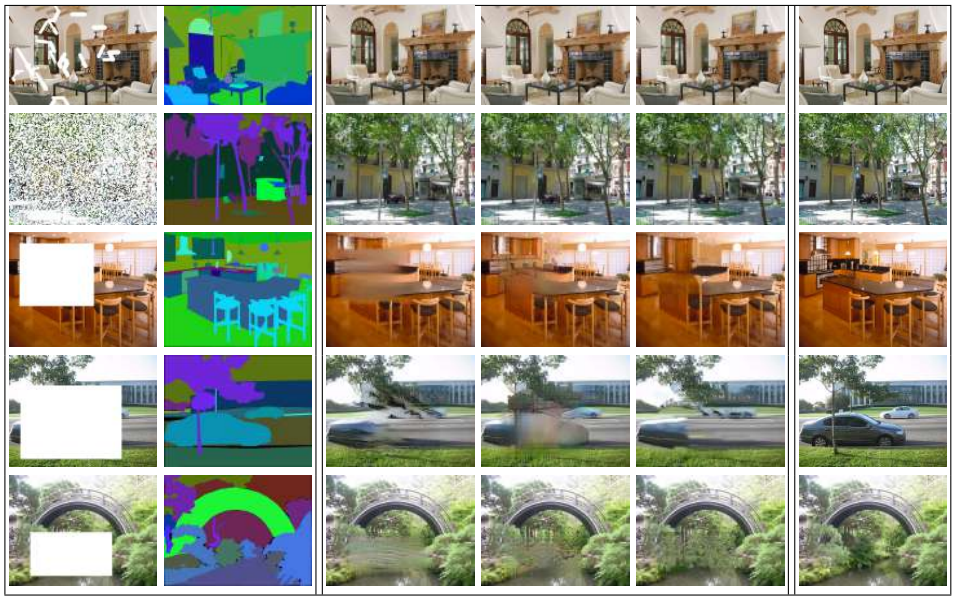

Fig. 4: Qualitative comparison over the top three methods in Track 2. Semantic labels help the model from ArtIst to better locate object boundaries. Although rainbow team yields better perceptual performance, it is unable to assign consistent object boundaries. We depict the same images as in Fig. 3 for comparison in both tracks.

We propose this novel way of masking in order to create a distorted image that has holes distributed across its dimensions, while forming small *islands* of missing pixels that are less trivial to fill compared to just salt and pepper noise. Cellural automata have been utilized in automatic content generation for games [25], but to our knowledge, this is the first time used for mask generation for image inpainting.

Our cellural automaton mask is a 2-dimensional grid of pixels having two states: either masked or not. We initialize our grid by randomly assigning a state to each pixel. At each step, every pixel's state is changed based on a majority vote on its neighborhood. We are using a Moor neighborhood: a 3-by-3 square, essentially applying a median filter. To facilitate creating islands of various sizes, we downscale the size of the mask before calculating the states of the automaton and re-scaling to the original size of the image to be masked. When re-scaling, we use the morphological operation of dilation to produce smoother edges. The resulted masks can be found in Fig. 2. To generate each mask, we randomly choose between down-scaling 1,2,4 or 8 times, and the number of applied steps (2-5).

Table 5: Results per mask type for three top solutions of Track 2

Team	Mask Type	LPIPS↓	PSNR↑	SSIM↑	MAE↑
Rainbow	Box	0.16	22.15	0.81	0.04
	Cellural Automata	0.10	26.85	0.90	0.02
	Free-Form	0.05	32.84	0.95	0.01
ArtIst	Box	0.16	23.07	0.83	0.03
	Cellural Automata	0.16	25.99	0.88	0.03
	Free-Form	0.06	33.08	0.95	0.01
DLUT	Box	0.16	21.70	0.81	0.04
	Cellural Automata	0.18	24.49	0.85	0.03
	Free-Form	0.07	32.06	0.94	0.02
Masked Images	Box	0.32	11.16	0.63	0.14
	Cellural Automata	0.65	8.77	0.26	0.22
	Free-Form	0.27	15.61	0.69	0.09

2.4 Tracks

Track 1: Image Extreme Inpainting In this track, we provide only the degraded image with the corresponding mask to fill in the missing pixels. Formally, the aim of this track is to learn a mapping function $\mathbb{G}$ in order to produce a coherent and perceptually looking image $X = \mathbb{G}(X_m, M)$, where X_m and M are the degraded image and the mask image, respectively. See Fig. 1 upper row for a random example of inputs and ground-truth. For this track, only the images are to be used. The use of any other additional information (semantics, object instances) was explicitly not allowed.

Track 2: Image Extreme Inpainting guided by pixel-wise semantic labels Here the task is also to complete the missing pixel information, but using a semantic image as guidance, so the generated image should resemble the same objects and shapes as in the semantic region. It can be formally depicted as $X = \mathbb{G}(X_m, M, S)$, where S is the semantic map annotation corresponding to the ground-truth X. We depict an example of input images in Fig. 1 bottom row. For the semantic images, we provided a subset of the original ADE20k's semantic classes list, and the new number of classes is 51. The use of any other additional information, *e.g.* object/instance information, was not allowed.

2.5 Challenge Phases

The challenge had three phases: (1) *Development phase*: the participants had access to training inputs and ground-truth, and the input images of the validation set. During this phase, the participants were free to select the number and type of masks described in Section 2.3. (2) Validation phase: the participants had the opportunity to compute validation performance using PSNR and SSIM metrics by submitting their results on the server. A validation leaderboard was also available. (3) Final test phase: the participants got access to the test images and

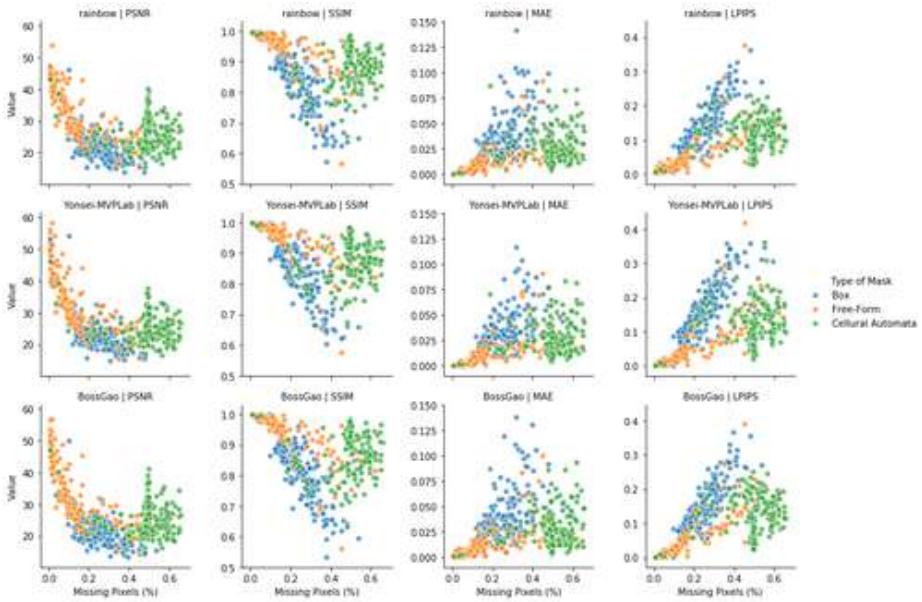

Fig. 5: A visualization of the performance of the top three methods for Track 1 in relation to the percentage of the missing pixels in the input image. Each individual point corresponds to an image of the test set. For PSNR and SSIM larger values are better while for MAE and LPIPS smaller values indicate better performance. For all methods we observe that while *cellural automata* usually remove more pixels than the *box* masks, the latter are difficult to inpaint and consistenly produce worse results for all metrics. The easiest type of mask to inpaint is the Free-Form mask. The missing pixels are less in number and not concentrated as in the *box* case

