SemIE: Semantically-Aware Image Extrapolation – Supplementary material

Bholeshwar Khurana[1,2] Soumya Ranjan Dash[1,2] Abhishek Bhatia[1,2]
Aniruddha Mahapatra[2] Hrituraj Singh[2,3] Kuldeep Kulkarni[2]

[1]IIT Kanpur [2]Adobe Research India [3]Triomics

{bkhurana, sodash, abbhatia, anmahapa, kulkulka}@adobe.com, hrituraj@triomics.in

1. Network Architectures

The stage-1 uses PSPnet [13] to obtain the semantic label maps of the input images. The generators used in stage-2 (semantic extrapolation), stage-3 (panoptic label map generation) and stage-4 (RGB image generation) are inspired from SPADE [7] generator and consist of 9 spade residual blocks. The multiscale discriminators used in stage-2 and stage-4 are similar to that in pix2pixHD [11]. The encoder of stage-4 (that forms VAE [4] with the generator) is inspired from the encoder used in SPADE [7]. The co-occurrence patch discriminator of stage-4 is inspired from the one used in Swapping AutoEncoders [8]. Our stage-3 which converts the semantic label map into panoptic label maps [1] is a pure-generator network and is inspired by [1].

2. Panoptic Map generation (stage-3)

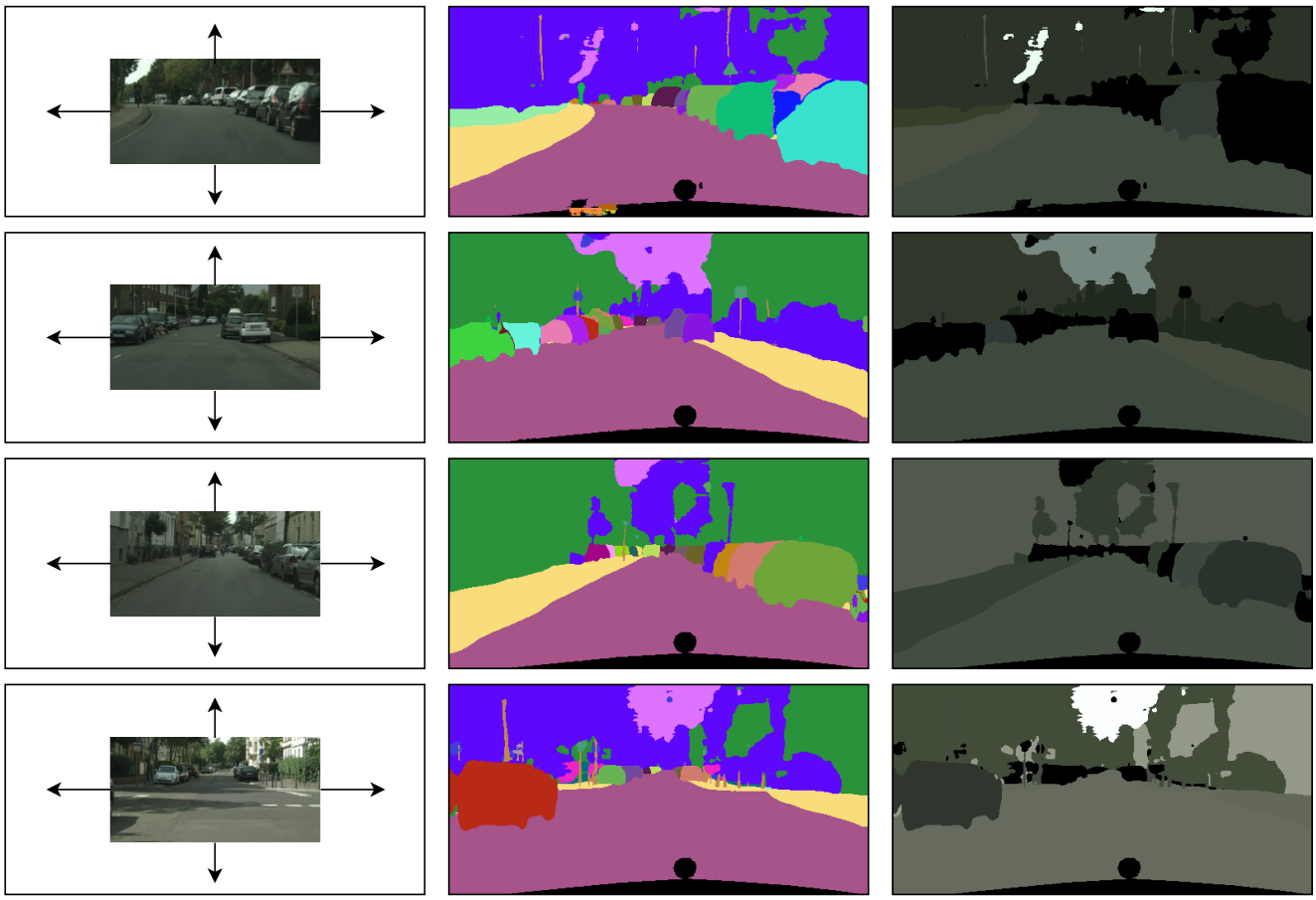
The stage-2 of our pipeline extrapolates the semantic label maps. However, in order to differentiate between multiple instances of the same class, we need to generate the panoptic label map from the thus extrapolated semantic label maps. In a typical panoptic label segmentation set-up, we have access to the full image. However, in our case we do not have access to the full image, but we have access to only the extrapolated semantic label map which is obtained as output of stage-2. To obtain the panoptic label maps from outputs of stage-2, we take inspiration from [1]. Panoptic-DeepLab proposes a method to convert the image into two parallel branches 1. semantic label maps and 2. pixel-wise instance center maps and x and y off-set maps from the instance centers. The center maps and the off-set maps thus predicted are used in conjunction with the semantic label maps to obtain the final panoptic label map. For the panoptic segmentation branch, [1] obtain class-agnostic instance centers and off-sets from the instance centers for every location. Class-agnostic centers refer to center locations for the different instances that belong to the 'things' categories. In addition, for every pixel that belongs to the 'things' categories, we define the x-offsets and y-offsets as δx and δy, respectively, of that pixel location from the center of the instance the pixel belongs to. Here, instead of having the above mentioned two parallel branches, we train a network to obtain the center maps and the x and y offset maps from the semantically extrapolated label maps that are the outputs of stage-2. The ground-truth center maps are represented by Gaussian blobs of standard deviation of 8 pixels, centered at the instance centers. We use a simple L_2 loss to compute the instance center loss and L_1 loss to compute the offset losses. The final loss for stage-3 is the weighted sum of the center loss and the offset losses.

