Suplementary Material for DejaVu: Conditional Regenerative Learning to Enhance Dense Prediction

Shubhankar Borse Debasmit Das * Hyojin Park * Hong Cai Risheek Garrepalli
Fatih Porikli
Qualcomm AI Research †
{sborse, debadas, hyojinp, hongcai, rgarrepa, fporikli}@qti.qualcomm.com

Appendices

A. Introduction

As part of the supplementary materials for this paper, we present our hyper-parameters and show more visual and quantitative results as an extension to the ones shown in the paper. The supplementary materials contain:

- Additional experiments and ablation studies. These include:
  - Results using the DejaVu-CL Cyclic consistency loss.
  - Results comparing with Inpainting-based augmentation strategies and Masked Auto-encoders.
  - Experiment studying the effect of various redaction strategies on the two CRM modules.
  - Results obtained by varying the capacity of the proposed CRM module.
  - Analyzing the added training cost of training with DejaVu loss.
- Implementation details and Hyperparameters for all experiments
- Additional qualitative results as an extension to the ones shown in the paper.

B. Additional Experiments

In this section, we provide additional ablation experiments as an extension to the ones shown in the paper. For every experiment, we report our results on the Cityscapes dataset, and apply the Vanilla training strategy as explained in Section C, which uses no pre-trained checkpoints / bells and whistles for providing a fair comparison.

B.1. DejaVu-CL Cyclic consistency loss

In Table B.1, we report the results of using cyclic consistency loss (DejaVu-CL, as explained in Section 3.4. of the main paper) along with DejaVu, with an HRNet18 base model on the Cityscapes semantic segmentation dataset. Then, we add Gaussian noise to the input images and control the noise levels by tweaking the varianceof the noise distribution, σ . For no noise level, having DejaVu loss improves over the baseline by about **2 mIoU** pts. It is to be noted that in the noise-less setting, DejaVu-CL does not provide any improvement over just using DejaVu. However, with increasing noise levels, DejaVu-CL provides higher gains, i.e. **1.3 mIoU** pts and **3 mIoU** pts for $\sigma = 0.1$ and $\sigma = 0.3$ respectively. This study concludes that training with Dejavu-CL generates a network which is more robust, and hence produces more consistent results.

*These authors contributed equally to this work.
†Qualcomm AI Research, an initiative of Qualcomm Technologies, Inc.

Base Network	**DejaVu**	**DejaVu-CL**	*mIoU*($\sigma = 0.0$)	*mIoU*($\sigma = 0.1$)	*mIoU*($\sigma = 0.3$)
HRNet18	✗	✗	75.2	70.7	61.5
	✓	✗	**77.0**	72.1	63.7
	✓	✓	**77.0**	**73.4**	**66.8**

Table B.1. **On Cityscapes:** Effect of training the baseline model with DVSA loss, varying noise.

B.2. Comparing with inpainting-based augmentation strategies and Masked Auto-encoders

In Table B.2, we vary the training method using an HRNet18 backbone on the Cityscapes dataset. For fair comparison, we use the **vanilla training strategy** as described in Section C to train the baseline and DejaVu models. To compare with Masked Autoencoders (MAE) [6], we pretrain using the masked image modelling technique proposed by them [6]. Furthermore, we also perform training using image redaction as the input augmentation type. MAE-based pretraining strategy performs better (75.8) compared to the baseline (75.2). However, using redaction based augmentations produces better recognition performance (76.0) compared to the baseline. Training with our DejaVu loss improves the baseline compared to both these methods by a minimum of **1 mIoU** point. This observation shows that using the same number of additional regenerated images as a data augmentation scheme does not provide the performance improvements that DejaVu can achieve and hence the loss gradients propagated from our CRM module enhances the base networks feature representation.

Base Network	**Augmentation**	**MAE**	**DejaVu**	*mIoU*↑
HRNet18	✗	✗	✗	75.2
	✓	✗	✗	76.0
	✗	✓	✗	75.8
	✗	✗	✓	**77.0**

Table B.2. **On Cityscapes:** Comparison with vanilla training using data augmentation, MAE pretraining and DejaVu.