had to submit their inpainted images along with the description of the method, code, and model weights for further reproducibility.

3 Challenge Results

For more than 150 participants registered on the two tracks combined, 11 teams entered the final phase of Track 1 and submitted results, code/executables, and factsheets. In Track 2, 6 teams entered the final phase. Track 2 participants also participated in Track 1. The Methods of the teams that entered the final phase are described in Section 4, the technical details of the teams in Table 1, and the team's affiliation is shown in Section 4.11. Table 2 and 3 report the final results of Track 1 and 2 respectively, on the test data of the challenge. Noteworthy, for the top three methods in terms of both perceptual and fidelity metrics, we run a perceptual study to determine the winner of the challenge. Based on our results,

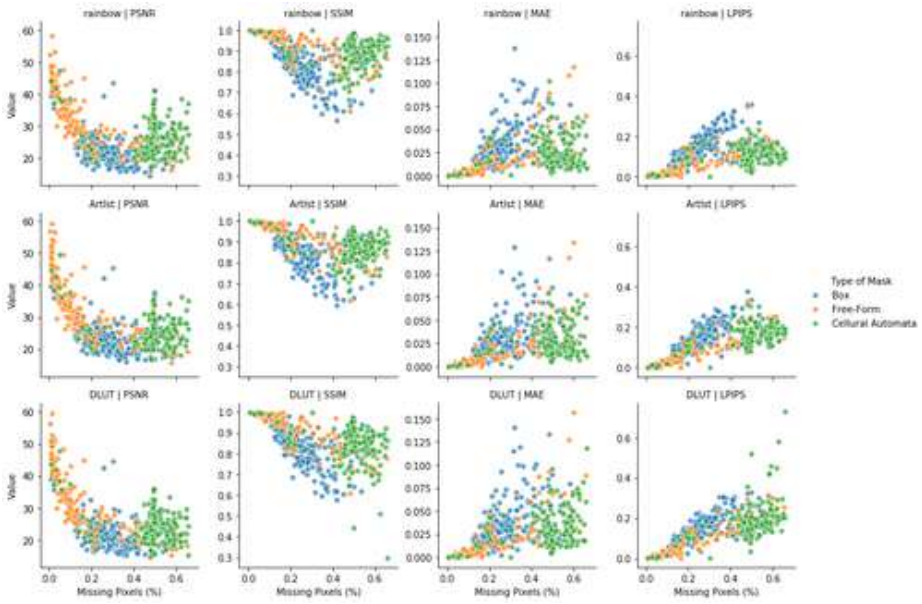

Fig. 6: A visualization of the performance of the top three methods for Track 2 in relation to the percentage of the missing pixels in the input image. Each individual point corresponds to an image of the test set. For PSNR and SSIM larger values are better while for MAE and LPIPS smaller values indicate better performance. We make similar observations with Track 1, as described in Fig. 5

Rainbow generated in overall more photo-realistic inpainted images than Yonsei-MVPLab and BossGao in Track 1 and than ArtIst and DLUT in Track 2, hence ranking 1st in both tracks of the first AIM Extreme Inpainting Challenge. We have to note that for Track 1 Yonsei-MVPLab performed better in the fidelity metrics.

3.1 Architectures and main ideas

All the proposed methods are GAN-based solutions. In most cases, a variant of PatchGAN [15] is used for the discriminator, while many generators imitate the structure of recent state-of-the-art architectures with two Stage Architectures, one coarsely inpainting, and the other taking care of the finer details [32,33]. The Edge-Connect approach [21] was also explored. Interestingly, the work of Hui *et al.* [12], the team winning this competition, was also employed by the second-best team of Track 1, Yonsei-MVPLab. The teams that performed the best in the competition propose novel components to their architectures.

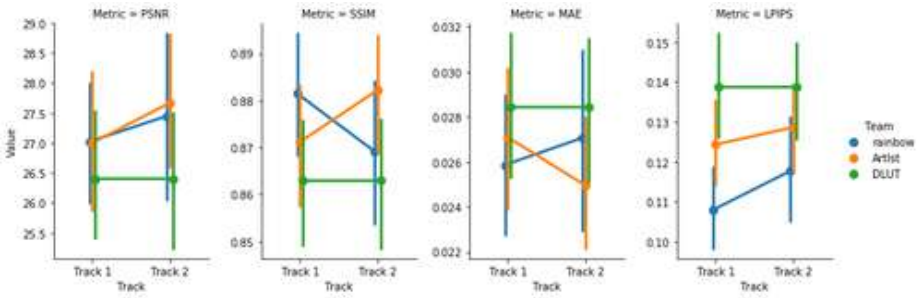

Fig. 7: Point plot of the performance of the top three teams that participated in both tracks, calculated in a subset of the test set shared between them. *DLUT* submitted the same model for both tracks, without utilizing the additional information. We observe that the top two teams produce worse perceptual results (LPIPS) when using semantics. For the second track, the first team is worse for the majority of metrics while the second team is better compared to the results of the first track, indicating that the semantic information can be both beneficial and detrimental to the inpainting task, based on how it is integrated into the network.

3.2 Handling of extreme image sizes

Many of the participants reported that their solution could not handle high-resolution images without prompting an *our of memory* error (OOM). Two approaches were mainly reported to tackle this issue. To circumvent this issue, many contestants downscaled the input images prior to inpainting and up-scaled the results with either a classical method, *e.g.* bilinear upsampling, or using a Super Resolution network. This solution was only used by some teams to handle the big box holes. Alternatively, the input image was divided into patches, which after independent processing were stitched together to produce the final output.

3.3 Handling of different mask types

The three different mask types impose different problems and a few of the teams decided to handle them independently. We conducted an analysis on the results of the top three performing teams for each track, which can be found in Table 4 and 5 respectively. As a comparison baseline, we show the metrics results when computed against the input image with holes. For the baseline case, the cellural automata masks remove the most pixels and thus produce the worst results, yet we can see that compared to the massive chunks of information removed from the box masks, cellural automata masks are easier to inpaint. In Fig. 5 and 6 we can observe how the top three teams of each track performed relative to the percentage of missing information.