During the test time, we adapt the procedure mentioned by [1] to group the pixels based on the predicted centers and off-sets to form instance masks. The instance masks and the semantic label map (the input to stage-3) are combined by majority voting to obtain the panoptic label map. The thus obtained panoptic label map is used in stage-4 for our instance-aware context normalization as well as to obtain the instance boundary maps.

3. Instance-aware Context Normalization (IaCN)

This module takes in input the cropped RGB image and the generated panoptic label map from stage-3. The panoptic label map is used to get the partial instances. Partial instances refer to the instances which are part of the input image and need to be completed in the final out-painted image. The per-channel (for RGB) mean color for each partial instance is calculated as mean pixel value of all the pixels belonging to that instance. The feature map is obtained by copying these mean colors (per-channel) to their respective extrapolated part of the partial instances in the outside region. For all the instances that do not belong to partial instances, the feature map values are zero. Figure 1 shows some of the input-output pairs for IaCN module.

[1]We use the term 'panoptic label map' and 'instance map' interchangeably



(a) Input image (b) Instance map (c) Feature map obtained by IaCN

Figure 1. **Input-Output for IaCN**. It computes the mean colors for all the partial instances from the cropped image and just 'pastes' them in the extrapolated regions of the corresponding instances. For all the other instances, it has zero value (black).

4. Detailed Training and Testing algorithms

The detailed training algorithms for stage-2 and stage-4 are given in Algorithm 1 and 2.

To train stage-2 (as shown in Algorithm 1), we use the ground truth segmentation map S_{gt}, ground truth panoptic map P_{gt} and cropped segmentation map S^c_{gt} (obtained from stage-1) as the inputs. We obtain segmentation map, S^z_{gt}, of desired resolution by zero padding S^c_{gt}. An instance boundary map, B is created from P_{gt}. Extrapolated segmentation map S_{pog} is generated using the generator G^2 which has an extra output channel (apart from the input classes) for the boundary map. The multiscale discriminator, $D^2_{\text{multiscale}}$ distinguishes between the generated segmentation map (S_{pog}) and the ground truth segmentation map (S_{gt}). The model tries to minimize the train objective function for semantic label map extrapolation (Section 5.1). Finally, the parameters of G^2, $D^2_{\text{multiscale}}$ are updated accordingly.

To train stage-4 (as shown in Algorithm 2), we ob-

Algorithm 1: Training algorithm for stage-2

Input:
Ground Truth Segmentation Map:
$S_{gt} \in \{0,1\}^{2h \times 2w \times c}$,
Ground Truth Panoptic Map: $P_{gt} \in \mathbb{R}^{2h \times 2w \times 1}$,
Cropped Segmentation Map: $S^c_{gt} \in \{0,1\}^{h \times w \times c}$

Generate $S^z_{gt} \in \{0,1\}^{2h \times 2w \times c}$ by zero-padding S^c_{gt}
Generate Boundary Map
$B \longleftarrow GetBoundary(P_{gt})$
for *epoch in maxEpochs* **do**
  $S_{pog} \longleftarrow G^2(L_1)$
  $D^2_{\text{multiscale}}$ distinguishes between S_{pog} and S_{gt}
  Minimize the objective function
  Update the parameters of G^2, $D^2_{\text{multiscale}}$
return G^2

tain X_{com} by concatenating ground truth segmentation map (S_{gt}), input image (X^c_{gt}), boundary map obtained from ground truth instance map and the feature map obtained using IaCN module. The out-painted image (Y) is generated using the generator of stage-4 G^4, which takes in X_{com} and the encoded input image ($E^4(X^c_{gt})$). The multiscale discriminator $D^4_{\text{multiscale}}$ tries to distinguish between the generated image (Y) and the ground truth image (X_{gt}). The co-occurrence patch discriminator (D^4_{patch}) tries to distinguish between fake patch ($crop(Y)$) and real patch ($crop(X_{gt})$) obtained from the fake image and real images respectively. For each pair of fake patch and real patch, the patch discriminator takes 4 reference patches ($crops(X_{gt})$) from the real image. We take a total of 4 fake patch, real patch and reference patches combinations for one image. All the patches are of size 64×64. We, then, minimize the final objective function (Section 5.2) to update the parameters of $G^4, E^4, D^4_{\text{multiscale}}$ and D^4_{patch}.