B.3. Varying the capacity of the proposed CRM module

In Table B.3, we vary the capacity of the regeneration module CRM-F by varying the number of stages in the network. Results show that increasing the number of stages increases the segmentation performance to a value of 77.0 mIoU, when the number of stages is 2. However, increasing the capacity beyond 2 stages do not improve the performance as it does not provide additional context and higher complexity might slowly lead to overfitting.

Base Network	**DejaVu**	**Stages**	*mIoU*↑
HRNet18	✗	✗	75.2
	✓	1	76.7
	✓	2	**77.0**
	✓	3	76.9
	✓	4	76.6

Table B.3. **On Cityscapes:** Studying the tradeoff between capacity of the CRM-F network and the performance. As observed, the CRM module needs to be representative enough to generate the whole image. However, the performance reduces after adding more network capacity as its ability to distill the appropriate context into the base network reduces.

B.4. Studying the effect of various redaction strategies on different CRM modules

In Table B.4, we study the effect of using different spatial redaction types (Checkerboard and Random, as illustrated in Figure 3 of the main paper) and how they affect the performance when we switch between the two regeneration modules - the one-shot CRM-F and the recursive CRM-R, as described in Section 3.2 of the main paper. Checkerboard is a structured redaction type and seems to work better for CRM-F while the random redaction type seems to work better for CRM-R. CRM-R has a recursive structure and simulates a large field of view. It uses a difference based process to better handle reconstruction from noisy images. On the other hand, CRM-F needs a structured noise type to reconstruct the input image due to its single-shot architecture, eluding to a smaller field of view even while having a similar network capacity compared to the CRM-R.

Base Network	**DejaVu**	**Redaction**	**CRM-F**	**CRM-R**	*mIoU*↑
HRNet18	✗	✗	✗	✗	75.2
	✓	Checkerboard	✓	✗	**77.0**
	✓	Random	✓	✗	76.5
	✓	Checkerboard	✗	✓	76.8
	✓	Random	✗	✓	**77.0**

Table B.4. **On Cityscapes:** Comparing the difference between both kinds of CRM modules, and how DejaVu performance is affected by clubbing them with various kinds of spatial redaction (Checkerboard v/s Random).

Base Network	Dataset	**DejaVu**	*mIoU*↑	*seconds/iter*↓	*Memory(MB)*↓
HRNet18-OCR	Cityscapes	✗	80.7	2.2	✗
		✓	82.0	2.4	✗
PoolFormer-M48	ADE20K	✗	42.4	8.9	3890
		✓	43.3	11.0	3903

Table B.5. Studying the tradeoff between added training cost and increase in mIoU after trining with the proposed DejaVu loss. The measurements are reported using a single Tesla A100 GPU with batch size of 1.

B.5. Analyzing the added training cost when training with DejaVu loss

In Table B.5, we analyze the added training cost in terms of memory and seconds per iteration. From the results, the HRNet18-OCR model trained with DejaVu improves the mIoU on Cityscapes by **1.3** points with a **7% increase** in average training time. The PoolFormer-M48 model trained with DejaVu loss improves mIoU on ADE20K by **1.1** points with a **21% increase** in average training time.

C. Implementation details

In this section, we provide implementation details and hyperparams for all our experiments.

C.1. Experiments on Cityscapes

For the Cityscapes dataset, we build our code based on the code provided by Hierarchical Multi-Scale Attention [9] authors[1].

Baselines: We train HRNet [8], HRNet-OCR [10] and HRNet-OCR-HMS [9] baselines by varying the width multiplier (18, 48, etc. as provided in the original paper [8]). For experiments in Section 4 of the main paper, we train the baseline (and DejaVu) models using the **Enhanced Training Strategy**. For the experiments in Section B of the Appendix, we train all models using the **Vanilla training strategy** to perform fair comparisons with other methods such as Masked Autoencoders [6] and our Inpainting-based augmentation training strategy. For all experiments, we use 4 nvTesla A100 GPUs.

Vanilla training strategy: For the vanilla training strategy used in Tables B.1, B.2, B.3, and B.4, we initialize weights randomly, and train each network starting with a learning rate of 0.01 and apply the polynomial learning schedule for 175 epochs, setting the power β=2.

