DeeperLab: Single-Shot Image Parser

Tien-Ju Yang[1], Maxwell D. Collins[2], Yukun Zhu[2], Jyh-Jing Hwang[2,3], Ting Liu[2],
Xiao Zhang[2], Vivienne Sze[1], George Papandreou[2], Liang-Chieh Chen[2]
MIT[1], Google Inc.[2], UC Berkeley [3]

Abstract

We present a single-shot, bottom-up approach for whole image parsing. Whole image parsing, also known as Panoptic Segmentation, generalizes the tasks of semantic segmentation for 'stuff' classes and instance segmentation for 'thing' classes, assigning both semantic and instance labels to every pixel in an image. Recent approaches to whole image parsing typically employ separate standalone modules for the constituent semantic and instance segmentation tasks and require multiple passes of inference. Instead, the proposed DeeperLab image parser performs whole image parsing with a significantly simpler, fully convolutional approach that jointly addresses the semantic and instance segmentation tasks in a single-shot manner, resulting in a streamlined system that better lends itself to fast processing. For quantitative evaluation, we use both the instance-based Panoptic Quality (PQ) metric and the proposed region-based Parsing Covering (PC) metric, which better captures the image parsing quality on 'stuff' classes and larger object instances. We report experimental results on the challenging Mapillary Vistas dataset, in which our single model achieves 31.95% (val) / 31.6% PQ (test) and 55.26% PC (val) with 3 frames per second (fps) on GPU or near real-time speed (22.6 fps on GPU) with reduced accuracy.

1. Introduction

This paper addresses the problem of efficient whole image parsing (in short, image parsing) [1], also known as *Panoptic Segmentation* [2]. Image parsing is a long-lasting unsolved problem in computer vision and a basic component of many applications, such as autonomous driving. The high difficulty lies in the fact that image parsing unifies two challenging tasks, semantic segmentation and instance segmentation. Semantic segmentation focuses on partitioning the whole image into multiple semantically meaningful regions, regardless of whether the semantic class is countable (a 'thing' class) or uncountable (a 'stuff' class). In contrast, instance segmentation only handles the region related to the 'thing' classes but requires telling different instances apart.

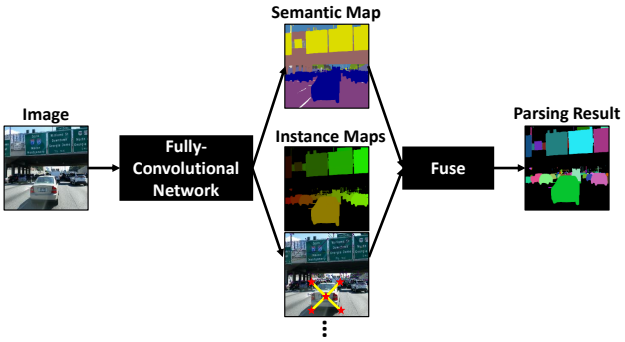

Figure 1: The proposed single-shot, bottom-up image parser, DeeperLab. The *per-pixel* semantic and instance predictions are generated using a *single* pass of a fully-convolutional network. These predictions are then fused into the final image parsing result by a fast algorithm.

As the combination of them, image parsing attempts to segment the whole image for both the 'thing' and 'stuff' classes and separate different 'thing' instances.

Although there have been a few works trying to solve the image parsing problem, efficiency is usually not taken into account. Efficiency is important for practical deployment. For example, Kirillov *et al*. [2] report excellent parsing results, but the computational cost can be high due to the multiple passes of several complicated networks. The computation can also be more intense when high-resolution images are used as the input; *e.g*., the Mapillary Vistas dataset [3] contains images with a resolution of up to 4000×6000.

In this work, we aim to design an image parser that achieves a good balance between accuracy and efficiency. We propose a single-shot, bottom-up image parser, called *DeeperLab*. As shown in Fig. 1, DeeperLab generates the *per-pixel* semantic and instance predictions using a *single* pass of a fully-convolutional network. These predictions are then fused into the final image parsing result by a fast algorithm. The runtime of DeeperLab is nearly independent of the number of detected object instances, which makes DeeperLab favorable for image parsing of complex scenes.

For quantitative evaluation, we argue that the recently

proposed instance-based Panoptic Quality (PQ) metric [2] often places disproportionate emphasis on small instance parsing, as well as on 'thing' over 'stuff' classes. To remedy these effects, we propose an alternative region-based *Parsing Covering (PC)* metric*, which adapts the Covering metric [4], previously used for class-agnostics segmentation quality evaluation, to the task of image parsing. We report quantitative results with both PQ and PC metrics.

Our main contributions are summarized below.

- We propose several neural network design strategies for efficient image parsers, especially reducing memory footprint for high-resolution inputs. These innovations include extensively applying depthwise separable convolution, using a shared decoder output with simple two-layer prediction heads, enlarging kernel sizes instead of making the network deeper, employing space-to-depth and depth-to-space rather than upsampling, and performing hard data mining. Detailed ablation studies are also provided to show the impact of these strategies in practice.

- We propose an efficient single-shot, bottom-up image parser, DeeperLab, based on the proposed design strategies. For example, on the Mapillary Vistas dataset, our Xception-71 [5, 6, 7] based model achieves 31.95% PQ (val) / 31.6% (test) and 55.26% PC (val) with 3 frames per second (fps) on GPU. Our novel *Wider* version of the MobileNetV2 [8] based model can achieve near real-time performance (22.61 fps on GPU) with reduced accuracy.

- We propose an alternative metric, Parsing Covering, to evaluate image parsing results from a region-based perspective.

We report results on additional datasets (Cityscapes, Pascal VOC 2012, and COCO) in the supplementary material.

2. Related Work

Image parsing: The task of image parsing refers to decomposing images into constituent visual patterns, such as textures and object instances. It unifies detection, segmentation, and recognition. Tu *et al*. [1] present the first attempt for image parsing in a Bayesian framework. Since then, there have been several works aiming to jointly perform detection and segmentation for whole scene understanding with AND-OR graphs [9, 10], Exemplars [11, 12, 13], or Conditional Random Fields [14, 15, 16, 17, 18, 19]. These early works evaluated image parsing results with separate metrics (*e.g*., one for object detection and one for semantic segmentation). There has been renewed interest in this task, also called Panoptic Segmentation, with the introduction of the unified instance-based Panoptic Quality (PQ) metric [2] into several benchmarks [20, 3].

Semantic segmentation: Most of the state-of-the-art semantic segmentation models are built upon fully convolutional neural networks (FCNs) [21, 22] and further improve the performance by incorporating different innovations. For example, it has been known that contextual information is essential for pixel labeling [23, 24, 25, 26, 27, 28, 29]. Following this idea, several works [30, 31, 32, 33, 34, 35] adopt image pyramids to encode contexts with different input sizes. Recently, PSPNet [36] proposes using spatial pyramid pooling [37, 38] at several grid scales (including image-level pooling [39]), and DeepLab [35, 40] proposes applying several parallel atrous convolutions [41, 42, 21, 43, 44] with different rates (called Atrous Spatial Pyramid Pooling, or ASPP). By effectively utilizing the multi-scale contextual information, these models demonstrate promising accuracy on several segmentation benchmarks. Another effective way is the employment of the encoder-decoder structure [45, 46, 47]. Typically, the encoder-decoder networks [48, 45, 46, 49, 50, 51, 52, 53, 54, 7, 55, 56] capture the context information in the encoder path and recover the object boundary in the decoder path. To maximize the accuracy on image parsing, the proposed DeeperLab utilizes most of these techniques, which are the FCN, ASPP and encoder-decoder structure.

Instance segmentation: Current solutions for instance segmentation could be roughly categorized into top-down and bottom-up methods. The top-down approaches [57, 58, 59, 60, 61, 62, 63] obtain instance masks by refining the predicted boxes from state-of-the-art detectors [64, 65, 66]. FCIS [58] employs the position-sensitive score maps [67]. Mask-RCNN [60], built on top of FPN [68], attaches another segmentation branch to Faster-RCNN [64] and demonstrates outstanding performance. Additionally, some methods directly aim for mask proposals instead of bounding box proposals, including [69, 70, 71, 72, 67]. On the other hand, the bottom-up approaches [73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84] generally adopt a two-stage processing: pixel-level predictions produced by the segmentation module are clustered together to form the instance-level predictions. Recently, PersonLab [85] predicts person keypoints and person instance segmentation, while DeNet [86] and CornerNet [87] detect instances by predicting the corners of their bounding boxes. Our work is similar in the sense that we also produce keypoints for instance segmentation, which however is only part of our whole image parsing pipeline.

Evaluation metrics: Semantic segmentation results can be evaluated by region-based metrics or contour-based metrics. Region-based metrics measure the proportion of correctly labelled pixels, including overall pixel accuracy [23, 24], mean class accuracy [24], and mean IOU (intersection-

*The code is available at http://deeperlab.mit.edu.

over-union) [88]. In contrast, contour-based metrics focus on the labeling precision around the segment boundaries. For example, [25] measures the pixel accuracy or IOU within a Trimap in a narrow band around segment boundaries. Class-agnostic segmentation can be evaluated with the Covering metric [4]. We refer the interested readers to [4, 89] for an overview of the related literature.