Algorithm 2: Training algorithm for stage-4

Input:
Cropped Image: $X^c_{gt} \in \mathbb{R}^{h \times w \times 3}$,
Ground Truth Image: $X_{gt} \in \mathbb{R}^{2h \times 2w \times 3}$,
Ground Truth Segmentation Map:
$S_{gt} \in \{0,1\}^{2h \times 2w \times c}$,
Ground Truth Panoptic Map: $P_{gt} \in \mathbb{R}^{2h \times 2w \times 1}$

$X_{com} \longleftarrow S_{gt} \oplus X^c_{gt} \oplus GetBoundary(P_{gt}) \oplus IaCN(X^c_{gt}, P_{gt})$
for *epoch in maxEpochs* **do**
  $Y \longleftarrow G^4(\mathrm{X}_{com}, E^4(X^c_{gt}))$
  $D^4_{\text{multiscale}}$ distinguishes between Y and X_{gt}
  D^4_{patch} distinguishes between $crop(Y)$ and $crop(X_{gt})$, taking $crops(X_{gt})$ as the ref. patches
  Minimize the objective function
  Update the parameters of $G^4, E^4, D^4_{\text{multiscale}}, D^4_{\text{patch}}$
return E^4, G^4

The detailed testing algorithm is given in Algorithm 3. The semantic label map S^c corresponding to input image X is obtained from PSPnet [13] in stage-1. The extrapolated semantic label map S_{pog} is generated from generator G^2 of stage-2. S_{pog} is fed into stage-3 (Panoptic Label Generatior) to get panoptic map P'. S_{pog}, X^c_{gt}, boundary map obtained from P' and output of $IaCN$ are concatenated into X'_{com}, which is given as input to generator G^4 in stage-4 to generate the extrapolated RGB image Y.

Algorithm 3: Testing algorithm

Input: Image: $X \in \mathbb{R}^{h \times w \times 3}$
Output: Outpainted Image: $Y \in \mathbb{R}^{2h \times 2w \times 3}$

$S^c \longleftarrow$ **PSPNet**(X^c_{gt}) `// Stage-1`
$S_{pog} \longleftarrow G^2(S^c)$ `// Stage-2`
$P' \longleftarrow$ **PanopticLabelMap**(S_{pog}) `// Stage-3`
$X'_{com} \longleftarrow S_{pog} \oplus X \oplus GetBoundary(P') \oplus IaCN(X, P')$
$Y \longleftarrow G^4(X'_{com}, E^4(X))$ `// Stage-4`

5. Objective Functions

This section describes all the loss functions used in different stages of training in detail. All the notations for different variables is the same as described in Section 4.

5.1. Stage-2

The objective function for stage-2 includes 4 losses: GAN loss, discriminator feature matching loss, focal loss and cross-entropy loss.

GAN loss: S^c_{gt} is the semantic label map corresponding to input image, S_{gt} is the corresponding extrapolated ground truth semantic label map. B is the ground truth boundary map obtained from ground truth instance map P_{gt}. S_{com} is channel-wise contatenation of S_{gt} and B. S_{pog} is the combined extrapolated sematic label map and boundary map synthesised by G^2. We replace GAN hinge loss used in [7] with least square loss ($\mathcal{L}_{GAN}$).

Feature matching loss: For stability in GAN training, we use feature matching loss ($\mathcal{L}_{FM}$) which is defined as,

$$\mathcal{L}_{FM} = \sum_i \frac{1}{N_i}[\|D^{2(i)}(S_{com}) - D^{2(i)}(S_{pog})\|_1] \quad (1)$$

where $D^{2(i)}$ represents $i-$th layer of discriminator D^2 with N_i elements.

Cross-entropy loss: We apply a cross-entropy loss ($\mathcal{L}_{CE}$) on input Y''YY and Y defined as,

$$l(z,y) = -(ylog(z) + (1-y)log(1-z) \quad (2)$$

$$\mathcal{L}_{CE} = \sum_{h,w,c} l(z,y) \quad (3)$$

where $l(z,y)$ defines element wise cross-entropy loss for particular (h,w,c), and $\mathcal{L}_{CE}$ is the total cross-entropy loss. y and z are the ground-truth and generated values at each (h,w,c) location in S_{com} and S_{pog} respectively.

Focal loss: To better account for representation of rare semantic classes in the generated semantic label map, we

use focal loss [5] ($\mathcal{L}_{FL}$) defined as,

$$\mathcal{L}_{FL} = \sum_{h,w,c} l(z,y) \times (1-z)^{\gamma} \tag{4}$$

Training objective: The overall training objective for this stage, hence, is $\mathcal{L}_2$,

$$\mathcal{L}_2 = \min_{G^2}(\mathcal{L}_{GAN} + \lambda_{FM}\mathcal{L}_{FM} + \lambda_{CE}\mathcal{L}_{CE} + \lambda_{FL}\mathcal{L}_{FL}) \tag{5}$$

5.2. Stage-4

The objective function for stage-4 includes 5 losses: GAN loss, discriminator feature matching loss, perceptual loss, KL-Divergence loss and Patch co-occurrence discriminator loss.