3.4 The effect of the semantic guidance

In order to be able to fairly compare the results of the two tracks, a subset of their test set was shared. In Fig. 7 we can see how the performance on a variety of metrics differs between the classical inpainting case, where no additional information is used, and the semantically guided case. The results indicate that semantic information can both increase and decrease the performance on the inpainting task, based on how its processing was implemented in the network. Some teams use the same network they used for the first track to produce results for the second one while other trivially incorporate the semantic information in their pipeline. These factors may contribute to counter-intuitive result that many teams produced worse results in the second track.

3.5 Conclusions

The AIM 2020 Challenge on Image Extreme Inpainting gauges the current solutions and proposes a benchmark for the problem of image extreme inpainting. Different methodologies and architectures have been proposed to tackle this problem. Most of these solutions built upon traditional image inpainting methods, which has been qualitative and quantitatively validated in this report that does not necessarily hold for the extreme case. Using semantic information as guidance does not always increase the performance of an image inpainting. We believe that AIM 2020 Challenge on Image Extreme Inpainting could boost research in this challenging direction.

As qualitative and quantitative results show, image extreme inpainting is an unexplored area, where state-of-the-art classical image inpainting methods fail to generalize. Remarkably, the number of missing pixels is not a critical factor for image inpainting, but the type of mask type.

4 Challenge Methods and Teams

4.1 Rainbow

For both task 1 and task 2, Rainbow proposes a one-stage model (see Figure 8) modified from Hui *et al.* [12], which utilizes dense combinations of dilated convolutions to obtain larger and more effective receptive fields. To better train the generator, and in addition to the standard VGG feature matching loss, they design a novel self-guided regression loss to dynamically correct low-level features of VGG guided by the current pixel-wise discrepancy map. Besides, they devise a geometrical alignment constraint item to penalize the coordinate center of estimated image high-level features away from the ground-truth. Moreover, for track 2, they add SPADE ResBlock [24] in the decoder to introduce a semantic map for semantic guided image inpainting task.

With self-guided regression loss, geometrical alignment constraint, VGG feature matching loss, discriminator feature matching loss, realness adversarial

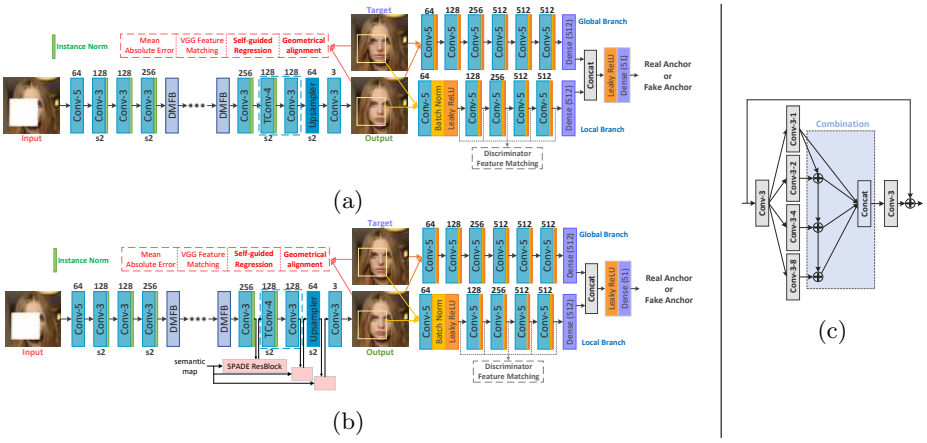

Fig. 8: Overview of Rainbow method. (a) The framework of the solution for classic image inpainting. (b) The framework of the solution for semantic guided image inpainting. (c) The structure of dense multi-scale fusion block (DMFB), "Conv-3-8" indicates 3×3 convolution layer with the dilation rate of 8.

loss [29], and mean absolute error (MAE) loss, our overall loss function is defined as

$$\mathcal{L}_{total} = \mathcal{L}_{mae} + \lambda \left(\mathcal{L}_{self-guided} + \mathcal{L}_{fm\_vgg}\right) + \eta \mathcal{L}_{fm\_dis} + \mu \mathcal{L}_{real\_adv} + \gamma \mathcal{L}_{align}. \quad (1)$$

Except for $\mathcal{L}_{real\_adv}$, the detailed description of other losses can be found in [12]. Different from the standard GAN, RealnessGAN's discriminator outputs a distribution as the measure of realness.

$$\mathcal{L}_{real\_adv} = \mathbb{E}_{z \sim p_z}\left[\mathcal{D}_{KL}\left(\mathcal{A}_1 \| D\left(G\left(z\right)\right)\right)\right] - \mathbb{E}_{z \sim p_z}\left[\mathcal{D}_{KL}\left(\mathcal{A}_0 \| D\left(G\left(z\right)\right)\right)\right], \quad (2)$$

where $\mathcal{D}_{KL}\left(\cdot \| \cdot\right)$ denotes Kullback-Leibler (KL) divergence. Two virtual ground-truth distributions are needed to stand for the realness distributions of real and fakes images. They refer to these two distributions as $\mathcal{A}_1$ (real) and $\mathcal{A}_0$ (fake). Concretely, $\mathcal{A}_1$ and $\mathcal{A}_0$ are chosen to resemble the shapes of a Gaussian distribution $\mathcal{N}\left(\mathbf{0}, \mathbf{I}\right)$ and a Uniform distribution $\mathcal{U}\left(\mathbf{0}, \mathbf{I}\right)$, respectively.

It is set $\lambda = 10$, $\eta = 5$, and $\gamma = 1$ in Equation 1. The training procedure is optimized using Adam optimizer with $\beta_1 = 0.5$ and $\beta_2 = 0.9$. Learning rate $2e{-}4$, and batch size is 16. The training patches are 256×256. For training, given a raw image $\mathbf{I}_{gt}$, a binary image mask $\mathbf{M}$ (value 0 for known pixels and 1 denotes unknown ones). In this way, the input image $\mathbf{I}_{in}$ is obtained from the raw image as $\mathbf{I}_{in} = \mathbf{I}_{gt} \odot (1 - \mathbf{M})$. Our inpainting generator takes $[\mathbf{I}_{in}, \mathbf{M}]$ as input, and produces prediction $\mathbf{I}_{pred}$. The final output image is $\mathbf{I}_{out} = \mathbf{I}_{in} + \mathbf{I}_{pred} \odot \mathbf{M}$. All input and output are linearly scaled to $[-1, 1]$. For arbitrary masks, the random regular regions are cropped and set to the local discriminator.