Instance segmentation is usually formulated as mask detection [57, 58, 60], considered as a refinement of bounding box detection. Thus, the task is typically measured with AP^r, which involves computing the intersection-over-union w.r.t. mask overlaps instead of box overlaps [90]. The segmentation quality is evaluated by averaging AP^r results at different mask overlap accuracy thresholds ranging from 0.5 to 0.95 [20, 91]. Another line of work [92, 74, 76, 93, 78] adopts the region-based Covering metric to evaluate the instance segmentation results, which is applicable to methods that do not allow overlapped predictions.

Image parsing results can be evaluated with the instance-based Panoptic Quality (PQ) metric [2], which treats all image regions with the same 'stuff' class as a single instance. An issue with the PQ metric is that all object instances are treated the same irrespective of their size, and thus the PQ metric may place disproportionate emphasis on small instances, as well as on 'things' over 'stuff' classes.

3. Methodology

We propose an efficient single-shot, bottom-up neural network for image parsing, motivated by DeepLab [7] and PersonLab [85], which is illustrated in Fig. 2. The proposed network adopts the encoder-decoder paradigm. For efficiency, the semantic segmentation and instance segmentation are generated from the *shared decoder output* and then fused to produce the final image parsing result.

For image parsing, the network usually operates on high resolution inputs (*e.g.*, 1441×1441 on resized Mapillary Vistas images in our experiments), which leads to high memory usage and latency. Below, we provide details of each component design regarding how we address this challenge and achieve a balance between accuracy and latency/memory footprint during both training and inference.

3.1. Encoder

We have experimented with two networks built on the efficient depthwise separable convolution [8]: the standard Xception-71 [5, 6, 7] for higher accuracy, and a novel *Wider* variant of MobileNetV2 [94] for faster inference.

Although standard MobileNetV2 performs well on the ImageNet image classification task with an input size of 224×224, it fails to capture long-range context information given its limited receptive field (491×491 pixels) for the task of image parsing with high-resolution inputs. Stacking more 3×3 convolutions is a common practice to increase the receptive field, as is done in Xception-71. However, the extra layers introduce more feature maps, which dominate the memory usage. Considering the limited computation resources, we propose to replace all the 3×3 convolutions in MobileNetV2 with 5×5 convolutions. This approach efficiently increases the receptive field to 981×981 while maintaining the same amount of memory footprint for feature maps and only mildly increasing the computation cost. We refer to the resulting backbone as *Wider MobileNetV2*.

Additionally, we augment the network backbone with the effective ASPP module (Atrous Spatial Pyramid Pooling) [35, 40]. ASPP applies several parallel atrous convolutions with different rates to further increase the receptive field. The feature map at the encoder output has stride 16, *i.e.*, its spatial resolution is equal to the input size downsampled by a factor of 16 across each spatial dimension.

3.2. Decoder

The goal of the decoder module is to recover detailed object boundaries. Following DeepLabV3+ [7], we adopt a simple design that combines the activations at the output of the encoder (with stride 16) with low-level feature maps from the network backbone (with stride 4). The number of channels of the concatenated ASPP outputs and the low-level feature map are first individually reduced by 1×1 convolution and then concatenated together. DeepLabV3+ bilinearly upsamples the reduced ASPP outputs before concatenation in order to account for the different spatial resolutions; however, the upsampling operation significantly increases the memory consumption. In this work, we apply the space-to-depth operation [95, 96] (Fig. 3) to the reduced low-level feature map, which keeps the memory usage of feature maps the same.

Similar to the encoder, the decoder uses two large kernel (7×7) depthwise convolutions to further increase the receptive field. The resultant feature map has 4096 channels, which is then reduced by depth-to-space (Fig. 3, reverse operation of space-to-depth), yielding a feature map with 256 channels and stride 4, which are used as the input for the image parsing prediction heads.

3.3. Image Parsing Prediction Heads

The proposed network contains five prediction heads, each of which is directly attached to the shared decoder output and consists of two convolution layers with kernel sizes of 7×7 and 1×1 respectively. One head (with 256 filters for the first 7×7 layer) is specific for semantic segmentation, while the other four (each with 64 filters for the first 7×7 layer) are used for class-agnostic instance segmentation.

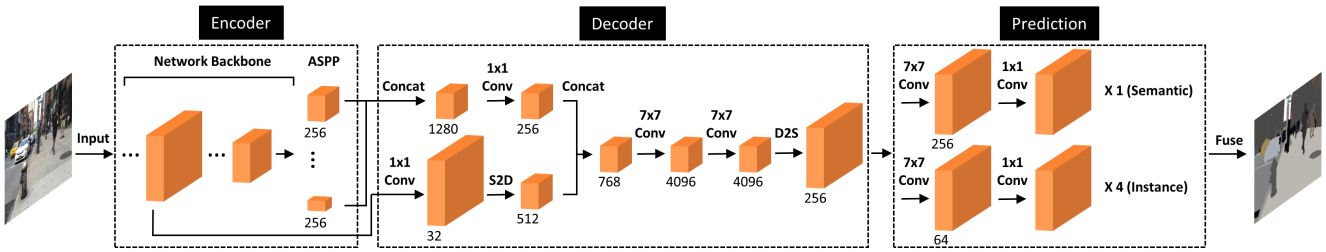

Figure 2: The proposed single-shot, bottom-up network architecture employs the encoder-decoder structure and produces per-pixel semantic and instance predictions. The number of channels of each feature map is specified in the figure.

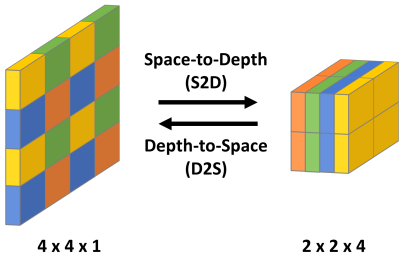

Figure 3: An example of the space-to-depth (S2D) and depth-to-space (D2S) operations. The S2D operation moves activations from the spatial dimension to the channel dimension and the D2S operation is the inverse.

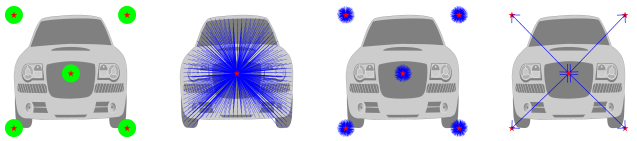

Figure 4: Four prediction maps generated by our instance-related heads: (a) keypoint heatmap, (b) long-range offset, (c) short-range offset, and (d) middle-range offset. The red stars denote the keypoints, the green disk denotes the target for keypoint prediction, and the blue lines/arrows denote the offsets from the current pixel to the target keypoint.

3.3.1 Semantic Segmentation Head

The semantic segmentation prediction is trained to minimize the bootstrapped cross-entropy loss [97, 98, 50], in which we sort the pixels based on the cross-entropy loss and we only backpropagate the errors in the top-K positions (hard example mining). We set $K = 0.15 \cdot N$, where N is the total number of pixels in the image. Moreover, we weigh the pixel loss based on instance sizes, putting more emphasis on small instances. Specifically, our proposed weighted bootstrapped cross-entropy loss is defined by:

$$\ell = -\frac{1}{K}\sum_{i=1}^{N} w_i \cdot \mathbb{1}[p_{i,y_i} < t_K] \cdot \log p_{i,y_i}, \qquad (1)$$

where y_i is the target class label for pixel i, $p_{i,j}$ is the predicted posterior probability for pixel i and class j, and $\mathbb{1}[x] = 1$ if x is true and 0 otherwise. The threshold t_K is set in a way that only the pixels with top-K highest losses are selected. We set the weight $w_i = 3$ for pixels that belong to instances with an area smaller than 64×64 and $w_i = 1$ everywhere else. By doing so, the network is trained to focus on both hard pixels and small instances.

3.3.2 Instance Segmentation Heads

Similar to [85, 86, 87], we adopt a *keypoint-based* representation for object instances. In particular, we consider the four bounding box corners and the center of mass as our $P = 5$ object keypoints.

Following PersonLab [85], we define four prediction heads, which are used for instance segmentation: a keypoint heatmap as well as long-range, short-range, and middle-range offset maps. Those predictions focus on predicting different relations between each pixel and the keypoints of its corresponding instance, which we fuse to form the class-agnostic instance segmentation as detailed in Sec. 3.4.1.

The keypoint heatmap (Fig. 4a) predicts whether a pixel is within a disk of radius R pixels centered in the corresponding keypoint. The target activation is equal to 1 in the interior of the disks and 0 elsewhere. We use the same disk radius $R = 25$ regardless of the size of an instance, so that the network pays the same attention to both large and small instances. The predicted keypoint heatmap contains P channels, one for each keypoint. We penalize prediction errors by the standard sigmoid cross entropy loss.

The long-range offset map (Fig. 4b) predicts the position offset from a pixel to all the corresponding keypoints, encoding the *long-range* information for each pixel. The predicted long-range offset map has $2P$ channels, where every two channels predict the offset in the horizontal and

vertical directions for each keypoint. We employ L_1 loss for long-range offset prediction, which is only activated at pixels belonging to object instances.

The short-range offset map (Fig. 4c) is similar to the long-range offset map except that it only focuses on pixels within the disk of radius $R = 25$ pixels around the keypoints, *i.e.*, the pixels having a value of one in the target heatmap (the green disk in Fig. 4a). The short range offset map also has $2P$ channels and are used to improve keypoint localization. We employ L_1 loss, which is only activated at the interior of the disks.