GAN loss: X^c_{gt} is the input image. X_{gt} is the extrapolated ground truth image. S_{gt} is the ground truth semantic label map and P_{gt} is the ground truth instance map corresponding to S_{gt}. X_{com} is channel-wise concatenation of X^c_{gt}, S_{gt}, boundary map obtained from P_{gt} and output of $IaCN$ module. Y is the extrapolated RGB image synthesised by G^4. For GAN loss ($\mathcal{L}_{GAN}$) we use hinge loss similar to [7]

Feature matching loss: For stability in GAN training we use feature matching loss ($\mathcal{L}_{FM}$) which is defined as,

$$\mathcal{L}_{FM} = \sum_i \frac{1}{N_i}[\|D^{4(i)}(X^c_{gt}) - D^{4(i)}(Y)\|_1] \tag{6}$$

where $D^{4(i)}$ represents $i-$th layer of discriminator D^4 with N_i elements.

Perceptual loss: We use VGG19 as feature extractor to minimize the **L1** loss between features extracted for Z' and Y. Particularly, we define perceptual loss ($\mathcal{L}_{VGG}$) as,

$$\mathcal{L}_{VGG} = \sum_i \frac{1}{N_i}[\|\Phi^{(i)}(X^c_{gt}) - \Phi^{(i)}(Y)\|_1] \tag{7}$$

where Φ^i denotes $i-$th layer of VGG19 network.

KL divergence loss: We use KLD loss ($\mathcal{L}_{KLD}$), similar to [7] to train the VAE (defined in Section 3.4 in main paper). It is defined as,

$$\mathcal{L}_{KLD} = \mathbb{D}_{\mathbb{KL}}(q(z|x)||p(z)) \tag{8}$$

where q is variation distribution and $p(z)$ is a standard Gaussian distribution.

Patch Co-Occurrence discriminator loss: In addition to multiscale discriminator D^4, we also use a patch co-occurrence discriminator (described in detail in Section 3.4 of main paper). The patch co-occurrence discriminator loss $\mathcal{L}_{CooccurGAN}$ is defined as,

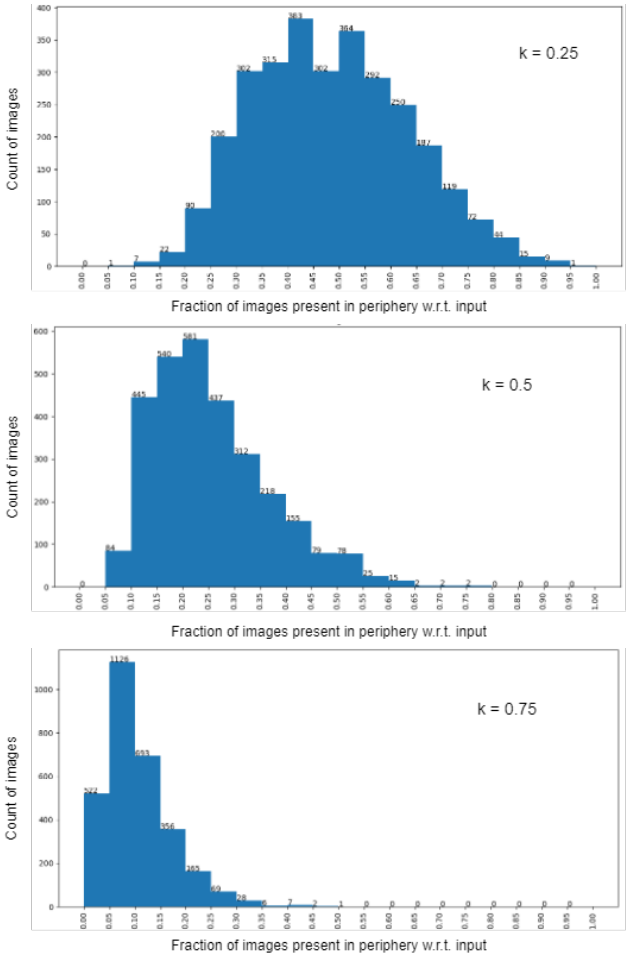


Figure 2. Fraction/total number of images in the cityscapes training set where the ratio of number of instances outside the input crop to the number of instances inside the input crop. For $k = 0.25$, the peak occurs around 50%, for $k = 0.5$, the peak occurs around 25% and for $k = 0.75$, the peak occurs around 10%.

$$\mathcal{L}_{CooccurGAN} = \mathbb{E}_{Y,X^c_{gt}}[-log(D^4_{patch}(crop(G(Y)), crop(X^c_{gt}), crops(X^c_{gt})))] \tag{9}$$

where $crop(X^c_{gt})$ function takes a random patch of 64×64 from image X^c_{gt} and $crops(X^c_{gt})$ takes 4 random patches from image X^c_{gt}, which serve as the reference patches.

Training objective: The overall training objective for this stage, hence, is $\mathcal{L}_4$,

$$\mathcal{L}_4 = \min_{G^4}(\mathcal{L}_{GAN} + \lambda_{FM}\mathcal{L}_{FM} + \lambda_{VGG}\mathcal{L}_{VGG} + \lambda_{VGG}\mathcal{L}_{VGG} + \mathcal{L}_{CooccurGAN}) \tag{10}$$

6. Additional Implementation details

We trained and tested our models on Cityscapes [2] and ADE-20K bedroom subset [14] datasets. For stage-2 we used a batch size of 8, while for stage-4 we used a batch size of 16. All the experiments were run on 4 16GB Nvidia Tesla V100 GPUs. Both the datasets were trained for 200 epochs. We used the TTUR [3] update rule.