During the inference stage, our network cannot directly process input ultra-high-resolution images (up to 3872×2592px) with regular masks. When convo-

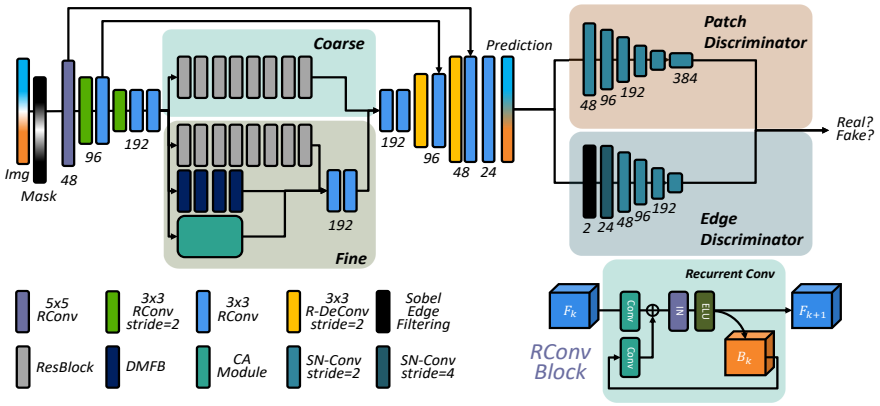

Fig. 9: Overall network structures of Yonsei-MVPLab method. Basic building block is Recurrent Convolution (RConv) that is depicted in the lower-right side.

lution operation is performed on masked regions, it cannot utilize features on known areas due to the large masked patches and the limited receptive field of the proposed generator. Thus, they downscale the input images and then generate low-resolution painted results. After that, bicubic interpolation is employed to produce high-resolution images. For irregular masks, it can regard as a low-level vision task. Because of the complete image with ultra-high-resolution, they crop it into four overlapped patches and then send them to the generator sequentially for solving the issue of OOM (out of memory).

4.2 Yonsei-MVPLab

This team proposes three novel components for the task of image inpainting: Recurrent Convolution (RConv), Edge Discriminator, and Frequency Guidance Loss. The overall structure is shown in Fig. 9. RConv and R-DeConv indicate Recurrent Convolution and Recurrent Deconvolution, respectively. DMFB and CA Module indicate Multi-scale Fusion Block in [12], and contextual attention module in [32], respectively. They follow the coarse-to-fine two-stage scheme as in similar inpainting algorithms[32,33]. For the coarse route, residual blocks with dilated convolutions used in EdgeConnect [21] are stacked. For the fine route, the same residual blocks and additional parallel blocks: DMFB [12] and CA [32] modules are used.

Like GatedConv [33], Recurrent Convolution is a basic building block for every layer. The recurrent Convolution block is depicted in the lower-right side of Fig. 9. It saves the feature of a coarse route into memory and fusion with the feature from the fine route. It effectively delivers gradients from the last output side of the network to the very first coarse layers via memory. For the discriminator, they modify the patch-based discriminator used in GatedConv [33]. Therefore, a novel discriminator, called Edge Discriminator, is applied after the Sobel edge

filter. Sobel edge filter is implemented as convolution without gradient update that extracts horizontal and vertical edges. These edges are passed into the edge discriminator. It is much simple and not depend on the results of the predicted edges as in [21]. For our novel Frequency Guidance Losses, which is inspired by [8], They filter the coarse output using a low-pass filter and the fine output using a high-pass filter. Then compare them with the same filtered ground truth images. For the coarse route, an L1 loss is applied to the low-pass filtered images, and for the fine route, an L1 loss is applied to the high-pass filtered images only in the hole region.

For some large holes, a selective up-scaling network is used for these images. These images are selected via convolution onto masks. For the up-scaling network, RRDBs in ESRGAN [27] with 48 channels are used after the main network. The hinge loss [19] is employed as the adversarial loss to train the discriminators. For the generator, a total of eight losses are used. For the coarse rouse: l1 loss, adversarial loss, and perceptual loss [16]. For the fine route: style loss, feature matching loss, and two frequency guidance losses. Style loss and feature matching loss are borrowed from those of EdgeConnect [21]. The system is pretrained on Places dataset [36], and then fine-tune on the ADE20K [37] dataset. Training patch size is 256×256 (480×480 for up-scaling network). The learning rate for discriminator is 0.0004, and the learning rate for the generator and up-scaling network is 0.0001. The learning rate is decayed every 25 epochs by half. The entire system is trained for 100 epochs.

4.3 BossGao

In general, BossGao has designed two-stage solutions for extreme image inpainting, as shown in Fig. 10. The first stage tries to recover coarse inpainting results which are relatively smooth for missing regions while keeping the content as semantically similar as possible. Our stage 2 uses the extract framework of DeepFill_v2 [33], where gated convolution is designed to assigning soft attention scales to each feature point for adaptively modulation of different regions whose different shapes of valid points. In addition, contextual attention is adopted to leverage valid regions for better hallucination. In this challenge, our proposed image inpainting solution exhibits mask awareness in the following aspects: 1) they develop different networks to handle block mask, brush mask and cellural automata mask, respectively. 2) To handle extreme large block masks, they proposed a novel mask aware dynamic filtering (MADF) coarse inpainting network as stage 1. In MADF, filters for each convolution window are generated from features of the corresponding region of the mask. Furthermore, it is a 7-level downsample-upsample UNet architecture and is capable to generate relatively better coarse results than the coarse inpainting network of DeepFill_v2 because of its larger receptive field. With this solution, block masks can be filled in with much more visually plausible results, especially when the image is of very high resolution and the block mask is relatively large.

The details of stage 1 coarse network for box masks are as follows. The encoder E generates multiple level feature maps. Let denote the finest level

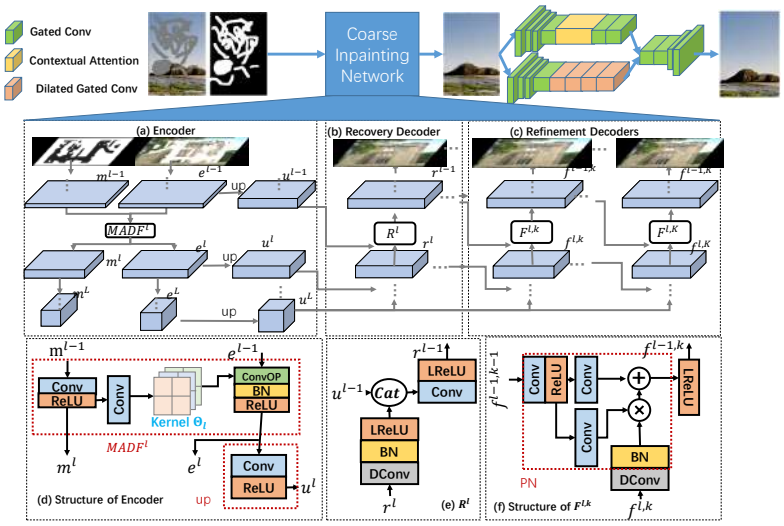

Fig. 10: Two stage framework overview, Image Inpainting With Mask Awareness. Stage 1 fills missing regions coarsely and stage 2 refines the results. In this implementation, they use the exact stage 2 network configuration as DeepFill_v2 [33]. The architecture of the stage 1 network used for block mask is shown in the bottom. $MADF^l$ and R^l are the MADF module and recovery decoder block at l-th level respectively. $F^{l,k}$ represents the k-th refinement decoder block at l-th level. m^l is the l-th mask feature map of the encoder. The convolution operation marked in green in the encoder takes e^{l-1} as input and its kernel for each convolution window is generated from the corresponding region of m^{l-1}.