The middle-range offset map (Fig. 4d) predicts the offset among keypoint pairs, defined in a *directed keypoint relation graph (DKRG)*. This map is used to group keypoints that belong to the same instance (*i.e.*, instance detection via keypoints). As shown in Fig. 4d, we adopt the star graph [99], where the mass center is bi-directionally connected to the other four box corners. The predicted middle-range offset map has $2E$ channels, where E is the number of directed edges in the DKRG ($E = 8$ in the star graph). Similarly, we use two channels for each edge to predict the horizontal and vertical offsets and employ L_1 loss during training, which is only activated at the interior of the disks.

3.4. Prediction Fusion

We first explain how to merge the four predictions (keypoint heatmap, long-range, short-range, and middle-range offset maps) into a single class-agnostic instance segmentation map. Given the predicted semantic and instance segmentation maps, the final fusion step assigns both semantic and instance labels to every pixel in the image.

3.4.1 Instance Prediction

We generate the instance segmentation map from the four instance-related prediction maps similarly to PersonLab [85]. We will highlight the main steps and differences in the following paragraphs.

Recursive offset refinement: We observe that the predictions that are closer to the corresponding keypoints are more accurate. Therefore, we recursively refine the offset maps by itself and/or each other as in PersonLab [85].

Keypoint localization: For each keypoint, we perform Hough-voting on the short-range offset map and use the corresponding value in the keypoint heatmap (after sigmoid activation) as the voting weight to generate the short-range score map. We propose to also perform Hough-voting on the long-range offset map (using a weight equal to one for every vote) to generate the long-range score map. These two score maps are merged into one by taking per-pixel weighted sum. We then localize the keypoints by finding the local maxima in the resultant fused score map. Finally, we use the *Expected-OKS* score [85] to rescore all keypoints.

Instance detection: We cluster the keypoints to detect instances by using a fast greedy algorithm. All the keypoints are first pushed into a priority queue and popped one at a time. If the popped keypoint is in the proximity of the corresponding keypoint of an already detected instance, we reject it and continue the process. Otherwise, we follow the predicted middle-range offsets to identify the positions of the remaining four keypoints, thus forming a newly detected instance. The confidence score of the detected instance is defined as the average of its keypoint scores. After all the instances are detected, we use bounding box non-maximum suppression to remove overlapping instances.

Assignment of pixels to instances: Finally, given the detected instances, we assign an instance label to each pixel by using the predicted long-range offset map, which encodes the pixel-wise offset to the keypoints. Specifically, we assign each pixel to the detected instance whose keypoints have the smallest L_2-distance to the pixel's predicted keypoints (*i.e.*, its image location plus its predicted long-range offset).

3.4.2 Semantic and Instance Prediction Fusion

We opt for a simple merging method without any other post-processing, such as removal of small isolated regions in the segmentation maps. In particular, we start from the predicted semantic segmentation by considering 'stuff' (*e.g.*, sky) and 'thing' (*e.g.*, person) classes separately. Pixels predicted to have a 'stuff' class are assigned with a single unique instance label. For the other pixels, their instance labels are determined from the instance segmentation result while their semantic labels are resolved by the majority vote of the corresponding predicted semantic labels.

3.5. Evaluation Metrics

Herein, we briefly review the Panoptic Quality (PQ) metric [2] and propose the Parsing Covering (PC) metric, extended from the existing Covering metric [4].

Given a groundtruth segmentation S and a predicted segmentation S', PQ is defined as follows:

$$PQ = \frac{\sum_{(R,R')\in TP} IOU(R, R')}{|TP| + \frac{1}{2}|FP| + \frac{1}{2}|FN|}, \tag{2}$$

where R and R' are groundtruth regions and predicted regions respectively, and $|TP|$, $|FP|$, and $|FN|$ are the number of true positives, false postives, and false negatives. The matching is determined by a threshold of 0.5 Intersection-Over-Union (IOU).

PQ treats all regions of the same 'stuff' class as one instance, and the size of instances is not considered. For example, instances with 10×10 pixels contribute equally to

the metric as instances with 1000×1000 pixels. Therefore, PQ is sensitive to false positives with small regions and some heuristics could improve the performance, such as removing those small regions (as also pointed out in the open-sourced evaluation code from [2]). Thus, we argue that PQ is suitable in applications where one cares equally for the parsing quality of instances irrespective of their sizes.

There are applications where one pays more attention to large objects, *e.g.*, portrait segmentation (where large people should be segmented perfectly) or autonomous driving (where nearby objects are more important than far away ones). Motivated by this, we propose to also evaluate the quality of image parsing results by extending the existing Covering metric [4], which accounts for instance sizes. Specifically, our proposed metric, Parsing Covering (PC), is defined as follows:

$$Cov_i = \frac{1}{N_i} \sum_{R \in S_i} |R| \cdot \max_{R' \in S'_i} IOU(R, R'), \tag{3}$$

$$N_i = \sum_{R \in S_i} |R|, \tag{4}$$

$$PC = \frac{1}{C} \sum_{i=1}^{C} Cov_i, \tag{5}$$

where S_i and S'_i are the groundtruth segmentation and predicted segmentation for the i-th semantic class respectively, and N_i is the total number of pixels of groundtruth regions from S_i. The Covering for class i, Cov_i, is computed in the same way as the original Covering metric except that only groundtruth regions from S_i and predicted regions from S'_i are considered. PC is then obtained by computing the average of Cov_i over C semantic classes. We plan to open-source our implementation of the PC metric to facilitate its adoption by other researchers.

We note that Covering has been used in several instance segmentation works [92, 74, 76, 93, 78]. The proposed PC is a simple extension of the Covering to evaluate image parsing results. It was pointed out in [76] that Covering does not penalize the false positives. This is because, in [76], the Covering for the background class is not evaluated, which absorbs other classes' false positives. In the case of image parsing, this will not happen since *all* the classes and every pixel will be taken into account.

Another notable difference between PQ and the proposed PC is that there is no matching involved in PC and hence no matching threshold. As an attempt to treat equally 'thing' and 'stuff', the segmentation of 'stuff' classes still receives partial PC score if the segmentation is only partially correct. For example, if one out of three equally-sized trees is perfectly segmented, the model will get the same partial score by using PC regardless of considering 'tree' as 'stuff' or 'thing'.

4. Experimental Results

We demonstrate the effectiveness and efficiency of DeeperLab and present ablation studies on the Mapillary Vistas [3]. This dataset contains 66 semantic classes in a variety of traffic-related images, whose sizes range from 1024×768 to higher than 4000×6000. We report both Panoptic Quality (PQ) [2] and the proposed Parsing Covering (PC) for accuracy, and speed on a desktop CPU and GPU[†]. We report results on other datasets (Cityscapes, Pascal VOC 2012, and COCO) in the supplementary material.

All the models are trained end-to-end without piecewise pretraining of each component except that the backbone is pretrained on ImageNet-1K [100]. The training configuration is the same as that in [40]. In short, we employ the same learning rate schedule (*i.e.*, "poly" policy [39] with an initial learning rate of 0.01), fine-tune batch normalization [101] parameters for all layers, and use random scale data augmentation during training. The training batch sizes are 28 and 16 when employing MobileNetV2 (MNV2) [94] and Xception-71 [5, 6, 7] as the network backbone, respectively. Similar to [36, 3], we resize the images to 1441 pixels at the longest side to handle the large input variations on Mapillary Vistas and randomly crop 721×721 patches during training.

Our numbers are reported with *single-scale* inference. Moreover, we do not employ any heuristic post-processing such as small region removal or assigning VOID label to low confidence predictions.

4.1. Performance on Mapillary Vistas

DeeperLab aims to achieve a balance between accuracy and speed, which facilitates the deployment of image parsing. In this section, we will analyze both of the accuracy and speed of the proposed DeeperLab[‡].

Validation set performance: We summarize the accuracy and speed of DeeperLab on the validation set in Tab. 1, where the networks are trained longer than those in the ablation study, 500K *vs.* 200K iterations, respectively. Our Xception-71 based model[§] attains 31.95% PQ and 55.26% PC, while the Wider MobileNetV2 based model achieves 25.20% PQ and 49.80% PC with faster inference (6.19 *vs.* 3.09 fps on GPU). We have also experimented with an even faster *Light Wider MobileNetV2* variant, which employs a simpler decoder with 3×3 kernels and fewer filters (128 instead of 256 filters). The speed increases to 9.37 fps on

[†]CPU: Intel(R) Xeon(R) CPU E5-1650 v3 @ 3.50GHz, GPU: Tesla V100-SXM2

[‡]To the best of our knowledge, there is no 1) peer-reviewed works that release the models or report the latency numbers with 2) single-model, 3) single-scale settings on Mapillary Vistas at the time of the preparation of this work.

[§]Our Xception-71 based model attains mIOU 55.30% for the semantic segmentation task, outperforming 53.12% in [102].