Enhanced training strategy: For the enhanced training strategy used in Tables 1 and 7 of the main paper, we initialize weights using ImageNet pretrained checkpoints and train each network starting with a lr of 0.01 and apply the polynomial learning schedule for 175 epochs, setting the power β=2. We also use the extended coarsely labeled data in Cityscapes and the pseudo-labels provided by [9]. These hyperparameters are the same as related papers which build on HRNet [1,2] in their codebase[2].

DejaVu: To train the models using our DejaVu loss, we use the same strategy as the baseline network in every Table. We initialize the CRM module with stored weights. We perform a hyperparameter search to obtain the most optimal type of redaction, along with the block size b or threshold t as indicated in Section 3.1 of the main paper. For the experiments in Table 1 of the paper, we use the 'random blocks' spatial redaction, with a block size $b = 32$ and threshold $t = 0.4$. We use the CRM-R module with number of recursion steps in Section 3.2 set as 20. From Equations 1 and 2 of the main paper, we set $\gamma = 1$, $\gamma_1 = 1$ and $\gamma_2 = 0.3$ as the loss weights for the DejaVu loss. For the DejaVu-TS experiment shown in Table 7, we add the loss term from Equation 3 in the paper to our main loss using a weight of 0.3. For the DejaVu-CL experiment

[1]https://github.com/NVIDIA/semantic-segmentation

[2]https://github.com/Qualcomm-AI-research/InverseForm

shown in Table B.1, we add the cyclic loss term to our main loss using a weight of 0.1. These hyper-parameters are set for the segmentation task, and used throughout the paper.

DejaVu-SA: To train the models using our DejaVu-SA attention block and obtain our results in Figure 8 of the paper, we use the enhanced training strategy for every point on the graph, including base networks (HRNet18, HRNet19 and HRNet20) and the HRNet18-DejaVu model. We varied the embedding dimension of the Multi Head Attention (MHA) block (setting it at $32, 64, 128$ for the respective plot points in increasing order) to scale our HRNet18-DejaVu model. We apply the spectral redaction using a bandstop filter, with a block size of $b = 32$. From Equations 1 and 2 of the main paper, we set $\gamma = 1$, $\gamma_1 = 1$ and $\gamma_2 = 0.3$ as the loss weights for the DejaVu loss.

C.2. Experiments on ADE20K

We report results on ADE20K in Table 3 of the main paper. For all the experiments on the ADE20K dataset, we build our code on top of the mmsegmentation [4] repository[3].

Baselines: To reproduce the PoolFormer48 with SemanticFPN baseline, we apply the 40K training schedule using this config. To reproduce the UpPerNet with ViT-B baseline, we apply the 80K learning schedule using this config. As for DenseCLIP, we use this config to train using the 80K schedule. Our baseline numbers are obtained by reproducing the scores on our end, and hence might be slightly different from reported. We observe a considerable difference mainly in DenseCLIP, as reported by several issues. For all experiments, we use 2 nvTesla A100 GPUs.

DejaVu: To train the models using our DejaVu loss, we use the same strategy as the baseline networks. As set in Section C.1, we use the 'random blocks' spatial redaction, and set a block size $b = 32$ and threshold $t = 0.4$. We use the CRM-R module with number of recursion steps in Section 3.2 set as 20. From Equations 1 and 2 of the main paper, we set $\gamma = 1$, $\gamma_1 = 1$ and $\gamma_2 = 0.3$ as the loss weights for the DejaVu loss.

C.3. Experiments on COCO

We report results on COCO in Table 2 of the main paper. For all the experiments on the COCO dataset, we build our code on top of the mmdetection [3] repository[4].

Baselines: To reproduce the Mask2Former with Swin-T baseline, we apply the 160K training schedule using this config. For reproducing Mask2Former using the Swin-L backbone, we train this config with the 320k schedule. For all experiments, we use 4 nvTesla A100 GPUs.

DejaVu: To train the models using our DejaVu loss, we use the same strategy as the baseline networks. For Panoptic Segmentation, we use the 'checkerboard' spatial redaction and set a block size of $b = 8$. We use the CRM-F module with two stages. From Equations 1 and 2 of the main paper, we set $\gamma = 1$, $\gamma_1 = 1$ and $\gamma_2 = 0.1$ as the loss weights for the DejaVu loss. As MSE loss has a lower weight, we observe that the regenerated image in Figure 6 of the main paper signifies the dominance of LPIPS score.

