Panoptic-DeepLab: 
A Simple, Strong, and Fast Baseline for Bottom-Up Panoptic Segmentation

Bowen Cheng¹,², Maxwell D. Collins², Yukun Zhu², Ting Liu², Thomas S. Huang¹, Hartwig Adam², Liang-Chieh Chen²

¹UIUC  ²Google Research

Abstract
In this work, we introduce Panoptic-DeepLab, a simple, strong, and fast system for panoptic segmentation, aiming to establish a solid baseline for bottom-up methods that can achieve comparable performance of two-stage methods while yielding fast inference speed. In particular, Panoptic-DeepLab adopts the dual-ASPP and dual-decoder structures specific to semantic, and instance segmentation, respectively. The semantic segmentation branch is the same as the typical design of any semantic segmentation model (e.g., DeepLab), while the instance segmentation branch is class-agnostic, involving a simple instance center regression. As a result, our single Panoptic-DeepLab simultaneously ranks first at all three Cityscapes benchmarks, setting the new state-of-art of 84.2% mIoU, 39.0% AP, and 65.5% PQ on test set. Additionally, equipped with MobileNetV3, Panoptic-DeepLab runs nearly in real-time with a single 1025 × 2049 image (15.8 frames per second), while achieving a competitive performance on Cityscapes (54.1 PQ% on test set). On Mapillary Vistas test set, our ensemble of six models attains 42.7% PQ, outperforming the challenge winner in 2018 by a healthy margin of 1.5%. Finally, our Panoptic-DeepLab also performs on par with several top-down approaches on the challenging COCO dataset. For the first time, we demonstrate a bottom-up approach could deliver state-of-the-art results on panoptic segmentation.

1. Introduction
Panoptic segmentation, unifying semantic segmentation and instance segmentation, has received a lot of attention thanks to the recently proposed panoptic quality metric [34] and associated recognition challenges [46, 16, 53]. The goal of panoptic segmentation is to assign a unique value, encoding both semantic label and instance id, to every pixel in an image. It requires identifying the class and extent of each individual ‘thing’ in the image, and labelling all pixels that belong to each ‘stuff’ class.

The task of panoptic segmentation introduces challenges that preceding methods are unsuited to solve. Models typically used in the separate instance and semantic segmentation literature have diverged, and fundamentally different approaches dominate in each setting. For panoptic segmentation, the top-down methods [74, 33, 40, 43, 60], attaching another semantic segmentation branch to Mask R-CNN [25], generate overlapping instance masks as well as duplicate pixel-wise semantic predictions. To settle the conflict, the commonly employed heuristic resolves overlapping instance masks by their predicted confidence scores [34], or even by the pairwise relationship between categories [43] (e.g., ties should be always in front of person). Additionally, the discrepancy between semantic and instance segmentation results are sorted out by favoring the instance predictions. Though effective, it may be hard to implement the hand-crafted heuristics in a fast and parallel fashion. Another effective way is to develop advanced modules to fuse semantic and instance segmentation results [43, 40, 74]. However, these top-down methods are usually
slow in speed, resulted from the multiple sequential processes in the pipeline.

On the other hand, bottom-up methods naturally resolve the conflict by predicting non-overlapping segments. Only few works [75, 22] adopt the bottom-up approach, which typically starts with a semantic segmentation prediction followed by grouping operations to generate instance masks. Tackling panoptic segmentation in such a sequential order allows a simple and fast scheme, such as majority vote [75], to merge semantic and instance segmentation results. Although obtaining promising fast inference speed, bottom-up approaches still demonstrate inferior performance compared to top-down ones prevailing in public benchmarks [46, 16, 53].

The difficulties faced by top-down methods, and the dearth of previous investigations into complementary approaches motivate us to establish a simple, strong, and fast bottom-up baseline for panoptic segmentation. Our proposed Panoptic-DeepLab (Fig. 1) requires only three loss functions during training, and introduces extra marginal parameters as well as additional slight computation overhead when building on top of a modern semantic segmentation model. The design of the proposed Panoptic-DeepLab is conceptually simple, adopting dual-ASPP and dual-decoder modules specific to semantic segmentation and instance segmentation, respectively. The semantic segmentation branch follows the typical design of any semantic segmentation model (e.g., DeepLab [11]), while the instance segmentation branch involves a simple instance center regression [4, 30], where the model learns to predict instance centers as well as the offset from each pixel to its corresponding center, resulting in an extremely simple grouping operation by assigning pixels to their closest predicted center. Additionally, with fast GPU implementation of the merging operation, Panoptic-DeepLab delivers near real-time end-to-end panoptic segmentation prediction.