Method	Input Size	PQ (%)	PC (%)	fps (CPU)	fps (GPU)	Merge (ms)
Light Wider MNV2	721×721	17.59	43.43	0.77	22.61	45
Light Wider MNV2	1441×1441	22.36	47.52	0.18	9.37	145
Wider MNV2	1441×1441	25.20	49.80	0.09	6.19	145
Xception-71	1441×1441	31.95	55.26	0.06	3.09	145

Table 1: DeeperLab performance on the Mapillary Vistas validation set. Xception-71 based model attains higher accuracy while Wider MobileNetV2 (Wider MNV2) based model achieves faster inference. The model can be sped up by simplifying the decoder structure (Light Wider MNV2) with a small accuracy drop. With downsampled inputs, Light Wider MNV2 can reach near real-time speed.

Method	PQ	SQ	RQ	PQ^{Th}	SQ^{Th}	RQ^{Th}	PQ^{St}	SQ^{St}	RQ^{St}
Light Wider MNV2 †	17.3	66.2	22.7	9.1	62.6	12.8	28.2	71.0	35.8
Light Wider MNV2	22.6	69.6	29.3	15.4	67.2	21.1	32.1	72.9	40.2
Wider MNV2	25.3	70.6	32.3	17.6	65.5	23.4	35.5	77.4	44.0
Xception-71	31.6	75.5	40.1	25.0	73.4	33.1	40.3	78.3	49.3

Table 2: DeeperLab performance on the Mapillary Vistas test set. †: input size is downsampled by 2 (721×721).

GPU with a small accuracy drop. Additionally, if we downsample the input by 2, the Light Wider MobileNetV2 model can reach near real-time speed (22.61 fps on GPU). Note that we have not used any other tricks, such as folding batch norm or quantizing the models, to further speed up inference. Moreover, the extra step of fusing the semantic and instance segmentation is fast and mostly determined by the input resolution (145 ms for 1441×1441 image and 45 ms for 721×721 image with an unoptimized CPU implementation). Fig. 5 shows the qualitative results.

Test set performance: Our test set result is summarized in Tab. 2, where only PQ is provided by the test server.

4.2. Ablation Study

Wider MobileNetV2 backbone design: The original MobileNetV2 [94] employs 3×3 kernels in all convolutions. We experiment with different kernel sizes, such as 5×5 or 7×7, to enlarge the network's receptive field. As shown in Tab. 3, increasing the kernel size is a very effective approach. The PQ and PC are improved by 1.75% and 4.54% respectively when the 5×5 kernel size is used on Mapillary Vistas, which contains images with much higher resolutions than ImageNet. See Fig. 6 for a visual result. We opt for the 5×5 kernel size since using the 7×7 kernel size only marginally improves the performance, and the resulting network backbone is referred to as Wider MobileNetV2. Additionally, adopting the ASPP module [40] further improves accuracy for all settings.

Decoder and prediction head design: Given the Wider MobileNetV2 augmented with the ASPP module, we experiment with different decoder architectures in Tab. 4. The baseline, attaining a performance of 19.85% PQ and 42.98% PC, is obtained by directly attaching prediction heads (only one 1×1 convolution) to the feature maps. With a simple decoder, which concatenates the bilinearly upsampled (**BU**) reduced ASPP output (stride = 16) with the reduced low-level feature (stride = 4), the accuracy is improved by 0.93% PQ and 0.85% PC. We find that it is effective to increase the number of layers in the prediction heads. Adding one more 3×3 convolution layer in all the prediction heads (**DH**) further improves accuracy by 0.81% PQ and 1.12% PC. By enlarging the convolution kernel size from 3×3 to 7×7 in both the decoder and the extra convolution in the heads, the model achieves 22.31% PQ and 44.62% PC. Last, the accuracy is significantly improved by replacing the bilinear upsampling strategy by the proposed S2D/D2S strategy, reaching 23.48% PQ and 46.33% PC.

Hard pixel mining: We find hard pixel mining (**HPM**) beneficial. As explained in Sec. 3.3.1, we sort the pixels based on their losses and only backpropagate the top 15% pixels, the PQ is increased by 0.57%. If we increase the loss weight of instances smaller than 64×64 by $3\times$ (**SI**), the accuracy is improved by 0.2% PQ. To maximize the accuracy, we combine these two approaches and achieve 0.92% PQ

Kernel Size	ASPP	PQ (%)	PC (%)
3×3		16.17	34.80
5×5		17.92	39.34
7×7		18.27	40.33
3×3	✓	19.21	41.07
5×5	✓	19.85	42.98
7×7	✓	20.14	43.40

Table 3: The comparison between different Wider MobileNetV2 designs. Employing a larger kernel size in all the convolutions of MobileNetV2 significantly improves the accuracy on Mapillary Vistas, where the images have much larger resolutions than ImageNet. Moreover, the ASPP module is effective for all settings.

BU	S2D	DH	LK	PQ (%)	PC (%)
				19.85	42.98
✓				20.78	43.83
✓		✓		21.59	44.95
✓		✓	✓	22.31	44.62
	✓			21.12	44.86
	✓	✓		22.45	46.30
	✓	✓	✓	23.48	46.33

Table 4: The comparison between different decoder and prediction head designs with Wider MobileNetV2 augmented with ASPP. The baseline is obtained by directly attaching the prediction heads (one 1×1 convolution) to the output. **BU:** Decoder with Bilinear Upsampling. **S2D:** Decoder with D2S and S2D. **DH:** Deeper Heads with one more extra convolution in the prediction heads. **LK:** Using Larger 7×7 Kernels in both the decoder and the extra prediction head.

Figure 5: A few image parsing results on the Mapillary Vistas validation set with proposed DeeperLab based on Xception-71. The first row is the predicted semantic segmentation and the second row is the predicted instance segmentation. Note that our model does not generate any VOID labels.

(a) MobileNetV2

(b) Wider MobileNetV2

Figure 6: Instance segmentation results obtained by directly applying 1×1 convolutions to the feature maps extracted by MobileNetV2 and Wider MobileNetV2 (*i.e.*, no ASPP or any other modules). Because of the larger receptive field, Wider MobileNetV2 can segment large objects better (*e.g.*, the crosswalk and the rightmost pole).

HPM	SI	PQ (%)	PC (%)
		24.07	48.23
✓		24.64	48.34
	✓	24.27	48.42
✓	✓	24.99	49.23

Table 5: The comparison between different hard pixel mining approaches. **HPM:** Hard Pixel Mining. **SI:** Larger loss weights on Small Instances. The accuracy is maximized when both approaches are employed.

and 1% PC improvement over the baseline.

Directed keypoint relation graph: We compare two different directed keypoint relation graphs. The first one is the star graph as explained in Sec. 3.3.2. Another one is the rectangular graph, where the keypoints are connected in a rectangular shape, and there is no mass-center keypoint. As shown in Tab. 6, using the star graph results in 0.53% higher PQ and 1.89% higher PC. We think the mass-center keypoint is important for instance detection.

Star	Rectangle	PQ (%)	PC (%)
✓		24.07	48.23
	✓	23.54	46.44

Table 6: The comparison between alternative directed keypoint relation graphs. Employing the star graph where the mass-center keypoint is connected to the other four keypoints leads to higher accuracy than the rectangle graph.

ASPP	BU	S2D	HPM	SI	PQ (%)	PC (%)
✓	✓				27.79	50.80
✓		✓			28.34	50.88
✓		✓	✓		30.04	53.35
		✓	✓	✓	30.46	54.55

Table 7: Employing Xception-71 as the network backbone with different methods. **ASPP:** Encoder with Atrous Spatial Pyramid Pooling. **BU:** Decoder with Bilinear Upsampling. **S2D:** Decoder with D2S and S2D. **HPM:** Hard Pixel Mining. **SI:** Larger loss weights on Small Instances.

Deeper network backbone: In Tab. 7, we report the ablation study with Xception-71 [5, 6, 7] as the network backbone. Our best Xception-71-based model attains an accuracy of 30.46% PQ and 54.55% PC on the validation set.

5. Conclusion

We have proposed and demonstrated the effectiveness of the image parser, DeeperLab, for the challenging whole image parsing task. Our proposed model design attains a good trade-off between accuracy and speed. This is made possible by adopting a single-shot, bottom-up, and single-inference paradigm and integrating various design innovations. These innovations include extensively applying depthwise separable convolution, using a shared decoder output with simple two-layer prediction heads, enlarging kernel sizes instead of making the network deeper, employing space-to-depth and depth-to-space rather than upsampling, and performing hard data mining. Moreover, we have also proposed the 'Parsing Covering' (PC) metric to evaluate the parsing accuracy from the region based perspective. We hope the design strategies and the metric will facilitate future research into image parsing.

Acknowledgments We thank Peter Kontschieder for the valuable discussion about the Mapillary Vistas result format; Florian Schroff, Hartwig Adam, and Mobile Vision team for support.