feature map as u^1 and the coarsest feature map as u^L, and m^l and e^l are the corresponding feature maps of mask and input generated by the encoder at level l. Let assume a $k \times k$ convolution with stride s will be applied to $e^{l-1} \in R^{H \times W \times C_e^{l-1}}$, and there are $N_H \times N_W$ convolution windows in total. The mask-aware dynamic convolution layer marked in green operates as follows: on $m^{l-1} \in R^{H \times W \times C_m^{l-1}}$, $k \times k$ convolution with stride s and ReLU activation is firstly applied to generate $m^l \in R^{N_H \times N_W \times C_m^l}$, then they utilize 1×1 convolution to generate the kernel tensor $\Theta_l \in R^{N_H \times N_W \times D}$, where D equals $C_e^{l-1} \times k \times k \times C_e^l$. Finally, each of all the $N_H \times N_W$ windows in e^{l-1} is convolved using the kernel reshaped from the corresponding point of Θ_l, *i.e.*, the convolution kernel of the $[n_H, n_W]$-th window in e^{l-1} is reshaped from $\Theta_l[n_H, n_W, :]$. To reduce computational overhead, they choose to apply MADF to extract relatively low dimensional latent features and then map it to a higher dimension. Specifically, C_m^l is set to a relatively small value of 16 and output channel number C_e^l is limited by $\min(128, 16 * 2^l)$ for any l. Then they increase the channels of each e^l by 1×1 convolution with ReLU to produce u^l. Multiple cascaded decoders are used to refine the output step by step.

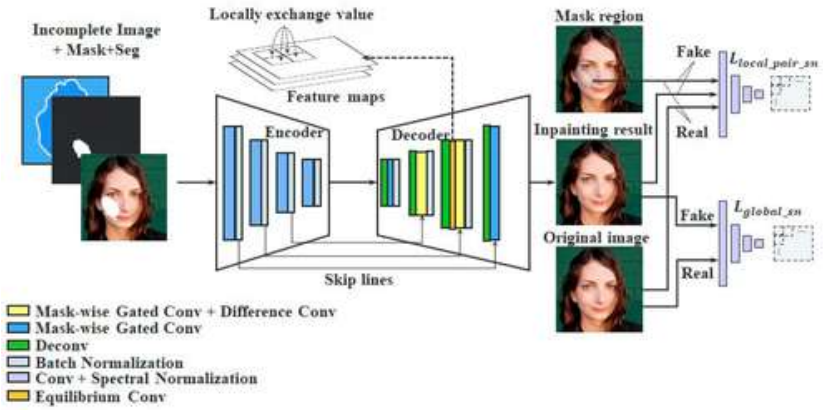

Fig. 11: Overall architecture of the ArtIst approach.

4.4 ArtIst

This solution proposes a one-stage inpainting solution [3] based on a single UNet architecture consisting of 6 different kinds of layers (operations), where the main contribution is different convolution-based blocks. First, Equilibrium Convolution (EConv) is designed to reduce undesired artifacts in the output image. Second, Mask-wise Convolutions (MGConv) are deployed to reduce the parameters in gated convolutions [33]. Lastly, in order to deal with the large mask challenge, they create a Differentiate Convolution block. A schematic of the architecture is depicted in Fig. 11.

4.5 DLUT

This method is mainly based on the inpainting system proposed by Zeng *et al*. [34]. It is an iterative inpainting model with a guided upsampling module as shown in Fig. 12. Due to the memory limitation, for high-resolution images whose the long side is greater than a threshold, they execute the inpainting system of Zeng *et al*. [34] on 2× downsampled input and then use an external super-resolution model [1] to get the inpainted images of the original size.

4.6 AI-Inpainting

The solution is based on Edge-Connect [20], which is mainly focused on small size images. The entire network consists of two sub-modules: an edge part and an in-painting part (see Figure 13). The edge module uses the masked image to generate the edge of the entire image, and the edge together with the masked image are fed into the inpainting module in order to generate the final result.

https://github.com/tensorlayer/srgan.git

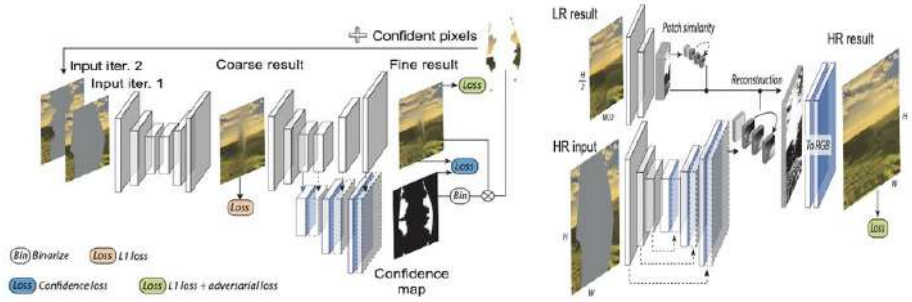

Fig. 12: Overview of DLUT method.

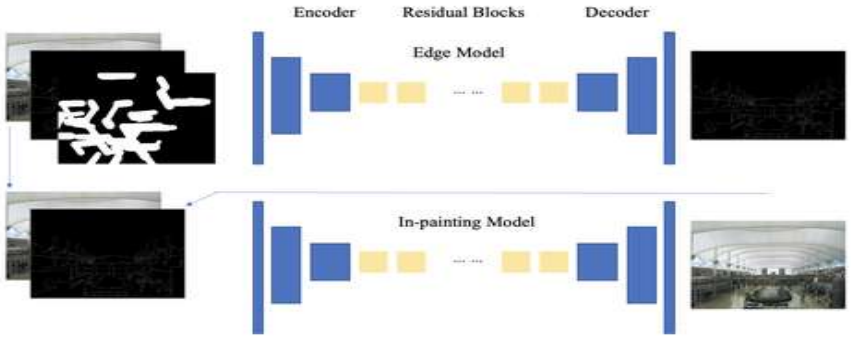

Fig. 13: Overview of AiriaBeijing method. The first module predicts semantic edges, the second uses these edges to estimate the results (zoom-in for better details).

This system categorizes images into three different groups according to their mask type. For box masks, the image will be resized to three smaller sizes in order to be filled up with details at different aspects, and thus, be able to run the model without running out of memory. After resizing back to original size, they merge three images with different weight into one image. For large images with other masks, they cut each image into many patches and fed them into the model. After all, they put the output images back to original position. This method might cause some color deviation between masks since they do not use global information of the whole image.