We conduct experiments on several popular panoptic segmentation datasets. On Cityscapes test set [16], a single Panoptic-DeepLab model (without fine-tuning on different tasks) achieves state-of-the-art performance of 65.5% PQ, 39.0% AP, and 84.2% mIoU, simultaneously ranking first on all three Cityscapes tasks when comparing with published works. On Mapillary Vistas [53], our best single model attains 40.6% PQ on val set, while employing an ensemble of 6 models reaches a performance of 42.2% PQ on val set and 42.7% PQ on test set, outperforming the winner of Mapillary Vistas Panoptic Segmentation Challenge in 2018 by a healthy margin of 1.5% PQ. For the first time, we show a bottom-up approach could deliver state-of-the-art panoptic segmentation results on both Cityscapes and Mapillary Vistas. On COCO [46] test-dev set, our Panoptic-DeepLab also demonstrates state-of-the-art results, performing on par with several top-down approaches. Finally, we provide extensive experimental results and disclose every detail in our system. We hope our Panoptic-DeepLab could serve as a solid baseline to facilitate the research on panoptic segmentation, especially from the bottom-up perspective.

2. Related Works

We categorize current panoptic segmentation methods [34] into two groups: top-down and bottom-up approaches. Top-down: Most state-of-the-art methods tackle panoptic segmentation from the top-down or proposal-based perspective. These methods are often referred to as two-stage methods because they require an additional stage to generate proposals. Specifically, Mask R-CNN [25] is commonly deployed to extract overlapping instances, followed by some post-processing methods to resolve mask overlaps. The remaining regions are then filled by a light-weight stuff segmentation branch. For example, TASCNet [40] learns a binary mask to enforce the consistency between ‘thing’ and ‘stuff’ predictions. Liu et al. [52] propose the Spatial Ranking module to resolve the overlapping instance masks. AUNet [43] introduces attention modules to guide the fusion between ‘thing’ and ‘stuff’ segmentation. Panoptic FPN [33] endows Mask R-CNN [25] with a semantic segmentation branch. UPSNet [74] develops a parameter-free panoptic head which resolves the conflicts in ‘thing’ and ‘stuff’ fusion by predicting an extra unknown class. Porzi et al. [60] integrate the multi-scale features from FPN [45] with a light-weight DeepLab-inspired module [9]. AdaptIS [66] generates instance masks with point proposals. Bottom-up: On the other hand, there are few bottom-up or proposal-free methods for panoptic segmentation. These works typically get the semantic segmentation prediction before detecting instances by grouping ‘thing’ pixels into clusters. The first bottom-up approach, DeepLab [75], adopts bounding box corners as well as object centers for class-agnostic instance segmentation, coupled with DeepLab semantic segmentation outputs [8, 10]. Recently, SSAP [22] proposes to group pixels based on a pixel-pair affinity pyramid [51] with an efficient graph partition method [31]. Unfortunately, given its simplicity (i.e., a single pass of the system for prediction), bottom-up approaches perform inferiorly to top-down methods at almost all public benchmarks. In this work, we aim to push the envelope of bottom-up approaches. We note that there are several instance segmentation works [77, 68, 76, 2, 48, 35, 56, 20, 18, 44, 37, 30, 51, 54, 6], which could be potentially extended to bottom-up panoptic segmentation. Additionally, our method bears a similarity to Hough-Voting-based methods [4, 39, 21, 5] and recent works by Kendall et al. [30], Uhrig et al. [69] and Neven et al. [54] in the sense that our class-agnostic instance segmentation is obtained by regressing foreground pixels to their centers. However, our method
Our Panoptic-DeepLab adopts dual-context and dual-decoder modules for semantic segmentation and instance segmentation predictions. We apply atrous convolution in the last block of a network backbone to extract denser feature map. The Atrous Spatial Pyramid Pooling (ASPP) is employed in the context module as well as a light-weight decoder module consisting of a single convolution during each upsampling stage. The instance segmentation prediction is obtained by predicting the object centers and regressing every foreground pixel (i.e., pixels with predicted ‘thing’ class) to their corresponding center. The predicted semantic segmentation and class-agnostic instance segmentation are then fused to generate the final panoptic segmentation result by the ”majority vote” proposed by DeeperLab.

is even simpler than theirs: we directly predict the instance center locations and group pixels to their closest predicted centers. As a result, our method does not require the clustering method OPTICS [1] used in [30], or the advanced clustering loss function proposed in [54]. Finally, our model employs the parallel multi-head prediction framework similar to [68, 36, 55].