References

[1] Z. Tu, X. Chen, A. L. Yuille, and S.-C. Zhu, "Image parsing: Unifying segmentation, detection, and recognition," *IJCV*, 2005. 1, 2

[2] A. Kirillov, K. He, R. Girshick, C. Rother, and P. Dollár, "Panoptic segmentation," *arXiv:1801.00868*, 2018. 1, 2, 3, 5, 6

[3] G. Neuhold, T. Ollmann, S. R. Bulò, and P. Kontschieder, "The mapillary vistas dataset for semantic understanding of street scenes.," in *ICCV*, 2017. 1, 2, 6

[4] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik, "Contour detection and hierarchical image segmentation," *PAMI*, 2011. 2, 3, 5, 6

[5] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *CVPR*, 2017. 2, 3, 6, 8

[6] H. Qi, Z. Zhang, B. Xiao, H. Hu, B. Cheng, Y. Wei, and J. Dai, "Deformable convolutional networks – coco detection and segmentation challenge 2017 entry," *ICCV COCO Challenge Workshop*, 2017. 2, 3, 6, 8

[7] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *ECCV*, 2018. 2, 3, 6, 8

[8] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv:1704.04861*, 2017. 2, 3

[9] S.-C. Zhu and D. Mumford, "A stochastic grammar of images," *Foundations and Trends in Computer Graphics and Vision*, 2007. 2

[10] L. Zhu, Y. Chen, Y. Lin, C. Lin, and A. Yuille, "Recursive segmentation and recognition templates for image parsing," *TPAMI*. 2

[11] T. Malisiewicz and A. A. Efros, "Recognition by association via learning per-exemplar distances," in *CVPR*, 2008. 2

[12] J. Tighe and S. Lazebnik, "Finding things: Image parsing with regions and per-exemplar detectors," in *CVPR*, 2013. 2

[13] P. Isola and C. Liu, "Scene collaging: Analysis and synthesis of natural images with semantic layers," in *ICCV*, 2013. 2

[14] A. Rabinovich, A. Vedaldi, C. Galleguillos, E. Wiewiora, and S. Belongie, "Objects in context," in *ICCV*, 2007. 2

[15] S. Gould, T. Gao, and D. Koller, "Region-based segmentation and object detection," in *NIPS*, 2009. 2

[16] G. Heitz, S. Gould, A. Saxena, and D. Koller, "Cascaded classification models: Combining models for holistic scene understanding," in *NIPS*, 2009. 2

[17] L. Ladickỳ, P. Sturgess, K. Alahari, C. Russell, and P. H. Torr, "What, where and how many? combining object detectors and crfs," in *ECCV*, 2010. 2

[18] J. Yao, S. Fidler, and R. Urtasun, "Describing the scene as a whole: Joint object detection, scene classification and semantic segmentation," in *CVPR*, 2012. 2

[19] M. Sun, B.-s. Kim, P. Kohli, and S. Savarese, "Relating things and stuff via objectproperty interactions," *TPAMI*, 2014. 2

[20] T.-Y. Lin *et al.*, "Microsoft coco: Common objects in context," in *ECCV*, 2014. 2, 3, 13

[21] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "Overfeat: Integrated recognition, localization and detection using convolutional networks," in *ICLR*, 2014. 2

[22] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *CVPR*, 2015. 2

[23] X. He, R. S. Zemel, and M. Carreira-Perpindn, "Multiscale conditional random fields for image labeling," in *CVPR*, 2004. 2

[24] J. Shotton, J. Winn, C. Rother, and A. Criminisi, "Textonboost for image understanding: Multi-class object recognition and segmentation by jointly modeling texture, layout, and context," *IJCV*, 2009. 2

[25] P. Kohli, P. H. Torr, *et al.*, "Robust higher order potentials for enforcing label consistency," *IJCV*, vol. 82, no. 3, pp. 302–324, 2009. 2, 3

[26] L. Ladicky, C. Russell, P. Kohli, and P. H. Torr, "Associative hierarchical crfs for object class image segmentation," in *ICCV*, 2009. 2

[27] S. Gould, R. Fulton, and D. Koller, "Decomposing a scene into geometric and semantically consistent regions," in *ICCV*, 2009. 2

[28] M. Mostajabi, P. Yadollahpour, and G. Shakhnarovich, "Feedforward semantic segmentation with zoom-out features," in *CVPR*, 2015. 2

[29] J. Dai, K. He, and J. Sun, "Convolutional feature masking for joint object and stuff segmentation," in *CVPR*, 2015. 2

[30] C. Farabet, C. Couprie, L. Najman, and Y. LeCun, "Learning hierarchical features for scene labeling," *PAMI*, 2013. 2

[31] D. Eigen and R. Fergus, "Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture," in *ICCV*, 2015. 2

[32] P. Pinheiro and R. Collobert, "Recurrent convolutional neural networks for scene labeling," in *ICML*, 2014. 2

[33] G. Lin, C. Shen, A. van den Hengel, and I. Reid, "Efficient piecewise training of deep structured models for semantic segmentation," in *CVPR*, 2016. 2

[34] L.-C. Chen, Y. Yang, J. Wang, W. Xu, and A. L. Yuille, "Attention to scale: Scale-aware semantic image segmentation," in *CVPR*, 2016. 2

[35] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *TPAMI*, 2017. 2, 3

[36] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *CVPR*, 2017. 2, 6

[37] K. Grauman and T. Darrell, "The pyramid match kernel: Discriminative classification with sets of image features," in *ICCV*, 2005. 2

[38] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," in *CVPR*, 2006. 2

[39] W. Liu, A. Rabinovich, and A. C. Berg, "Parsenet: Looking wider to see better," *arXiv:1506.04579*, 2015. 2, 6

[40] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," *arXiv:1706.05587*, 2017. 2, 3, 6, 7

[41] M. Holschneider, R. Kronland-Martinet, J. Morlet, and P. Tchamitchian, "A real-time algorithm for signal analysis with the help of the wavelet transform," in *Wavelets: Time-Frequency Methods and Phase Space*, pp. 289–297, Springer Berlin Heidelberg, 1989. 2

[42] A. Giusti, D. Ciresan, J. Masci, L. Gambardella, and J. Schmidhuber, "Fast image scanning with deep max-pooling convolutional neural networks," in *ICIP*, 2013. 2

[43] G. Papandreou, I. Kokkinos, and P.-A. Savalle, "Modeling local and global deformations in deep learning: Epitomic convolution, multiple instance learning, and sliding window detection," in *CVPR*, 2015. 2

[44] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Semantic image segmentation with deep convolutional nets and fully connected crfs," in *ICLR*, 2015. 2

[45] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *MICCAI*, 2015. 2

[46] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *arXiv:1511.00561*, 2015. 2

[47] A. Newell, K. Yang, and J. Deng, "Stacked hourglass networks for human pose estimation," in *ECCV*, 2016. 2

[48] H. Noh, S. Hong, and B. Han, "Learning deconvolution network for semantic segmentation," in *ICCV*, 2015. 2

[49] G. Lin, A. Milan, C. Shen, and I. Reid, "Refinenet: Multi-path refinement networks with identity mappings for high-resolution semantic segmentation," in *CVPR*, 2017. 2

[50] T. Pohlen, A. Hermans, M. Mathias, and B. Leibe, "Full-resolution residual networks for semantic segmentation in street scenes," in *CVPR*, 2017. 2, 4

[51] C. Peng, X. Zhang, G. Yu, G. Luo, and J. Sun, "Large kernel matters–improve semantic segmentation by global convolutional network," in *CVPR*, 2017. 2

[52] M. A. Islam, M. Rochan, N. D. Bruce, and Y. Wang, "Gated feedback refinement network for dense image labeling," in *CVPR*, 2017. 2

[53] Z. Wojna, V. Ferrari, S. Guadarrama, N. Silberman, L.-C. Chen, A. Fathi, and J. Uijlings, "The devil is in the decoder," in *BMVC*, 2017. 2

[54] J. Fu, J. Liu, Y. Wang, and H. Lu, "Stacked deconvolutional network for semantic segmentation," *arXiv:1708.04943*, 2017. 2

[55] Z. Zhang, X. Zhang, C. Peng, D. Cheng, and J. Sun, "Exfuse: Enhancing feature fusion for semantic segmentation," in *ECCV*, 2018. 2

[56] T. Xiao, Y. Liu, B. Zhou, Y. Jiang, and J. Sun, "Unified perceptual parsing for scene understanding," in *ECCV*, 2018. 2

[57] J. Dai, K. He, and J. Sun, "Instance-aware semantic segmentation via multi-task network cascades," in *CVPR*, 2016. 2, 3

[58] Y. Li, H. Qi, J. Dai, X. Ji, and Y. Wei, "Fully convolutional instance-aware semantic segmentation," in *CVPR*, 2017. 2, 3

[59] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei, "Deformable convolutional networks," in *ICCV*, 2017. 2

[60] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *ICCV*, 2017. 2, 3, 13

[61] L.-C. Chen, A. Hermans, G. Papandreou, F. Schroff, P. Wang, and H. Adam, "Masklab: Instance segmentation by refining object detection with semantic and direction features," in *CVPR*, 2018. 2

[62] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, "Path aggregation network for instance segmentation," in *CVPR*, 2018. 2

[63] C. Peng, T. Xiao, Z. Li, Y. Jiang, X. Zhang, K. Jia, G. Yu, and J. Sun, "Megdet: A large mini-batch object detector," in *CVPR*, 2018. 2

[64] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *NIPS*, 2015. 2

[65] J. Dai, Y. Li, K. He, and J. Sun, "R-fcn: Object detection via region-based fully convolutional networks," in *NIPS*, 2016. 2

[66] Z. Cai and N. Vasconcelos, "Cascade r-cnn: Delving into high quality object detection," in *CVPR*, 2018. 2

[67] J. Dai, K. He, Y. Li, S. Ren, and J. Sun, "Instance-sensitive fully convolutional networks," in *ECCV*, 2016. 2