C.4. Experiments on KITTI

We report results on KITTI self-supervised monocular depth estimation in Table 4 of the main paper. For this experiment, we build our code on top of the monodepth2 [5] repository[5].

Baseline: We train the baseline monodepth2 model with a ResNet50 backbone at $640x192$ resolution. We use the default hyperparameters provided by the authors.

DejaVu: To train the models using our DejaVu loss, we use the same hyperparameters as the baseline networks. For depth estimation, we use the 'bandstop' spectral redaction and set a block size of $b = 32$. We use the CRM-F module with two stages. From Equations 1 and 2 of the main paper, we set $\gamma = 1$, $\gamma_1 = 0.3$ and $\gamma_2 = 0.03$ as the loss weights for the DejaVu loss.

C.5. Experiments on NYUD-v2

We report results on NYUD-v2 for multi-task, partially supervised multi-task and single task learning, in Tables 6, 5 and 8 respectively. We report performance on segmentation, depth estimation and surface normal estimation. For this experiment, we build our work on top of the MTPSL [7] repository[6].

[3] https://github.com/open-mmlab/mmsegmentation
[4] https://github.com/open-mmlab/mmdetection
[5] https://github.com/nianticlabs/monodepth2
[6] https://github.com/VICO-UoE/MTPSL

Baselines: For Tables 5 and 6 of the main paper, we use both the SegNet-MTL and SegNet-XTC baselines as provided by the authors, for the multi task (partially supervised/fully supervised) learning setting. For Table 8 and the graph on Figure 7, we build our work on top of the single-task SegNet baseline. We use the default training scheme set by the authors, which consists of 200 epochs.

DejaVu: To train the models using our DejaVu loss, we use the same hyperparameters as the base networks. For the multi-task results in Table 5 and 6, we concatenate all predicted tasks with the redacted image. We use the 'bandstop' spectral redaction and set a block size of $b = 32$. For all experiments, we use the CRM-R module for 20 steps to perform regeneration. From Equations 1 and 2 of the main paper, we set $\gamma = 1$, $\gamma_1 = 1$ and $\gamma_2 = 0.3$ as the loss weights for the DejaVu loss.

D. Visualization

In this section, we provide further visual results as an extension to those in the main paper. We show qualitative comparisons in semantic segmentation on Cityscapes, panoptic segmentation on COCO, and surface normal estimation on NYUD-v2, with and without the DejaVu loss.