Keypoint representation: Recently, keypoint representations have been used for instance segmentation and object detection. Newell et al. [56] group pixels by embedding vectors. PersonLab [58] generates person segmentation masks and groups them into instances by learning offset to their detected keypoints. CornerNet [38] detects objects by predicting paired corners and group corners based on [56]. ExtremeNet [79] groups ‘extreme points’ [57] according to the relation to a center point. Zhou et al. [78] and Duan et al. [19] exploit instance centers for object detection. Following the same direction, we represent each instance by its center and take a step further by showing that such a simple representation is able to achieve state-of-the-art panoptic segmentation results on several challenging datasets. Different from keypoint-based detection, our Panoptic-DeepLab only requires class-agnostic object center prediction.

3. Panoptic-DeepLab

As shown in Fig. 2, our proposed Panoptic-DeepLab is deployed in a bottom-up and single-shot manner.
groundtruth instance centers are encoded by a 2-D Gaussian with standard deviation of 8 pixels [67]. In particular, we adopt the Mean Squared Error (MSE) loss to minimize the distance between predicted heatmaps and 2D Gaussian-encoded groundtruth heatmaps. We use $L_1$ loss for the offset prediction, which is only activated at pixels belonging to object instances. During inference, predicted foreground pixels (obtained by filtering out background ‘stuff’ regions from semantic segmentation prediction) are grouped to their closest predicted mass center, forming our class-agnostic instance segmentation results, as detailed below.

### 3.2. Panoptic Segmentation

During inference, we use an extremely simple grouping operation to obtain instance masks, and a highly efficient majority voting algorithm to merge instance and semantic segmentation into final panoptic segmentation.

**Simple instance representation:** We simply represent each object by its center of mass, $\{C_n : (i_n, j_n)\}$. To obtain the center point prediction, we first perform a keypoint-based non-maximum suppression (NMS) on the instance center heatmap prediction, essentially equivalent to applying max pooling on the heatmap prediction and keeping locations whose values do not change before and after max pooling. Finally, a hard threshold is used to filter out predictions with low confidence, and only locations with top-k highest confidence scores are kept. In experiments, we use max-pooling with kernel size 7, threshold 0.1, and $k = 200$.

**Simple instance grouping:** To obtain the instance id for each pixel, we use a simple instance center regression. For example, consider a predicted ‘thing’ pixel at location $(i, j)$, we predict an offset vector $O(i, j)$ to its instance center. $O(i, j)$ is a vector with two elements, representing the offset in horizontal and vertical directions, respectively. The instance id for the pixel is thus the index of the closest instance center after moving the pixel location $(i, j)$ by the offset $O(i, j)$. That is,

$$\hat{k}_{i,j} = \arg\min_k ||C_k - ((i, j) + O(i, j))||^2$$

where $\hat{k}_{i,j}$ is the predicted instance id for pixel at $(i, j)$.

We use semantic segmentation prediction to filter out ‘stuff’ pixels whose instance id are always set to 0.

**Efficient merging:** Given the predicted semantic segmentation and class-agnostic instance segmentation results, we adopt a fast and parallelizable method to merge the results, following the “majority vote” principle proposed in DeeperLab [75]. In particular, the semantic label of a predicted instance mask is inferred by the majority vote of the corresponding predicted semantic labels. This operation is essentially accumulating the class label histograms, and thus is efficiently implemented in GPU, which takes only 3 ms when operating on a 1025 $\times$ 2049 input.