[68] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *CVPR*, 2017. 2, 13

[69] J. Carreira and C. Sminchisescu, "CPMC: Automatic object segmentation using constrained parametric min-cuts," *PAMI*, vol. 34, no. 7, pp. 1312–1328, 2012. 2

[70] P. Arbeláez, J. Pont-Tuset, J. T. Barron, F. Marques, and J. Malik, "Multiscale combinatorial grouping," in *CVPR*, 2014. 2

[71] P. O. Pinheiro, R. Collobert, and P. Dollár, "Learning to segment object candidates," in *NIPS*, 2015. 2

[72] P. O. Pinheiro, T.-Y. Lin, R. Collobert, and P. Dollár, "Learning to refine object segments," in *ECCV*, 2016. 2

[73] X. Liang, Y. Wei, X. Shen, J. Yang, L. Lin, and S. Yan, "Proposal-free network for instance-level object segmentation," *arXiv preprint arXiv:1509.02636*, 2015. 2

[74] Z. Zhang, A. G. Schwing, S. Fidler, and R. Urtasun, "Monocular object instance segmentation and depth ordering with cnns," in *ICCV*, 2015. 2, 3, 6

[75] J. Uhrig, M. Cordts, U. Franke, and T. Brox, "Pixel-level encoding and depth layering for instance-level semantic labeling," *arXiv:1604.05096*, 2016. 2

[76] Z. Zhang, S. Fidler, and R. Urtasun, "Instance-level segmentation for autonomous driving with deep densely connected mrfs," in *CVPR*, 2016. 2, 3, 6

[77] M. Bai and R. Urtasun, "Deep watershed transform for instance segmentation," in *CVPR*, 2017. 2

[78] S. Liu, J. Jia, S. Fidler, and R. Urtasun, "Sgn: Sequential grouping networks for instance segmentation," in *ICCV*, 2017. 2, 3, 6

[79] A. Kirillov, E. Levinkov, B. Andres, B. Savchynskyy, and C. Rother, "Instancecut: from edges to instances with multicut," in *CVPR*, 2017. 2

[80] A. Newell and J. Deng, "Associative embedding: End-to-end learning for joint detection and grouping," in *NIPS*, 2017. 2

[81] A. Fathi, Z. Wojna, V. Rathod, P. Wang, H. O. Song, S. Guadarrama, and K. P. Murphy, "Semantic instance segmentation via deep metric learning," *arXiv:1703.10277*, 2017. 2

[82] B. D. Brabandere, D. Neven, and L. V. Gool, "Semantic instance segmentation with a discriminative loss function," *arXiv:1708.02551*, 2017. 2

[83] A. Kendall, Y. Gal, and R. Cipolla, "Multi-task learning using uncertainty to weigh losses for scene geometry and semantics," in *CVPR*, 2018. 2

[84] Y. Liu, S. Yang, B. Li, W. Zhou, J. Xu, H. Li, and Y. Lu, "Affinity derivation and graph merge for instance segmentation," in *ECCV*, 2018. 2

[85] G. Papandreou, T. Zhu, L.-C. Chen, S. Gidaris, J. Tompson, and K. Murphy, "Personlab: Person pose estimation and instance segmentation with a bottom-up, part-based, geometric embedding model," in *ECCV*, 2018. 2, 3, 4, 5

[86] L. Tychsen-Smith and L. Petersson, "Denet: Scalable real-time object detection with directed sparse sampling," in *ICCV*, 2017. 2, 4

[87] H. Law and J. Deng, "Cornernet: Detecting objects as paired keypoints," in *ECCV*, 2018. 2, 4

[88] M. Everingham, S. M. A. Eslami, L. V. Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes challenge a retrospective," *IJCV*, 2014. 3, 13

[89] G. Csurka, D. Larlus, F. Perronnin, and F. Meylan, "What is a good evaluation measure for semantic segmentation?.," in *BMVC*, 2013. 3

[90] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik, "Simultaneous detection and segmentation," in *ECCV*, 2014. 3

[91] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *CVPR*, 2016. 3, 13

[92] N. Silberman, D. Sontag, and R. Fergus, "Instance segmentation of indoor scenes using a coverage loss," in *ECCV*, 2014. 3, 6

[93] M. Ren and R. S. Zemel, "End-to-end instance segmentation with recurrent attention," in *CVPR*, 2017. 3, 6

[94] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *CVPR*, 2018. 3, 6, 7

[95] W. Shi, J. Caballero, F. Huszár, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang, "Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network," in *CVPR*, 2016. 3

[96] M. S. Sajjadi, R. Vemulapalli, and M. Brown, "Frame-recurrent video super-resolution," in *CVPR*, 2018. 3

[97] Z. Wu, C. Shen, and A. van den Hengel, "Bridging category-level and instance-level semantic image segmentation," *arXiv:1605.06885*, 2016. 4

[98] S. R. Bulò, G. Neuhold, and P. Kontschieder, "Loss max-pooling for semantic image segmentation," in *CVPR*, 2017. 4

[99] O. Veksler, "Star shape prior for graph-cut image segmentation," in *ECCV*, 2008. 5

[100] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *IJCV*, 2015. 6

[101] S. Ioffe and C. Szegedy, "Batch normalization: accelerating deep network training by reducing internal covariate shift," in *ICML*, 2015. 6

[102] S. R. Bulò, L. Porzi, and P. Kontschieder, "In-place activated batchnorm for memory-optimized training of dnns," in *CVPR*, 2018. 6

[103] R. Vemulapalli, O. Tuzel, M.-Y. Liu, and R. Chellappa, "Gaussian conditional random field network for semantic segmentation," in *CVPR*, 2016. 13

[104] Q. Li, A. Arnab, and P. H. Torr, "Weakly-and semi-supervised panoptic segmentation," in *ECCV*, 2018. 13

[105] B. Hariharan, P. Arbeláez, L. Bourdev, S. Maji, and J. Malik, "Semantic contours from inverse detectors," in *ICCV*, 2011. 13

Method	Input Size	PQ (%)	PC (%)	fps (CPU)	fps (GPU)	Merge (ms)
Light Wider MNV2	513×1025	39.32	64.66	0.96	23.99	45
Light Wider MNV2	1025×2049	48.06	69.70	0.24	10.21	154
Wider MNV2	1025×2049	52.33	74.04	0.10	6.71	154
Xception-71	1025×2049	56.53	75.63	0.07	3.24	154
Li *et al*. [104]	-	53.8	-	-	-	

Table 8: DeeperLab performance on the Cityscapes validation set. Xception-71 based model attains higher accuracy than [104] while Wider MobileNetV2 (Wider MNV2) based model achieves faster inference with comparable accuracy. The model can be further sped up by simplifying the decoder structure (Light Wider MNV2) with a small accuracy drop. With downsampled inputs, Light Wider MNV2 can reach near real-time speed.

Method	Input Size	PQ (%)	PC (%)	fps (CPU)	fps (GPU)	Merge (ms)
Light Wider MNV2	257×257	40.16	59.76	5.65	40.82	5
Light Wider MNV2	513×513	54.09	71.09	1.83	35.01	19
Wider MNV2	513×513	58.75	72.72	0.77	23.76	19
Xception-71	513×513	67.35	77.57	0.57	13.93	19
Li *et al*. [104] †	-	62.7	-	-	-	-
Li *et al*. [104] ‡	-	63.1	-	-	-	-

Table 9: DeeperLab performance on the PASCAL VOC 2012 validation set. Xception-71 based model attains higher accuracy than [104] without pretraining on COCO [20]. Moreover, the simplified Wider MNV2 (Light Wider MNV2) reaches real-time speed without downsampling inputs. †: fully-supervised without COCO. ‡: fully-supervised with COCO.

Method	Input Size	PQ (%)	PC (%)	fps (CPU)	fps (GPU)	Merge (ms)
Light Wider MNV2	321×321	17.51	39.15	3.13	33.84	8
Light Wider MNV2	641×641	24.10	48.47	0.88	20.81	25
Wider MNV2	641×641	27.91	52.38	0.43	17.19	25
Xception-71	641×641	33.79	56.82	0.33	10.59	25

Table 10: DeeperLab performance on the COCO validation set. Xception-71 based model attains higher accuracy while Wider MobileNetV2 (Wider MNV2) based model achieves faster inference. The model can be sped up by simplifying the decoder structure (Light Wider MNV2) with a small accuracy drop. With downsampled inputs, Light Wider MNV2 can reach real-time speed. Note our Xception-71 based model attains the mIOU of 55.26% for semantic segmentation task.

In this supplementary material,

- we show the results of *DeeperLab* on other datasets, which are Cityscapes [91], PASCAL VOC 2012 [88] and COCO [20]. The experimental setting is the same as that mentioned in the main paper, unless otherwise stated.
- we provide more visualized results for Mapillary Vistas [103].

A. Performance on Cityscapes

Experimental setting: Without using the extra coarse annotations in Cityscapes [91], the models are trained on the training set ($2,975$ images) and evaluated on the validation set (500 images) with the crop size of 721×721.