4.7 qwq

The proposed solution adopts a AFN network [30] as generator, and a Markovian Discriminator [18] to guide the inpainting process. As shown in Fig. 14, the generator consists of several fractally stacked Attentive Fractal Blocks, which

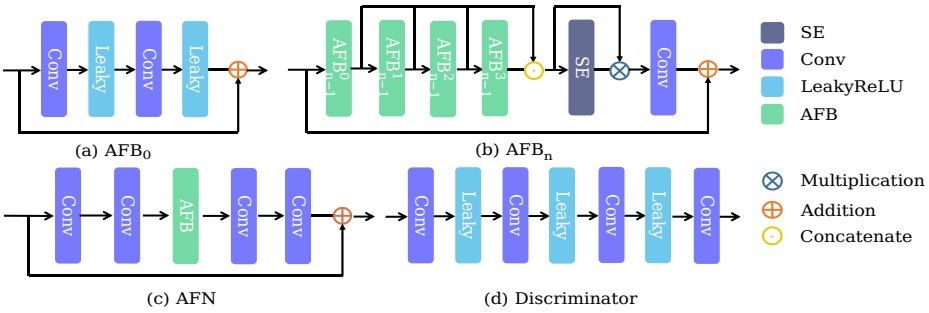

Fig. 14: Architecture of the qwq proposed framework. (a) shows the structure of the basic block AFB_0. (b) shows the hierarchical architecture of AFB_n, where $n \in [1, r]$ is the recursion level of the block. (c) illustrates the structure of AFN, which consists of encoding layers, an AFB_r, and decoding layers from left to right. (d) shows the structure of our discriminator.

are constructed via progressive feature fusion and channel-wise attention guidance. Shortcuts and residual connections at different scales effectively resolve the vanishing gradients and help the network to learn more key features. The progressive fusion of intermediate features lets the network handle rich information. The discriminator is constructed via stacking several convolution and leaky ReLU layers. The output of the discriminator is designed to be a one-channel confidence map, which penalizes the masked area.

4.8 CVIP Inpainting Team

Following the work of Yu *et al.* [32], this team proposes a model consisting of a two-stage system, namely a coarse network, and a refinement network. Fig. 15 shows the proposed solution. In addition to a regular branch, the coarse network uses an attention branch with a novel attention module namely the Global Spatial-Channel Attention (GSCA) module, which can capture the structural consistency by calculating global correlation among both spatial and channel features at the global level. For the refinement network, in addition to GSCA, they adopt a recurrent residual attention U-net (RR2U) [2] architecture with major modifications in the architectural level. The recurrent residual attention consists of a multi-stage GRUs [6] that computes the inter-layer feature dependencies.

4.9 DeepInpaintingT1

The proposed model [17] consists of two generators and two discriminators. The overall proposed architecture is displayed in Fig. 16. The first coarse generator G_1 at Coarse Reconstruction Stage (the top left corner) and the second refinement generator G_2 at the Refinement Stage (bottom) constitute the Deep

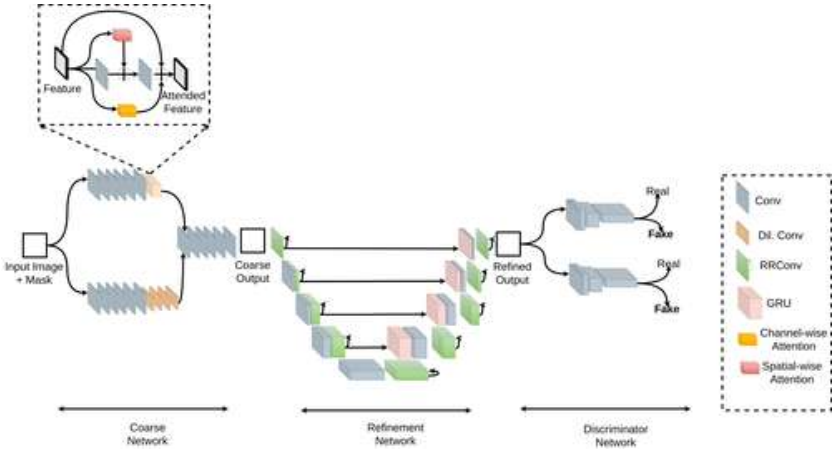

Fig. 15: Overall architecture of the CVIP approach. The model has two stages, namely a coarse network and a refinement network.

Generative Inpainting Network (DeepGIN) which is used in both training and testing. The two discriminators D_1 and D_2 located within Conditional Multi-Scale Discriminators area (the top right corner) are only used in training as an auxiliary network for generative adversarial training. The generator G_1 is trained to roughly reconstruct the missing regions and gives $\mathbf{I}_{coarse}$. The generator G_2 is trained to exquisitely decorate the coarse prediction with details and textures, and eventually forms the completed image $\mathbf{I}_{out}$. Three main ideas are proposed to accomplish the task of inpainting: *i*) Spatial Pyramid Dilation ResNet blocks (yellow and green blocks) with various dilation rates to enlarge the receptive fields such that information given by distant spatial locations can be used for both reconstruction and refinement; *ii*) Multi-Scale Self Attention (orange blocks) to enhance the coherency of the completed image by attending on the self-similarity of the image itself at three different scales; *iii*) Back Projection strategy (the bottom right shaded area) to encourage better alignment of the generated patterns and the reference ground truth. For the two discriminators (the top right corner), they operate at two input scales, 256×256 and 128×128 respectively, to encourage better details and textures of the locally generated patterns at different scales.

4.10 IPCV_IITM

Team IPCV_IITM proposes a method based on Contextual Residual Aggregation (CRA) network [31]. Although recent deep learning-based in-painting methods are more effective than classic approaches, however, due to memory limitations they can only handle low-resolution inputs, typically smaller than 1K. Meanwhile, the resolution of photos captured with mobile devices increases up to 8K. Naive up-sampling of the low-resolution inpainted result can merely yield a large

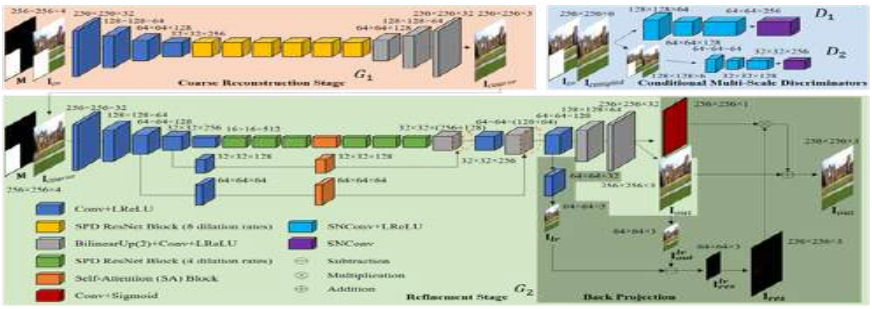

Fig. 16: Overview of DeepInpaintingT1 for extreme image inpainting.