### 3.3. Instance Segmentation

Panoptic-DeepLab can also generate instance segmentation predictions as a by-product. To properly evaluate the instance segmentation results, one needs to associate a confidence score with each predicted instance mask. Previous bottom-up instance segmentation methods use some heuristics to obtain the confidence scores. For example, DWT [2] and SSAP [22] use an average of semantic segmentation scores for some easy classes and use random scores for other harder classes. Additionally, they remove masks whose areas are below a certain threshold for each class. On the other hand, our Panoptic-DeepLab does not adopt any heuristic or post processing for instance segmentation. Motivated by YOLO [62], we compute the class-specific confidence score for each instance mask as

$$\text{Score}(\text{Objectness}) \times \text{Score}(\text{Class})$$

where $\text{Score}(\text{Objectness})$ is unnormalized objectness score obtained from the class-agnostic center point heatmap, and $\text{Score}(\text{Class})$ is obtained from the average of semantic segmentation predictions within the predicted mask region.

### 4. Experiments

**Cityscapes [16]:** The dataset consists of 2975, 500, and 1525 traffic-related images for training, validation, and testing, respectively. It contains 8 ‘thing’ and 11 ‘stuff’ classes.

**Mapillary Vistas [53]:** A large-scale traffic-related dataset, containing 18K, 2K, and 5K images for training, validation and testing, respectively. It contains 37 ‘thing’ classes and 28 ‘stuff’ classes in a variety of image resolutions, ranging from $1024 \times 768$ to more than $4000 \times 6000$

**COCO [46]:** There are 118K, 5K, and 20K images for training, validation, and testing, respectively. The dataset consists of 80 ‘thing’ and 53 ‘stuff’ classes.

**Experimental setup:** We report mean IoU, average precision (AP), and panoptic quality (PQ) to evaluate the semantic, instance, and panoptic segmentation results.

All our models are trained using TensorFlow on 32 TPUs. We adopt a similar training protocol as in [11]. In particular, we use the ‘poly’ learning rate policy [50] with an initial learning rate of 0.001, fine-tune the batch normalization [29] parameters, perform random scale data augmentation during training, and optimize with Adam [32] without weight decay. On Cityscapes, our best setting is obtained by training with whole image (i.e., crop size equal to $1025 \times 2049$) with batch size 32. On Mapillary Vistas, we resize the images to 2177 pixels at the longest side to handle the large input variations, and randomly crop $1025 \times 1025$ patches during training with batch size 64. On COCO, we resize the images to 1025 pixels at the longest side and train our models with crop size

<table>
<thead>
<tr>
<th>Method</th>
<th>Extra Data</th>
<th>Flip</th>
<th>MS</th>
<th>PQ (%)</th>
<th>AP (%)</th>
<th>mIoU (%)</th>
<th>Params (M)</th>
<th>M-Adds (B)</th>
</tr>
</thead>
<tbody>
<tr>
<td>w/o Extra Data</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TASCNet [40]</td>
<td></td>
<td></td>
<td></td>
<td>55.9</td>
<td>-</td>
<td>-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Panoptic FPN [33]</td>
<td></td>
<td></td>
<td></td>
<td>58.1</td>
<td>33.0</td>
<td>75.7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AUNet [43]</td>
<td></td>
<td></td>
<td></td>
<td>59.0</td>
<td>34.4</td>
<td>75.6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>UPSNet [74]</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>59.3</td>
<td>33.3</td>
<td>75.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Seamless [60]</td>
<td>✓</td>
<td>-</td>
<td></td>
<td>60.3</td>
<td>33.6</td>
<td>77.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AdapIS [66]</td>
<td>✓</td>
<td>-</td>
<td></td>
<td>62.0</td>
<td>36.3</td>
<td>79.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DeeperLab [75]</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>56.5</td>
<td>-</td>
<td>-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Panoptic-DeepLab</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>63.0</td>
<td>35.3</td>
<td>80.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Panoptic-DeepLab</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>63.4</td>
<td>36.1</td>
<td>80.9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Panoptic-DeepLab</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>64.1</td>
<td>38.5</td>
<td>81.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>w/ Extra Data</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TASCNet [40]</td>
<td>COCO</td>
<td></td>
<td></td>
<td>59.3</td>
<td>37.6</td>
<td>78.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TASCNet [40]</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>60.4</td>
<td>39.1</td>
<td>78.7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>UPSNet [74]</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>60.5</td>
<td>37.8</td>
<td>77.8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>UPSNet [74]</td>
<td>COCO</td>
<td></td>
<td></td>
<td>61.8</td>
<td>39.0</td>
<td>79.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Seamless [60]</td>
<td>COCO</td>
<td></td>
<td></td>
<td>65.0</td>
<td>-</td>
<td>80.7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Panoptic-DeepLab</td>
<td>MV</td>
<td></td>
<td></td>
<td>65.3</td>
<td>38.8</td>
<td>82.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Panoptic-DeepLab</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>65.5</td>
<td>39.4</td>
<td>82.6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Panoptic-DeepLab</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>67.0</td>
<td>42.5</td>
<td>83.1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2. Cityscapes val set. Flip: Adding left-right flipped inputs. MS: Multiscale inputs. MV: Mapillary Vistas.