Validation set performance: We summarize the accuracy and speed of DeeperLab on the validation set of Cityscapes in Tab. 8. Our Xception-71 based model outperforms [104] in terms of both Panoptic Quality (PQ) and Parsing Covering (PC), and our Wider MobileNetV2 achieves comparable accuracy at the speed of 6.71 fps on GPU. Moreover, our Light Wider MobileNetV2 with downsampled inputs attains near real-time speed (23.99 fps) on GPU. Fig. 7 and Fig. 10 show the qualitative results.

B. Performance on PASCAL VOC 2012

Experimental setting: We augment the training set of the original PASCAL VOC 2012 [88] with the extra annotations provided by [105], resulting in $10,582$ training images (*train_aug*). The models are trained on this *train_aug* set and evaluated on the validation set ($1,449$ images).

Validation set performancet: We summarize the accuracy and speed of DeeperLab on the validation set of PASCAL VOC 2012 in Tab. 9. Our Xception-71 based model outperforms [104] in terms of both Panoptic Quality (PQ) and Parsing Covering (PC) even without pretraining on COCO [20]. Moreover, our Light Wider MobileNetV2 attains real-time speed (35.01 fps on GPU) without downsampling inputs. Fig. 8 and Fig. 11 show the qualitative results.

C. Performance on COCO

Experimental setting: Although enlarging images has been shown to be effective on COCO [68, 60], we do not upsample the input images because of the consideration of speed. We leave exploring this augmentation as future work and focus on high-speed single-shot models in this work. Moreover, we do not perform hard pixel mining on COCO because it hurts the accuracy.

Validation set performance: We summarize the accuracy and speed of DeeperLab on the validation set of COCO in Tab. 10. Our Xception-71 based model attains 33.79% Panoptic Quality (PQ) and 56.82% Parsing Covering (PC) at the speed of 10.59 fps on GPU. Wider MobileNetV2 based model increases the speed to 17.19 fps on GPU at the cost of accuracy. Our Light Wider MobileNetV2 with downsampled inputs further pushes the speed to 33.84 fps on GPU. Fig. 9 and Fig. 12 show the qualitative results.

Test-dev set performance: The performance of our models on the test-dev set of COCO is reported in Tab. 11. We can see that the numbers on the test-dev set are very

Method	PQ	SQ	RQ	PQ^{Th}	SQ^{Th}	RQ^{Th}	PQ^{St}	SQ^{St}	RQ^{St}
Light Wider MNV2 †	18.0	71.3	23.6	18.5	71.8	24.3	17.2	70.5	22.6
Light Wider MNV2	24.5	73.2	31.5	26.9	73.7	34.6	20.9	72.5	26.9
Wider MNV2	28.1	75.3	35.8	30.8	75.7	39.1	24.1	74.6	30.9
Xception-71	34.3	77.1	43.1	37.5	77.5	46.8	29.6	76.4	37.4

Table 11: DeeperLab performance on the COCO test-dev set. The numbers on the test-dev set are very close to that on the validation set (Tab. 10). †: input size is downsampled by 2 (321×321).

close to that on the validation set (Tab. 10).

D. Performance on Mapillary Vistas

Fig. 13 shows the extra qualitative results.

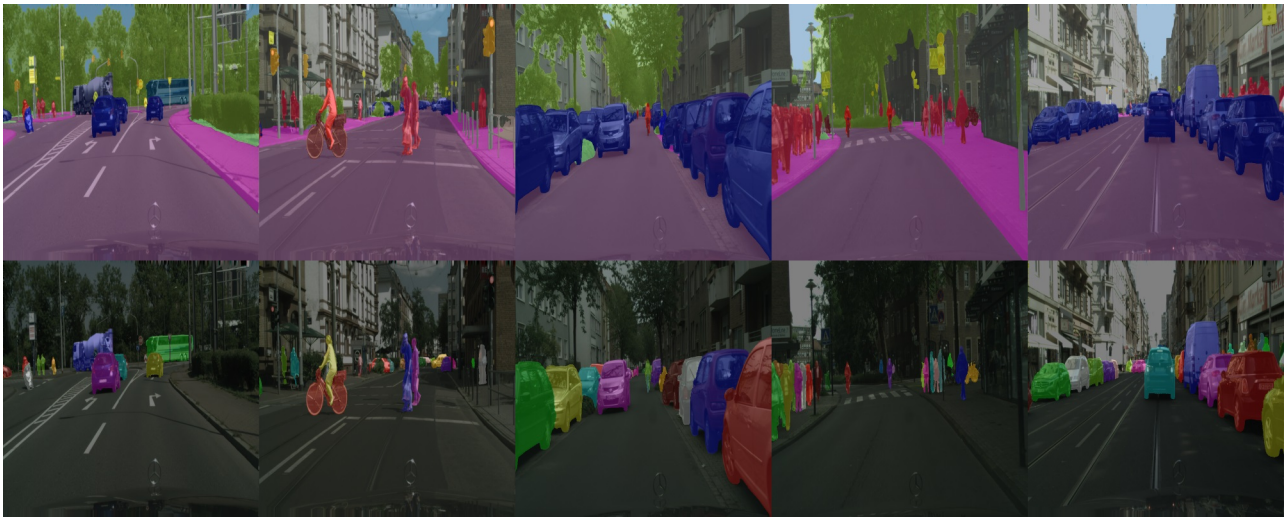

Figure 7: A few image parsing results overlaid on the original images on the Cityscapes validation set with the proposed DeeperLab based on Xception-71. The first row is the predicted semantic segmentation and the second row is the predicted instance segmentation. Note that our model does not generate any VOID labels.

Figure 8: A few image parsing results overlaid on the original images on the Pascal VOC 2012 validation set with the proposed DeeperLab based on Xception-71. The first row is the predicted semantic segmentation and the second row is the predicted instance segmentation. Note that our model does not generate any VOID labels.

Figure 9: A few image parsing results overlaid on the original images on the COCO validation set with the proposed DeeperLab based on Xception-71. The first row is the predicted semantic segmentation and the second row is the predicted instance segmentation. Note that our model does not generate any VOID labels.

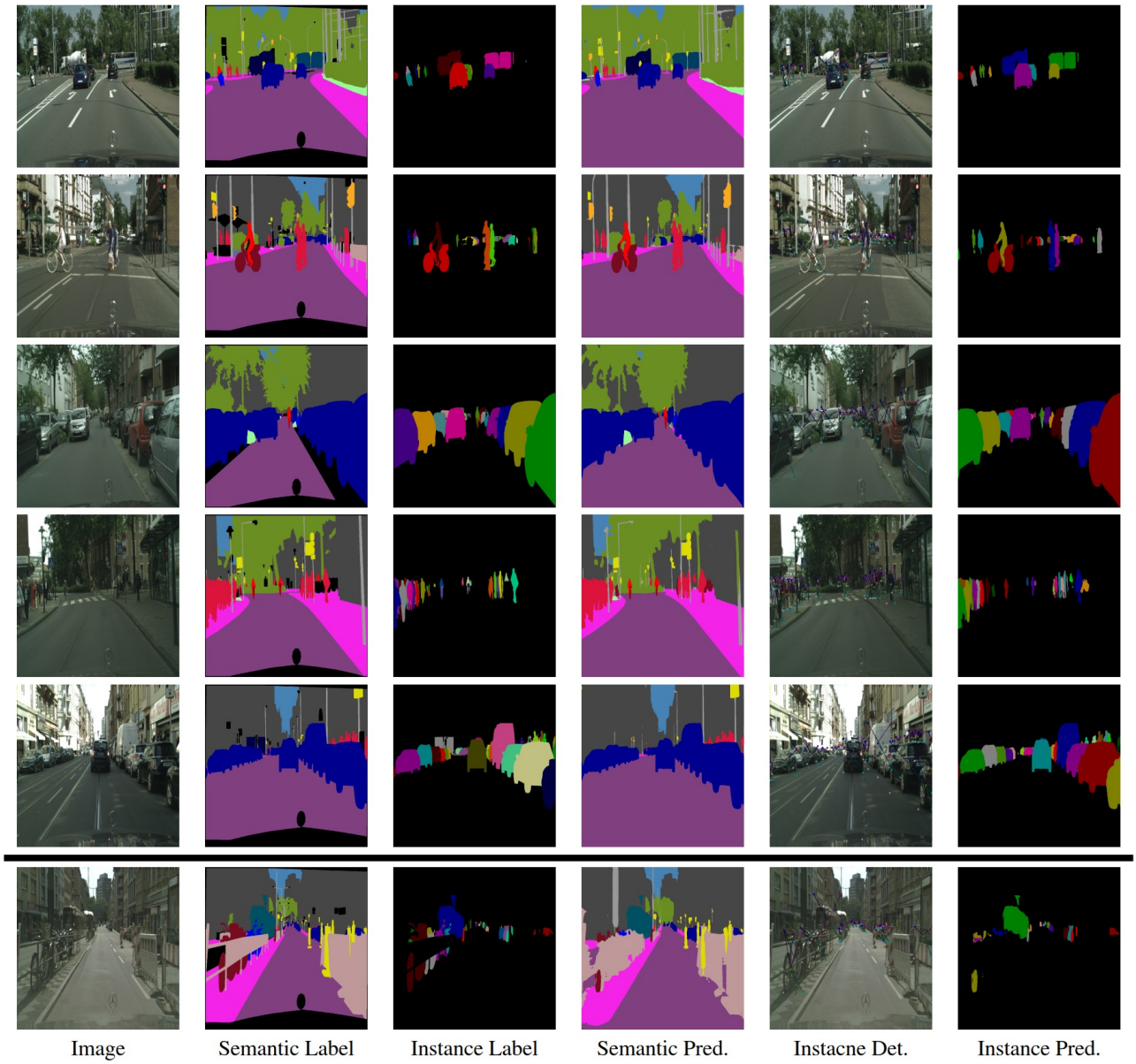

Figure 10: A few image parsing results compared with the ground-truth labels on the Cityscapes validation set with the proposed DeeperLab based on Xception-71. The last row is a failure case. Note that the 'Instance Detection' column shows the raw outputs of the keypoints and middle-range offsets of the detected instances, which will be further refined in the later stage.