7. Different crop analysis

We analyzed different crop sizes on Cityscapes dataset. For given images $X_0 (\in \mathbb{R}^{h \times w \times c})$ in the dataset, we tried 3 different crop ratios $k = \{0.25, 0.5, 0.75\}$. We, thus, obtain the cropped images $X (\in \mathbb{R}^{k.h \times k.w \times c})$. We calculated the number of instances outside the crop region with respect to those inside. If the number of instances in the region to be extrapolated is very high as compared to those inside, the information in the input data may not be sufficient to obtain a reasonable extrapolation. While if it is low, the number of new instances to be generated decreases, making the task too easy for our algorithm. After analyzing the results (Figure 2), we observed that the value of $k = 0.5$ gives a good trade-off between the two extremes.

8. Human Evaluation

We compare our method with the baselines also via human subjective study. The baselines included results from Partial Convolutions (PConv) [6], Boundless [10] and OutPainting-SRN [12]. About hundred random people were given a set of 20 randomly selected images from both the datasets. They were given unlimited time to make the selection. They were asked to rank the images based on two parameters, viz. Realistic appearance and New object generation. Plot 3 shows the evaluation results. We found that our results were strongly favoured by most of the people.

9. Infinite Zoom

Our model has ideally an infinite zooming out potential. We can zoom out an image upto a good extent by recursively passing through our model. We experimented this by training on modified Cityscapes dataset, made by removing the Mercedes at the bottom and then upsampling the images to the original dimension. We did this since if we "infinitely" zoomed out the original images, then there would have been Mercedes recursively at the bottom in the zoomed out images.

As shown in Figure 4, the input image is first passed through our model to generate the output image, extrapolated 2 times in length and breadth. The output image thus generated is downsampled to half the size of the output image and is zero padded to generate the input image for the next iteration. We continue this process recursively 6 times, in course of which we generate the $2\times, 4\times \cdots$ till $64\times$ extrapolated images. To generate the frames for the video, after generating the $2\times$ extrapolated image, we downsample it gradually by 2 pixels in height and 4 pixels in width and zero pad it. Thus, for every input-output transition we generate 64 frames. There are 6 input-output transitions and hence, 384 frames. We also add 30 frames of the initial input at the beginning of the video for better realisation of the input image. The videos corresponding to some of the data images can be found in the project page https://semie-iccv.github.io/.

10. Segmentation maps comparison

We highlighted the importance of using extra boundary channel in semantic label extrapolation for better shapes of the instances. In Figure 5 and 6, we compare our results with those generated without boundary channel. We further compare them with those generated using SPGNet [9] and SPGNet++.

11. Why do single stage approaches fail?

All the baseline methods except SPG-Net are based on direct image-to-image translation in a single stage. To analyse the strength of our approach and the failure of single stage approaches, we try direct image extrapolation by image-to-image translation. For this, during the training time, we used the RGB cropped input image and extrapolated it to the final RGB image using the stage-4, without IaCN and D_{patch}, directly (Figure 7) using SPADE network. We observed that the images are generated with mere texture extension at the periphery with minimal to no new object generation.

12. Additional results

In Figure 8 and 9, we show additional synthesized results from our method on Cityscapes dataset and compare them with Partial Convolutions (PConv) [6], Boundless [10], OutPainting-SRN [12], SPGNet [9] and SPGNet++.

In Figure 10, we show additional synthesized results from our method on ADE20K dataset and compare them with Partial Convolutions (PConv) [6], Boundless [10], OutPainting-SRN [12], SPGNet [9] and SPGNet++.

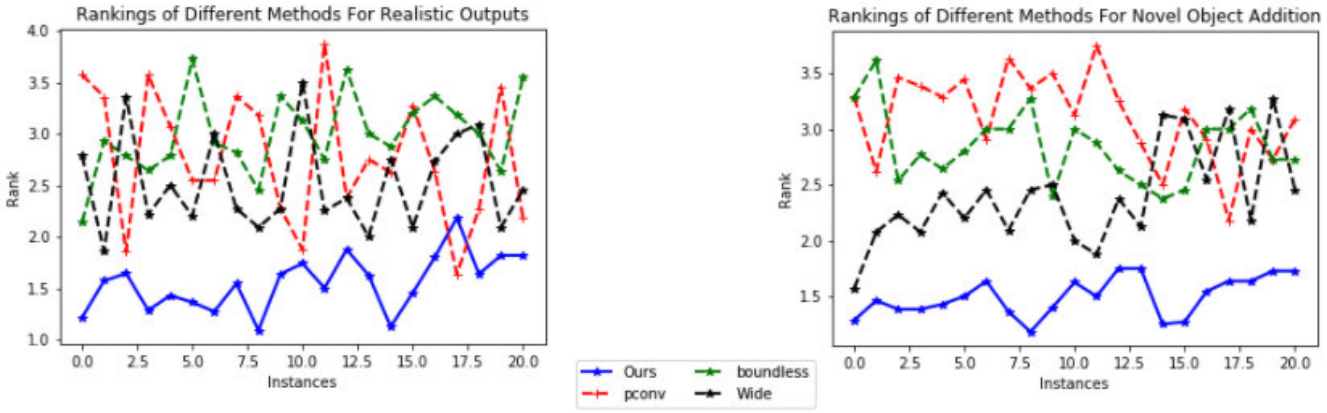


Figure 3. User preference study on the rank of various baselines and our method. Lower rank means better.