1025 × 1025 with batch size 64. We set training iterations to 60K, 150K, and 200K for Cityscapes, Mapillary Vistas, and COCO, respectively. During evaluation, due to the sensitivity of PQ [74, 40, 60], we re-assign to ‘VOID’ label all ‘stuff’ segments whose areas are smaller than a threshold. The thresholds on Cityscapes, Mapillary Vistas, and COCO are 2048, 4096, and 4096, respectively. Additionally, we adopt multi-scale inference (scales equal to {0.5, 0.75, 1, 1.25, 1.5, 1.75, 2}) for Cityscapes and Mapillary Vistas and {0.5, 0.75, 1, 1.25, 1.5} for COCO and left-right flipped inputs, to further improve the performance. For all the reported results, unless specified, Xception-71 [15, 61, 11] is employed as the backbone.

Panoptic-DeepLab is trained with three loss functions: weighted bootstrapped cross entropy loss for semantic segmentation head ($L_{sem}$) [75]; MSE loss for center heatmap head ($L_{heatmap}$) [67]; and 1L loss for center offset head ($L_{offset}$) [58]. The final loss $L$ is computed as follows.

$$L = \lambda_{sem}L_{sem} + \lambda_{heatmap}L_{heatmap} + \lambda_{offset}L_{offset}$$

Specifically, we set $\lambda_{sem} = 3$ for pixels belonging to instances with an area smaller than 64 × 64 and $\lambda_{sem} = 1$ everywhere else, following DeeperLab [75]. To make sure the losses are in the similar magnitude, we set $\lambda_{heatmap} = 200$ and $\lambda_{offset} = 0.01$.

4.1. Ablation Studies

We conduct ablation studies on Cityscapes validation set, as shown in Tab. 1. Replacing SGD momentum optimizer with Adam optimizer yields 0.7% PQ improvement. Instead of using the sigmoid cross entropy loss for training the heatmap (i.e., instance center prediction), it brings 0.8% PQ improvement by applying the Mean Squared Error (MSE) loss to minimize the distance between the predicted heatmap and the 2D Gaussian-encoded groundtruth heatmap. It is more effective to adopt both dual-decoder and
dual-ASPP, which gives us 0.7% PQ improvement while maintaining similar AP and mIoU. Employing a large crop size $1025 \times 2049$ (instead of $513 \times 1025$) during training further improves the AP and mIoU by 0.6% and 0.9% respectively. Finally, increasing the feature channels from 128 to 256 in the semantic segmentation branch achieves our best result of 63.0% PQ, 35.3% AP, and 80.5% mIoU.

Multi-task learning: For reference, we train a Semantic-DeepLab under the same setting as the best Panoptic-DeepLab (last row of Tab. 1), showing that multi-task learning does not bring extra gain to mIoU. Note that Panoptic-DeepLab adds marginal parameters and small computation overhead over Semantic-DeepLab.

### 4.2. Cityscapes

Val set: In Tab. 2, we report our Cityscapes validation set results. When using only Cityscapes fine annotations, our best Panoptic-DeepLab, with multi-scale inputs and left-right flips, outperforms the best bottom-up approach, SSAP, by 3.0% PQ and 1.2% AP, and is better than the best proposal-based approach, AdaptIS, by 2.1% PQ, 2.2% AP, and 2.3% mIoU. When using extra data, our best Panoptic-DeepLab outperforms UPSNet by 5.2% PQ, 3.5% AP, and 3.9% mIoU, and Seamless by 2.0% PQ and 2.4% mIoU. Note that we do not exploit any other data, such as COCO, Cityscapes coarse annotations, depth, or video.