Figure 11: A few image parsing results compared with the ground-truth labels on the Pascal VOC 2012 validation set with the proposed DeeperLab based on Xception-71. The last row is a failure case. Note that the 'Instance Detection' column shows the raw outputs of the keypoints and middle-range offsets of the detected instances, which will be further refined in the later stage.

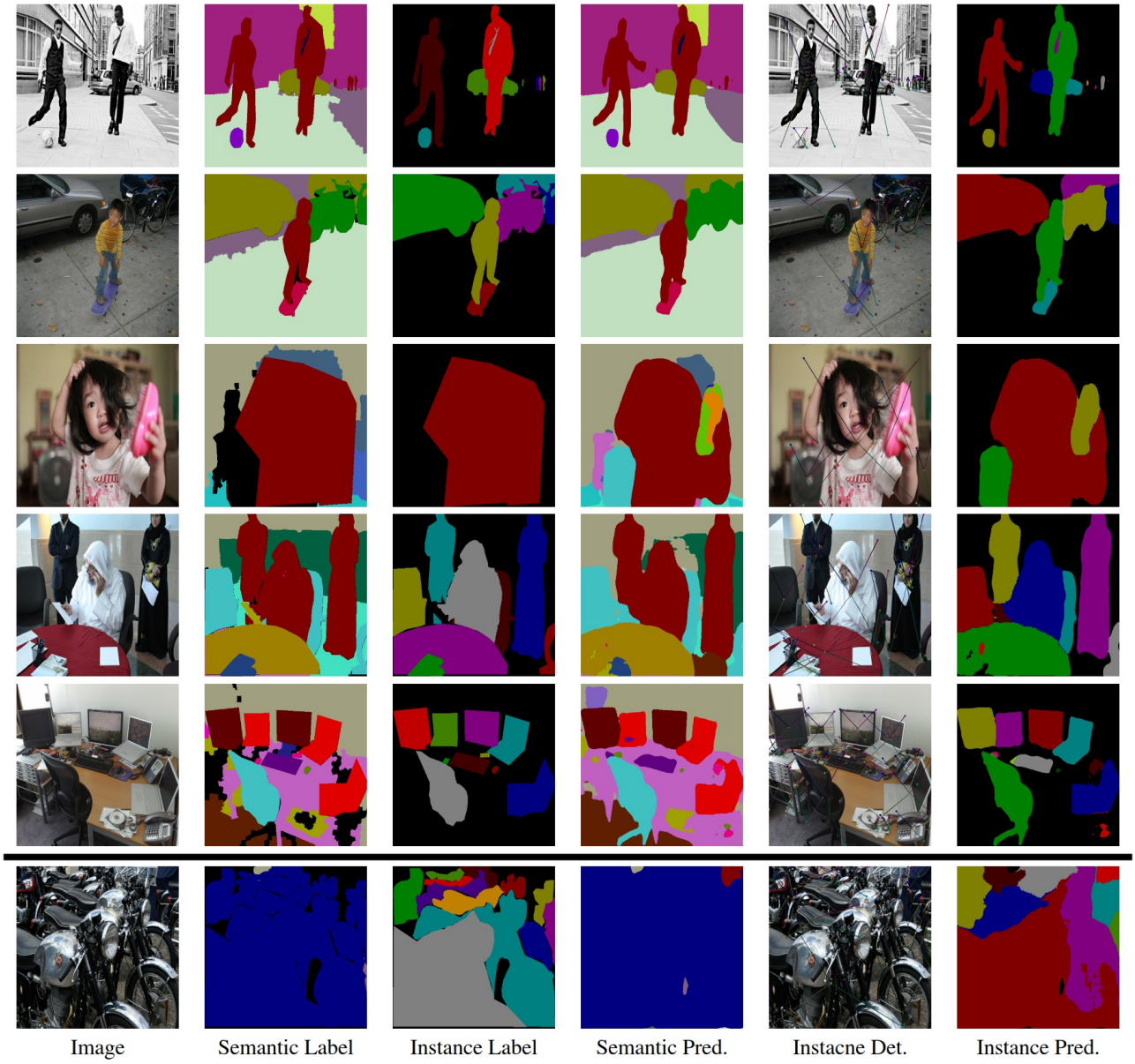

Figure 12: A few image parsing results compared with the ground-truth labels on the COCO validation set with the proposed DeeperLab based on Xception-71. The last row is a failure case. Note that the 'Instance Detection' column shows the raw outputs of the keypoints and middle-range offsets of the detected instances, which will be further refined in the later stage.

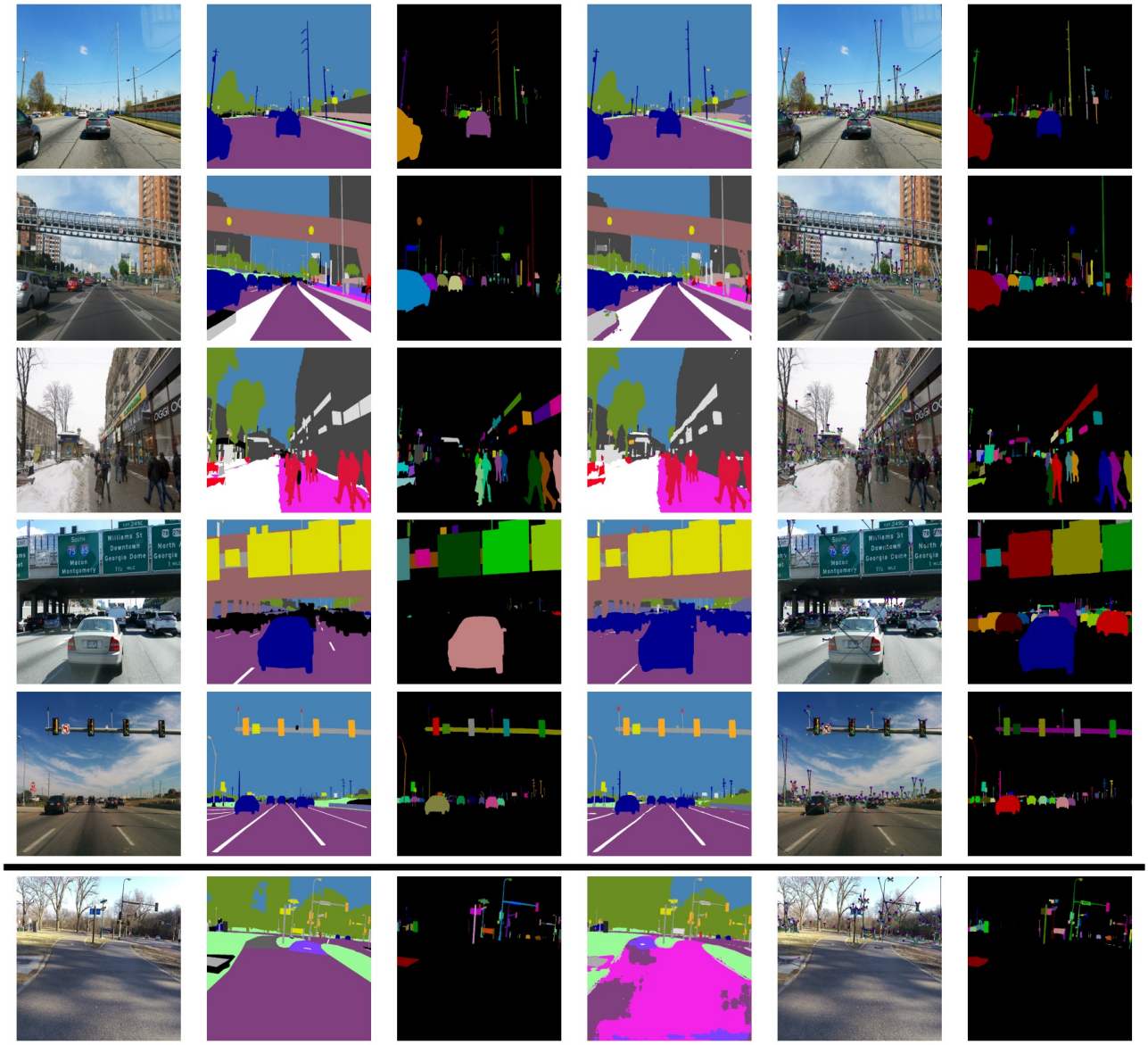

Figure 13: A few image parsing results compared with the ground-truth labels on the Mapillary Vistas validation set with the proposed DeeperLab based on Xception-71. The last row is a failure case. Note that the 'Instance Detection' column shows the raw outputs of the keypoints and middle-range offsets of the detected instances, which will be further refined in the later stage.