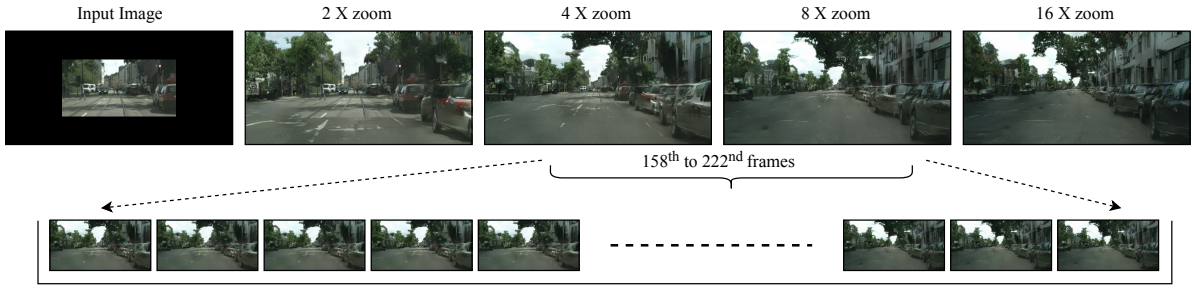


Figure 4. Given an input image, we extrapolate the image recursively to obtain $2\times, 4\times, 8\times, 16\times, 32\times, 64\times$ extrapolations. Using the extrapolated images, we obtain frames to make a video by performing naive 'interpolation' between the extrapolated images. We achieve this by taking several crops of increasing sizes followed by bicubic downsampling to ensure all frames are of the same size. We show in the figure how frames are generated by 'interpolating' between the $4\times$ and $8\times$ extrapolations.

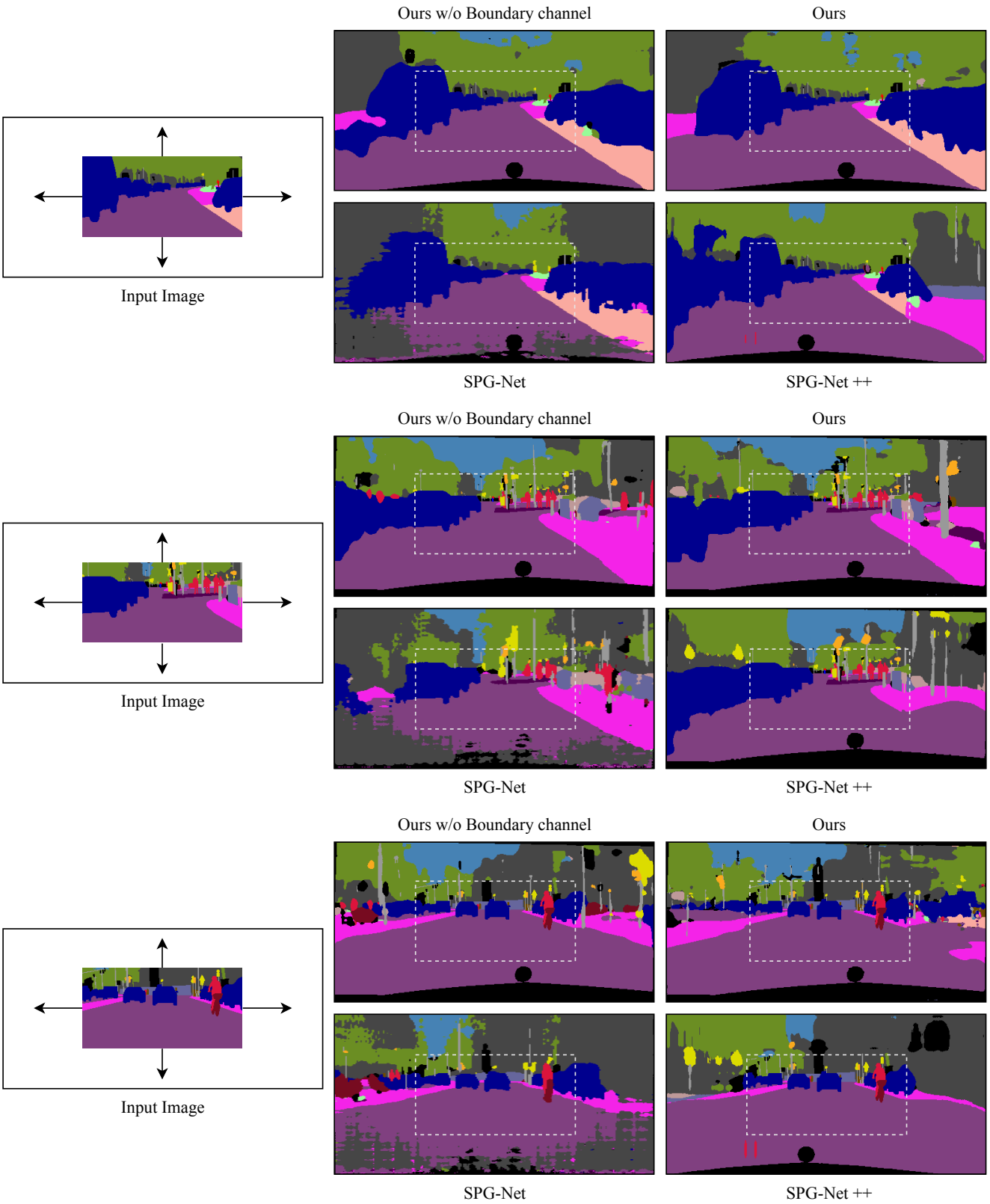


Figure 5. Additional semantic label extrapolation comparison. For each input image, the output of the PSPNet (stage-1) is shown in the leftmost column and the corresponding semantic extrapolation methods are shown. Top row shows results for the two variants of our method, with and without boundary channel regularizer. The bottom rows results for the two variants of the baseline, [9] and SPG-Net++.