Test set: On the test set, we additionally employ the trick proposed in [11] that applies atrous convolution in the last two blocks within the backbone, with rate 2 and 4 respectively, during inference. This trick brings an extra 0.4% AP and 0.2% mIoU on val set but no improvement over PQ. We do not use this trick for the Mapillary Vistas Challenge. As shown in Tab. 3, our single unified Panoptic-DeepLab achieves state-of-the-art results, ranking first at all three Cityscapes tasks, when comparing with published works. Our model ranks second in the instance segmentation track when also taking into account unpublished entries.

### 4.3. Mapillary Vistas

Val set: In Tab. 4, we report Mapillary Vistas val set results. Our best single Panoptic-DeepLab model, with multi-scale inputs and left-right flips, outperforms the bottom-up approach, Deeplab, by 8.3% PQ, and the top-down approach, Seamless, by 2.6% PQ. In Tab. 5, we report our results with three families of network backbones. We observe that naive HRNet-W48 slightly under-performs Xception-71. Due to the diverse image resolutions in Mapillary Vistas, we found it important to enrich the context information as well as to keep high-resolution features. Therefore, we propose a simple modification for HRNet [70] and Auto-DeepLab [47]. For modified HRNet, called HRNet+, we keep its ImageNet-pretrained head and further attach dual-ASPP and dual-decoder modules. For modified Auto-DeepLab, called Auto-DeepLab+, we remove the stride in the original 1/32 branch (which improves PQ by 1%). To summarize, using Xception-71 strikes the best accuracy and speed trade-off, while HRNet-W48+ achieves the best PQ of 40.6%. Finally, our ensemble of six models attains a 42.2% PQ, 18.2% AP, and 58.7% mIoU.

Test set: Tab. 6 summarizes our Mapillary Vistas test set results along with other top-performing methods. Our entry [12] with an ensemble of six models attains a performance of 42.7% PQ, outperforming the winner of Mapillary Vistas Panoptic Segmentation Challenge in 2018 by 1.5% PQ.
Table 7. COCO val set. Flip: Adding left-right flipped inputs. MS: Multiscale inputs.

<table>
<thead>
<tr>
<th>Method</th>
<th>Backbone</th>
<th>Flip</th>
<th>MS</th>
<th>PQ (%)</th>
<th>PQ50 (%)</th>
<th>PQ80 (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Panoptic-FPN [33]</td>
<td>ResNet-101</td>
<td></td>
<td></td>
<td>40.3</td>
<td>47.5</td>
<td>29.5</td>
</tr>
<tr>
<td>AdaptIS [66]</td>
<td>ResNeXt-101 [73]</td>
<td>✓</td>
<td></td>
<td>42.3</td>
<td>49.2</td>
<td>31.8</td>
</tr>
<tr>
<td>UPSNet [74]</td>
<td>ResNet-50</td>
<td></td>
<td></td>
<td>42.5</td>
<td>48.5</td>
<td>33.4</td>
</tr>
<tr>
<td>Detectron [71]</td>
<td>ResNet-101</td>
<td></td>
<td></td>
<td>43.0</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>UPSNet [74]</td>
<td>ResNet-50</td>
<td></td>
<td></td>
<td>43.2</td>
<td>49.1</td>
<td>34.1</td>
</tr>
<tr>
<td>Deeplab [75]</td>
<td>Xception-71</td>
<td>✓</td>
<td>✓</td>
<td>33.8</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SSAP [22]</td>
<td>ResNet-101</td>
<td>✓</td>
<td>✓</td>
<td>36.5</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 8. COCO test-dev set. Flip: Adding left-right flipped inputs. MS: Multiscale inputs.

<table>
<thead>
<tr>
<th>Method</th>
<th>Backbone</th>
<th>Flip</th>
<th>MS</th>
<th>PQ (%)</th>
<th>PQ50 (%)</th>
<th>PQ80 (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TASCNet [40]</td>
<td>ResNet-50</td>
<td></td>
<td></td>
<td>40.7</td>
<td>47.0</td>
<td>31.0</td>
</tr>
<tr>
<td>Panoptic-FPN [33]</td>
<td>ResNet-101</td>
<td></td>
<td></td>
<td>40.9</td>
<td>48.3</td>
<td>29.7</td>
</tr>
<tr>
<td>AdaptIS [66]</td>
<td>ResNeXt-101</td>
<td>✓</td>
<td></td>
<td>42.8</td>
<td>53.2</td>
<td>36.7</td>
</tr>
<tr>
<td>AUNet [43]</td>
<td>ResNeXt-152</td>
<td></td>
<td></td>
<td>46.5</td>
<td>55.8</td>
<td>32.5</td>
</tr>
<tr>
<td>UPSNet [74]</td>
<td>DCN-101 [17]</td>
<td>✓</td>
<td>✓</td>
<td>46.6</td>
<td>53.2</td>
<td>36.7</td>
</tr>
<tr>
<td>Deeplab [75]</td>
<td>Xception-71</td>
<td>✓</td>
<td>✓</td>
<td>34.3</td>
<td>37.5</td>
<td>29.6</td>
</tr>
<tr>
<td>SSAP [22]</td>
<td>ResNet-101</td>
<td>✓</td>
<td>✓</td>
<td>36.9</td>
<td>40.1</td>
<td>32.0</td>
</tr>
</tbody>
</table>