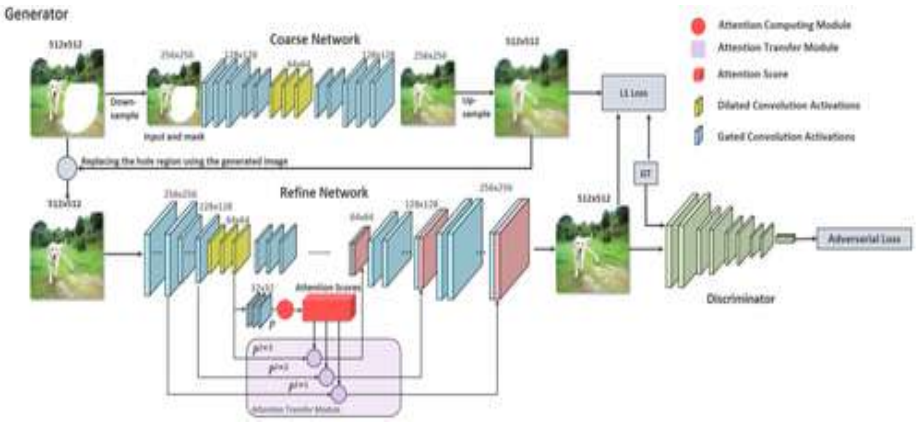

Fig. 17: Overview of IPCV_IITM method. Two models run in parallel to generate results that is evaluated using a discriminator module.

yet blurry result. Whereas, adding a high-frequency residual image onto the large blurry image can generate a sharp result, rich in details and textures. Motivated by this, CRA mechanism is employed that can produce high-frequency residuals for missing contents by weighted aggregating residuals from contextual patches, thus only requiring a low-resolution prediction from the network. Fig. 17 shows the network design proposed by this team. Since convolutional layers of the neural network only need to operate on low-resolution inputs and outputs, the cost of memory and computing power is thus well suppressed.

4.11 MultiCog

MultiCog uses a Pix2Pix Generative Adversarial Network (GAN) as its core [15]. The main idea here is selecting the two domains as unpainted images and painted images, where the learned mapping function is unpainted to painted. See schematic in Fig. 18.

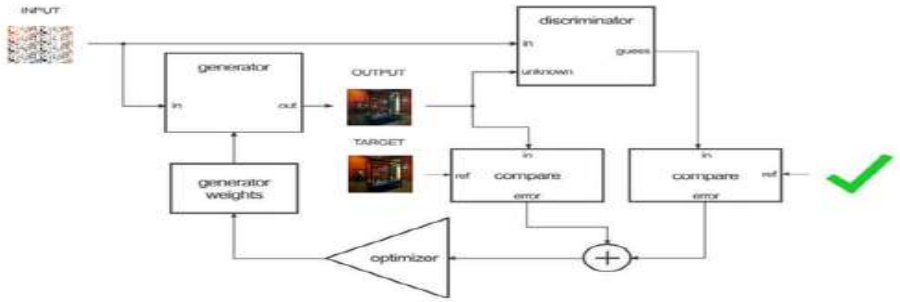

Fig. 18: Overview of MultiCog.

Acknowledgements

We thank the AIM 2020 sponsors: Huawei, MediaTek, Qualcomm AI Research, NVIDIA, Google and Computer Vision Lab / ETH Zürich.

Appendix A: Teams and affiliations

AIM2020 organizers

Members: Evangelos Ntavelis[1,2] (entavelis@ethz.ch), Siavash Bigdeli[2] (siavash.bigdeli@csem.ch), Andrés Romero[1] (roandres@ethz.ch), Radu Timofte[1] (radu.timofte@vision.ee.ethz.ch).
Affiliations: [1]Computer Vision Lab, ETH Zürich. [2]CSEM.

Rainbow

Title: Image fine-grained inpainting.
Members: Zheng Hui, Xiumei Wang, Xinbo Gao.
Affiliations: School of Electronic Engineering, Xidian University.

Yonsei-MVPLab

Title: Image Inpainting based on Edge and Frequency Guided Recurrent Convolutions.
Members: Chajin Shin, Taeoh Kim, Hanbin Son, Sangyoun Lee.
Affiliations: Image and Video Pattern Recognition Lab., School of Electrical and Electronic Engineering, Yonsei University, Seoul, South Korea.

BossGao

Title: Image Inpainting With Mask Awareness
Members: Chao Li, Fu Li, Dongliang He, Shilei Wen, Errui Ding
Affiliations: Department of Computer Vision (VIS), Baidu Inc.

ArtIst

Title: Fast Light-Weight Network for Image Inpainting
Members: Mengmeng Bai, Shuchen Li
Affiliations: Samsung R&D Institute China-Beijing (SRC-Beijing)

DLUT

Title: Iterative Confidence Feedback and Guided Upsampling for filling large holes and inpainting high-resolution images
Members: Yu Zeng[1], Zhe Lin[2], Jimei Yang[2], Jianming Zhang[2], Eli Shechtman[2], Huchuan Lu[1]
Affiliations: [1]Dalian University of Technology, [2]Adobe

AI-Inpainting Group

Title: MSEM: Multi-Scale Semantic-Edge Merged Model for Image Inpainting
Members: Weijian Zeng, Haopeng Ni, Yiyang Cai, Chenghua Li
Affiliations: Rensselaer Polytechnic Institute

qwq

Title: Markovian Discriminator guided Attentive Fractal Network
Members: Dejia Xu, Haoning Wu, Yu Han
Affiliations: Peking University

CVIP Inpainting Team

Title: Global Spatial-Channel Attention and Inter-layer GRU-based Image Inpainting
Members: Uddin S. M. Nadim, Hae Woong Jang, Soikat Hasan Ahmed, Jungmin Yoon, and Yong Ju Jung
Affiliations: Computer Vision and Image Processing (CVIP) Lab, Gachon University.