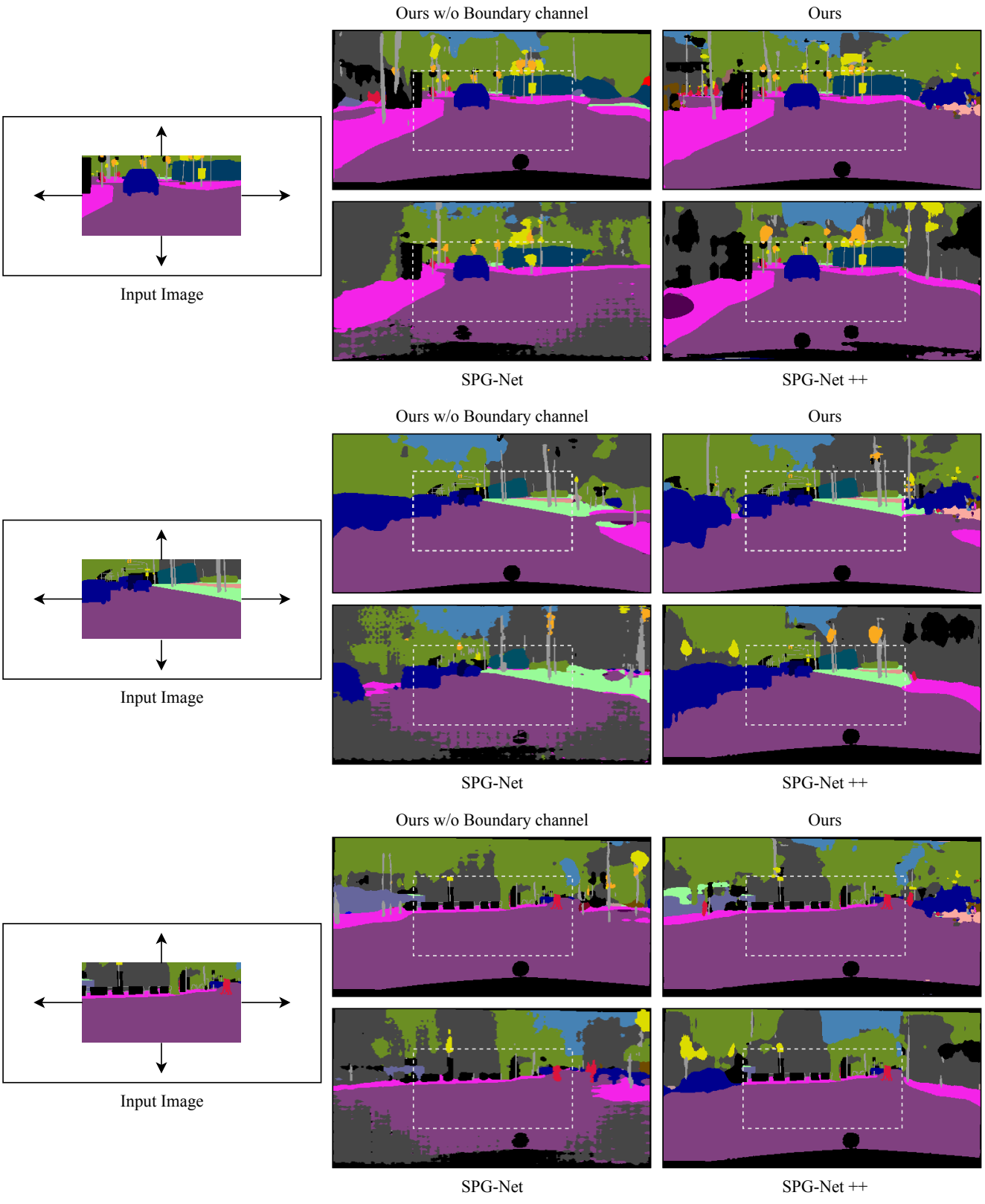


Figure 6. Additional semantic label extrapolation comparison. For each input image, the output of the PSPNet (stage-1) is shown in the leftmost column and the corresponding semantic extrapolation methods are shown. Top row shows results for the two variants of our method, with and without boundary channel regularizer. The bottom rows results for the two variants of the baseline, [9] and SPG-Net++.

(a) Input image (b) Extrapolation using single-stage appraoch (c) Extrapolation using our approach

Figure 7. Comparison between ours and single stage approach: (a) Input image (b) Extrapolated image using our single stage approach (c) Extrapolated image using our current approach.

Figure 8. Additional comparisons between our model and the baselines on Cityscapes dataset. The baselines include results from Partial Convolutions (PConv) [6], Boundless [10], OutPainting-SRN [12], SPGNet [9], and SPGNet++.