4.4. COCO

Val set: In Tab. 7, we report COCO val set result. With a single scale inference, our Panoptic-DeepLab outperforms the previous best bottom-up SSAP by 3.2% PQ and Deeplab [75] by 5.9% PQ. With multi-scale inference and horizontal flip, Panoptic-DeepLab achieves 41.2% PQ, setting a new state-of-the-art performance for bottom-up methods, and performing comparably with top-down methods.

Test-dev set: In Tab. 8, we report COCO test-dev set result. Our Panoptic-DeepLab is 4.5% PQ better than the previous best bottom-up SSAP on COCO and our 41.4% PQ is comparable to most top-down methods without using heavier backbone [73] or deformable convolution [17].

4.5. Runtime

In Tab. 9, we report the end-to-end runtime (i.e., inference time from an input image to final panoptic segmentation, including all operations such as merging semantic and instance segmentation) of Panoptic-DeepLab with three different network backbones (MobileNetV3 [27], ResNet-50 [26], and Xception-71 [15, 61]) on all three datasets. The inference speed is measured on a Tesla V100-SXM2 GPU with batch size of one. We further plot the speed-accuracy trade-off curve in Fig. 3. Our Panoptic-DeepLab achieves the best trade-off across all three datasets.

4.6. Discussion

Herein, we list a few interesting aspects in the hope of inspiring future works on bottom-up panoptic segmentation.

Scale variation: Fig. 4 shows visualization of Panoptic-DeepLab. In particular, the cross road (in last 2 rows), with
a large scale variation, is segmented into multiple small instances. On the other hand, top-down methods handle scale variation to some extent by the ROI Pooling [23] or ROIAlign [25] operations which normalize regional features to a canonical scale [24, 63]. Additionally, incorporating scale-aware information to feature pyramid [48] or image pyramid [65] may improve the performance of bottom-up methods.

**PQ\textsuperscript{Thing} vs. PQ\textsuperscript{Stuff}:** As shown in Tab. 6 and Tab. 8, Panoptic-DeepLab has higher PQ\textsuperscript{Stuff} but lower PQ\textsuperscript{Thing} when compared with other top-down approaches which better handle instances of large scale variation as discussed above. Combining the best from both bottom-up and top-down approaches is thus interesting to explore but beyond the scope of current work.

**Panoptic vs. instance annotations:** Most bottom-up panoptic segmentation methods only exploit the panoptic annotations. We notice there are two types of annotations in the COCO dataset, panoptic annotations and instance annotations. The former do not allow overlapping masks (thus creating occlusions among masks), while the latter allows overlaps, which might make the training target easier to optimize, similar to amodal segmentation [81, 41].

**End-to-end training:** Current bottom-up panoptic segmentation methods still require some post-processing steps to obtain the final panoptic segmentation, which may make it hard to end-to-end train the whole system.

5. Conclusion

We have presented Panoptic-DeepLab, a simple, strong, and fast baseline for bottom-up panoptic segmentation. Panoptic-DeepLab is simple in design, requiring only three loss functions during training and adds marginal parameters to a modern semantic segmentation model. Panoptic-DeepLab is the first bottom-up and single-shot panoptic segmentation model that attains state-of-the-art performance on several public benchmarks, and delivers near real-time end-to-end inference speed. We hope our simple and effective model could establish a solid baseline and further benefit the research community.
References


[76] Ziyu Zhang, Sanja Fidler, and Raquel Urtasun. Instance-level segmentation for autonomous driving with deep densely connected mrfs. In \textit{CVPR}, 2016. 2