DeepInpaintingT1

Title: Deep Generative Inpainting Network for Extreme Image Inpainting
Members: Chu-Tak Li, Zhi-Song Liu, Li-Wen Wang, Wan-Chi Siu, Daniel P.K. Lun
Affiliations: Centre for Multimedia Signal Processing, Department of Electronic and Information Engineering, The Hong Kong Polytechnic University, Hong Kong

IPCV IITM

Title: Contextual Residual Aggregation Network
Members: Maitreya Suin, Kuldeep Purohit, A. N. Rajagopalan
Affiliations: Indian Institute of Technology Madras, India

MultiCog

Title: Pix2Pix for Image Inpainting
Members: Pratik Narang[1], Murari Mandal[2], Pranjal Singh Chauhan[1]
Affiliations: [1]BITS Pilani, [2]MNIT Jaipur

References

1. A tensorflow implementation of srgan. `https://github.com/tensorlayer/srgan.git`
2. Alom, M.Z., Hasan, M., Yakopcic, C., Taha, T.M., Asari, V.K.: Recurrent residual convolutional neural network based on u-net (r2u-net) for medical image segmentation. arXiv preprint arXiv:1802.06955 (2018)
3. Bai, M., Li, S., Fan, J., Zhou, C., Zuo, L., Na, J., Jeong, M.: Fast light-weight network for extreme image inpainting challenge. In: European Conference on Computer Vision Workshops (2020)
4. Bau, D., Strobelt, H., Peebles, W., Wulff, J., Zhou, B., Zhu, J., Torralba, A.: Semantic photo manipulation with a generative image prior. ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH) **38**(4) (2019)
5. Caesar, H., Uijlings, J., Ferrari, V.: Coco-stuff: Thing and stuff classes in context. In: CVPR (2018), `https://arxiv.org/abs/1612.03716`
6. Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using rnn encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078 (2014)
7. El Helou, M., Zhou, R., Süsstrunk, S., Timofte, R., et al.: AIM 2020: Scene relighting and illumination estimation challenge. In: European Conference on Computer Vision Workshops (2020)
8. Fritsche, M., Gu, S., Timofte, R.: Frequency separation for real-world super-resolution. In: 2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW). pp. 3599–3608. IEEE (2019)
9. Fuoli, D., Huang, Z., Gu, S., Timofte, R., et al.: AIM 2020 challenge on video extreme super-resolution: Methods and results. In: European Conference on Computer Vision Workshops (2020)
10. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: Advances in neural information processing systems. pp. 2672–2680 (2014)
11. Hong, S., Yan, X., Huang, T.E., Lee, H.: Learning hierarchical semantic image manipulation through structured representations. In: Advances in Neural Information Processing Systems. pp. 2713–2723 (2018)
12. Hui, Z., Li, J., Wang, X., Gao, X.: Image fine-grained inpainting. arXiv preprint arXiv:2002.02609 (2020)
13. Ignatov, A., Timofte, R., et al.: AIM 2020 challenge on learned image signal processing pipeline. In: European Conference on Computer Vision Workshops (2020)

14. Ignatov, A., Timofte, R., et al.: AIM 2020 challenge on rendering realistic bokeh. In: European Conference on Computer Vision Workshops (2020)
15. Isola, P., Zhu, J.Y., Zhou, T., Efros, A.A.: Image-to-image translation with conditional adversarial networks (2016)
16. Johnson, J., Alahi, A., Fei-Fei, L.: Perceptual losses for real-time style transfer and super-resolution. In: European conference on computer vision. pp. 694–711. Springer (2016)
17. Li, C.T., Siu, W.C., Liu, Z.S., Wang, L.W., Lun, D.P.K.: DeepGIN: deep generative inpainting network for extreme image inpainting. In: European Conference on Computer Vision Workshops (2020)
18. Li, C., Wand, M.: Precomputed real-time texture synthesis with markovian generative adversarial networks. In: European conference on computer vision. pp. 702–716. Springer (2016)
19. Lim, J.H., Ye, J.C.: Geometric gan. arXiv preprint arXiv:1705.02894 (2017)
20. Nazeri, K., Ng, E., Joseph, T., Qureshi, F., Ebrahimi, M.: Edgeconnect: Structure guided image inpainting using edge prediction. In: The IEEE International Conference on Computer Vision (ICCV) Workshops (Oct 2019)
21. Nazeri, K., Ng, E., Joseph, T., Qureshi, F.Z., Ebrahimi, M.: Edgeconnect: Generative image inpainting with adversarial edge learning. arXiv preprint arXiv:1901.00212 (2019)
22. Ntavelis, E., Romero, A., Kastanis, I., Van Gool, L., Timofte, R.: Sesame: Semantic editing of scenes by adding, manipulating or erasing objects. In: Proceedings of the European Conference on Computer Vision (ECCV) (2020)
23. Ntavelis, E., Romero, A., Bigdeli, S.A., Timofte, R., et al.: AIM 2020 challenge on image extreme inpainting. In: European Conference on Computer Vision Workshops (2020)
24. Park, T., Liu, M.Y., Wang, T.c., Zhu, J.Y.: Semantic image synthesis with spatially-adaptive normalization. In: CVPR. pp. 2337–2346 (2019)
25. Shaker, N., Liapis, A., Togelius, J., Lopes, R., Bidarra, R.: Constructive generation methods for dungeons and levels, pp. 31–55. Springer International Publishing, Cham (2016). https://doi.org/10.1007/978-3-319-42716-4_3, https://doi.org/10.1007/978-3-319-42716-4_3
26. Son, S., Lee, J., Nah, S., Timofte, R., Lee, K.M., et al.: AIM 2020 challenge on video temporal super-resolution. In: European Conference on Computer Vision Workshops (2020)
27. Wang, X., Yu, K., Wu, S., Gu, J., Liu, Y., Dong, C., Qiao, Y., Change Loy, C.: Esrgan: Enhanced super-resolution generative adversarial networks. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 0–0 (2018)
28. Wei, P., Lu, H., Timofte, R., Lin, L., Zuo, W., et al.: AIM 2020 challenge on real image super-resolution. In: European Conference on Computer Vision Workshops (2020)
29. Xiangli, Y., Deng, Y., Dai, B., Loy, C.C., Lin, D.: Real or not real, that is the question. In: ICLR (2020)
30. Xu, D., Chu, Y., Sun, Q.: Moire pattern removal via attentive fractal network. In: The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops (June 2020)
31. Yi, Z., Tang, Q., Azizi, S., Jang, D., Xu, Z.: Contextual residual aggregation for ultra high-resolution image inpainting. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 7508–7517 (2020)

32. Yu, J., Lin, Z., Yang, J., Shen, X., Lu, X., Huang, T.S.: Generative image inpainting with contextual attention. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 5505–5514 (2018)
33. Yu, J., Lin, Z., Yang, J., Shen, X., Lu, X., Huang, T.S.: Free-form image inpainting with gated convolution. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 4471–4480 (2019)
34. Zeng, Y., Lin, Z., Yang, J., Zhang, J., Shechtman, E., Lu, H.: High-resolution image inpainting with iterative confidence feedback and guided upsampling. In: European Conference on Computer Vision. Springer (2020)
35. Zhang, K., Danelljan, M., Li, Y., Timofte, R., et al.: AIM 2020 challenge on efficient super-resolution: Methods and results. In: European Conference on Computer Vision Workshops (2020)
36. Zhou, B., Lapedriza, A., Khosla, A., Oliva, A., Torralba, A.: Places: A 10 million image database for scene recognition. IEEE transactions on pattern analysis and machine intelligence **40**(6), 1452–1464 (2017)
37. Zhou, B., Zhao, H., Puig, X., Fidler, S., Barriuso, A., Torralba, A.: Scene parsing through ade20k dataset. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 633–641 (2017)