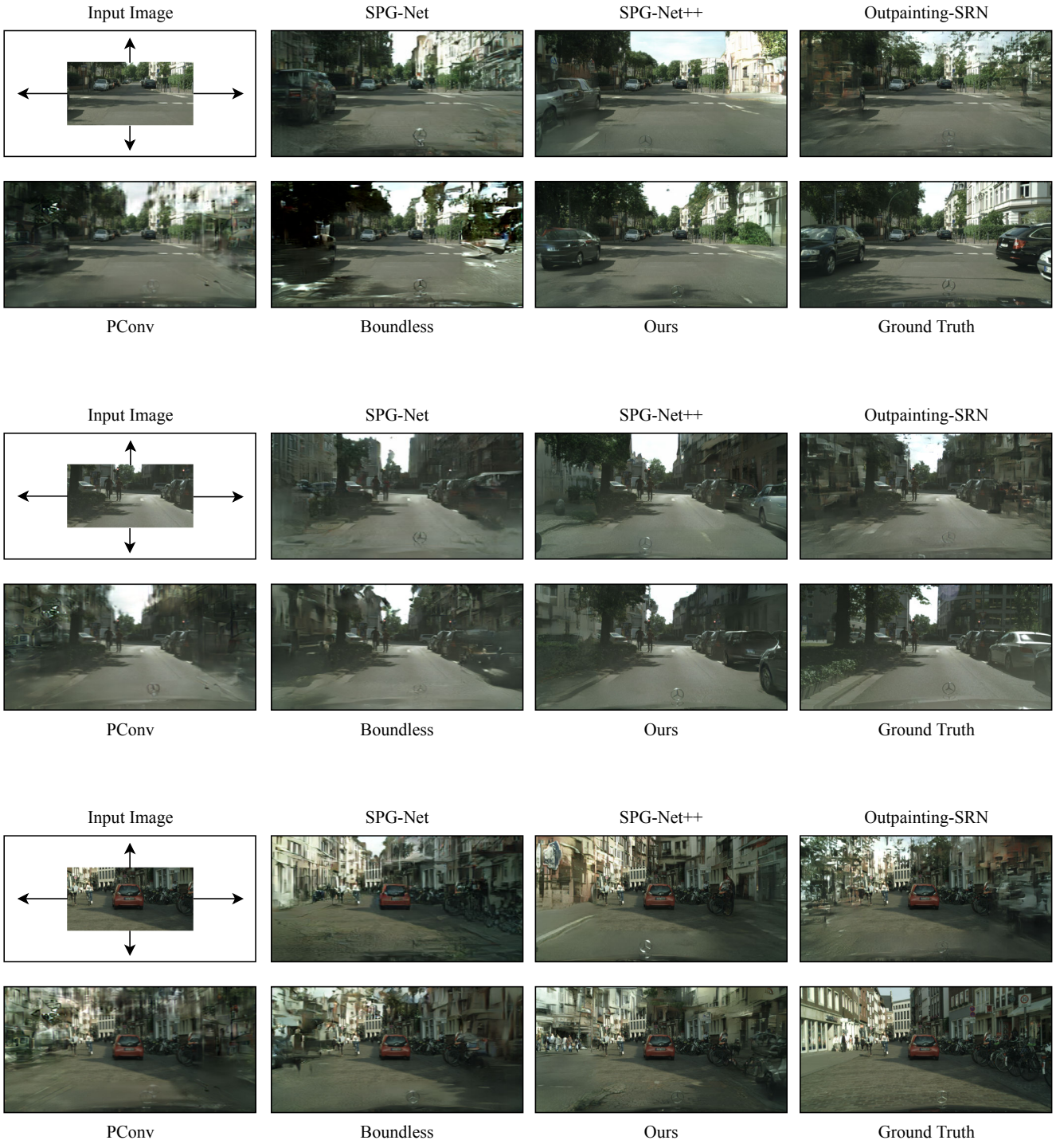


Figure 9. Additional comparisons between our model and the baselines on Cityscapes dataset. The baselines include results from Partial Convolutions (PConv) [6], Boundless [10], OutPainting-SRN [12] and SPGNet [9], SPGNet++.

Figure 10. Additional comparisons between our model and the baselines on ADE-20K dataset. The baselines include results from Partial Convolutions (PConv) [6], Boundless [10], OutPainting-SRN [12], SPGNet [9], and SPGNet++.

References

[1] Bowen Cheng, Maxwell D Collins, Yukun Zhu, Ting Liu, Thomas S Huang, Hartwig Adam, and Liang-Chieh Chen. Panoptic-deeplab: A simple, strong, and fast baseline for bottom-up panoptic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12475–12485, 2020. 1

[2] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3223, 2016. 5

[3] 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 *Advances in neural information processing systems*, pages 6626–6637, 2017. 5

[4] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 1

[5] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017. 4

[6] 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 *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 85–100, 2018. 5, 10, 11, 12

[7] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2337–2346, 2019. 1, 3, 4

[8] Taesung Park, Jun-Yan Zhu, Oliver Wang, Jingwan Lu, Eli Shechtman, Alexei Efros, and Richard Zhang. Swapping autoencoder for deep image manipulation. *Advances in Neural Information Processing Systems*, 33, 2020. 1

[9] 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. *arXiv preprint arXiv:1805.03356*, 2018. 5, 7, 8, 10, 11, 12

[10] Piotr Teterwak, Aaron Sarna, Dilip Krishnan, Aaron Maschinot, David Belanger, Ce Liu, and William T Freeman. Boundless: Generative adversarial networks for image extension. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 10521–10530, 2019. 5, 10, 11, 12

[11] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8798–8807, 2018. 1

[12] Yi Wang, Xin Tao, Xiaoyong Shen, and Jiaya Jia. Wide-context semantic image extrapolation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1399–1408, 2019. 5, 10, 11, 12

[13] 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. 1, 3

[14] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 5