D.1. Qualitative results on Cityscapes

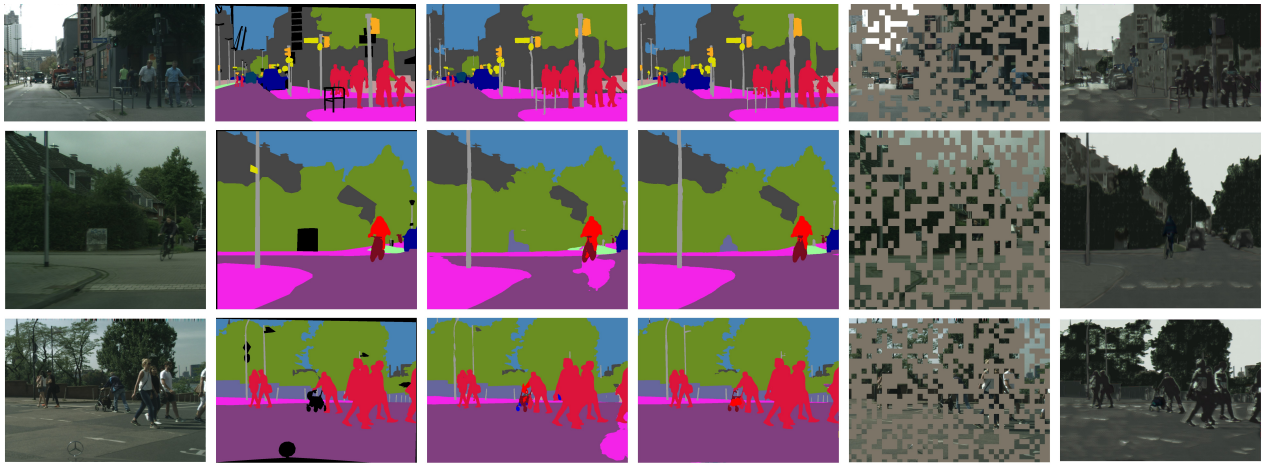
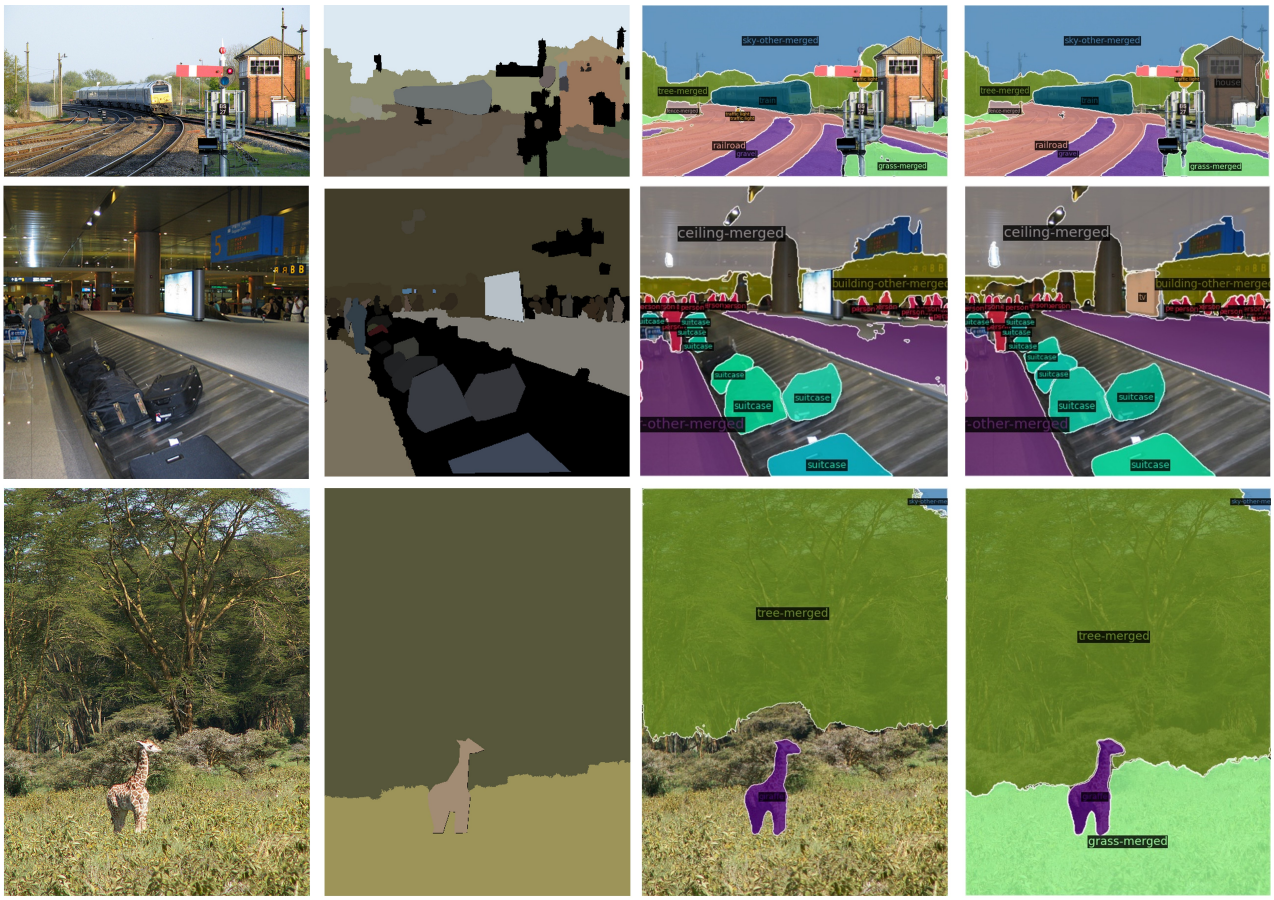


Figure D.1. Cityscapes benchmark Visualization for semantic segmentation task. (a) input image, (b) ground truth (c) baseline prediction, (d) our enhanced prediction, (e) redacted input, (f) regenerated image. Here the regenerated images are produced from predictions and redacted inputs. We use OCR as baseline.

We provide more results on Cityscapes for qualitative comparison with the base model. These are an extension to the ones shown in Figure 6 of the main paper. The base model used is HRNet18-OCR, which is evaluated in Table 1 of the main paper. We compare the final prediction with and without adding our DejaVu loss term. As observed from the images, the baseline has incorrect segmentation blobs in categories such as *sidewalk* and *road*. However, the results obtained by training using DejaVu loss show clear visual improvement, both in terms of overall accuracy and structure of the output.

D.2. Qualitative results on COCO



(a) Input **(b) Ground Truth** **(c) Baseline Prediction** **(d) Our Prediction**

Figure D.2. COCO benchmark Visualization for panoptic segmentation. (a) input image, (b) ground truth (c) baseline prediction, (d) our enhanced prediction, (e) redacted input, (f) regenerated image.

Figure D.2 shows more examples on COCO benchmark for panoptic segmentation, as an extension to the ones shown in Figure 6 of the main paper. The base model used is Mask2former with a Swin-T encoder, which was analyzed in Table 2 of the main paper. As observed from the images, it misses several object masks such as the *house* in the first row, the *TV* and *suitcases* in second row and *grass* in third row. However, the predictions obtained by adding the DejaVu loss not only improve the overall segmentation correctness, but also produce better aligned output boundaries.

D.3. Qualitative results on NYUD-v2

Figure D.3. NYUD-V2 Visualization benchmark Visualization for surface normal estimation. (a) input image, (b) ground truth (c) baseline prediction, (d) our enhanced prediction, (e) redacted input, (f) regenerated image. Here the regenerated images are produced from predictions and redacted inputs. We use SegNet-XTC as baseline.

We further visualize the results of surface normal estimation on NYUD-V2 benchmark, as an extension to the ones shown in Figure 6 of the main paper. We use the SegNet-XTC network as our base architecture. As noticed from the predictions in Figure D.3(c) produced by the baseline, the boundary of estimations are blurry and less consistent. However, our predictions obtained by using the DejaVu loss, as observed in Figure D.3(d), produce sharper boundaries and better align surface normal vectors on the same plane.

References

[1] Shubhankar Borse, Hong Cai, Yizhe Zhang, and Fatih Porikli. Hs3: Learning with proper task complexity in hierarchically supervised semantic segmentation. *arXiv preprint arXiv:2111.02333*, 2021. 3

[2] Shubhankar Borse, Ying Wang, Yizhe Zhang, and Fatih Porikli. Inverseform: A loss function for structured boundary-aware segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5901–5911, 2021. 3

[3] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, Zheng Zhang, Dazhi Cheng, Chenchen Zhu, Tianheng Cheng, Qijie Zhao, Buyu Li, Xin Lu, Rui Zhu, Yue Wu, Jifeng Dai, Jingdong Wang, Jianping Shi, Wanli Ouyang, Chen Change Loy, and Dahua Lin. MMDetection: Open mmlab detection toolbox and benchmark. *arXiv:1906.07155*, 2019. 4

[4] MMSegmentation Contributors. MMSegmentation: OpenMMLab semantic segmentation toolbox and benchmark. https://github.com/open-mmlab/mmsegmentation, 2020. 4

[5] Clément Godard, Oisin Mac Aodha, Michael Firman, and Gabriel J Brostow. Digging into self-supervised monocular depth estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3828–3838, 2019. 4

[6] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16000–16009, 2022. 2, 3

[7] Wei-Hong Li, Xialei Liu, and Hakan Bilen. Learning multiple dense prediction tasks from partially annotated data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18879–18889, 2022. 4

[8] Ke Sun, Yang Zhao, Borui Jiang, Tianheng Cheng, Bin Xiao, Dong Liu, Yadong Mu, Xinggang Wang, Wenyu Liu, and Jingdong Wang. High-resolution representations for labeling pixels and regions. *PAMI*, 2019. 3

[9] Andrew Tao, Karan Sapra, and Bryan Catanzaro. Hierarchical multi-scale attention for semantic segmentation. *arXiv:2005.10821*, 2020. 3

[10] Zian Wang, David Acuna, Huan Ling, Amlan Kar, and Sanja Fidler. Object instance annotation with deep extreme level set evolution. In *CVPR*, 2019. 3