Pixel Consensus Voting for Panoptic Segmentation

Haochen Wang
Carnegie Mellon University
whcw@cmu.edu

Ruotian Luo
TTI-Chicago
rluo@ttic.edu

Michael Maire
University of Chicago
mmaire@uchicago.edu

Greg Shakhnarovich
TTI-Chicago
greg@ttic.edu

Abstract

The core of our approach, Pixel Consensus Voting, is a framework for instance segmentation based on the Generalized Hough transform. Pixels cast discretized, probabilistic votes for the likely regions that contain instance centroids. At the detected peaks that emerge in the voting heatmap, backprojection is applied to collect pixels and produce instance masks. Unlike a sliding window detector that densely enumerates object proposals, our method detects instances as a result of the consensus among pixel-wise votes. We implement vote aggregation and backprojection using native operators of a convolutional neural network. The discretization of centroid voting reduces the training of instance segmentation to pixel labeling, analogous and complementary to FCN-style semantic segmentation, leading to an efficient and unified architecture that jointly models things and stuff. We demonstrate the effectiveness of our pipeline on COCO and Cityscapes Panoptic Segmentation and obtain competitive results. Code will be open-sourced.

1. Introduction

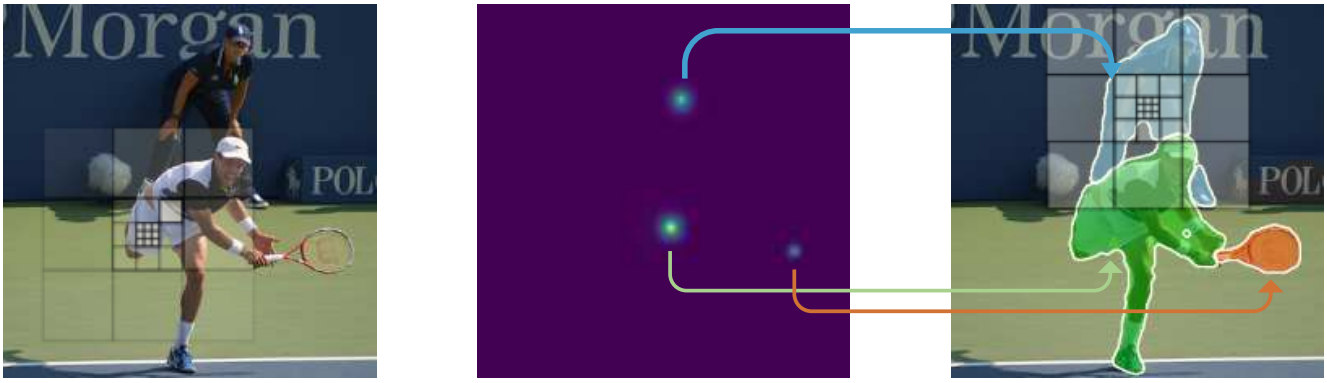
The development of visual recognition algorithms has followed the evolution of recognition benchmarks. PASCAL VOC [13] standardizes the task of bounding box object detection and the associated IoU/Average Precision metrics. At the time, the approaches defining the state-of-the-art, DPM [14] and later the R-CNN family [17, 48], address object detection by reasoning about densely enumerated box proposals, following the sliding window classification approach of earlier detectors [52, 51]. SDS [19] expands the scope of object detection to include instance mask segmentation, and introduces early versions of the mAP^{bbox} and mAP^{mask}, subsequently popularized by the COCO dataset [35]. Bounding boxes, however, remain the primary vehicle for object reasoning.

The more recently introduced task of panoptic segmentation [24] removes the notion of boxes altogether. It treats both "things" and "stuff" in a unified format, in which ground truth and predictions are expressed as labelled segment masks: instance and category for objects ("things"), and category only for "stuff" [5].

Our work focuses on this particular framework of image understanding. We propose an approach, Pixel Consensus Voting (PCV), that in line with this updated task definition, elevates *pixels* to first class citizen status. Every pixel contributes evidence for the presence, identity, and location of an object to which it may belong, as well as for an object or stuff category label. PCV aggregates and backprojects this evidence, in a Hough transform-like framework, so that detections emerge from the *consensus* among pixels that vote for consistent object hypotheses. Figure 1 summarizes our approach.

Notably, and in contrast to the currently dominant line of detection work derived from the R-CNN family [17, 48, 20, 7], our approach does not involve reasoning about bounding boxes. It can be seen as a descendant of early Hough-based methods such as the Implicit Shape Model (ISM) [27, 28]. Such methods traditionally have been referred to as "bottom-up". Different from earlier work, PCV leverages the power of convolutional networks to extract rich image features. Since these features have large receptive fields and presumably capture high-level semantic concepts, it is unclear whether the bottom-up designation remains appropriate. Another distinction from prior attempts to use voting is in our representation of those votes. Traditional methods treat voting as offset regression, and suffers from the problem of "regressing to the mean", where prediction is conflated with uncertainty. In PCV, we treat voting for object location as classification over discretized spatial cells. This allows for the representation of uncertainty and for "abstention" votes by non-object pixels. It also lends itself to efficient vote aggregation and backprojection using (dilated [56]) convolutional mechanisms.

Despite its simplicity, PCV achieves competitive quantitative results on COCO and Cityscapes panoptic segmentation benchmarks. On COCO, PCV outperforms all existing proposal-free or single-stage detection methods. Our work revisits a classic idea from a modern perspective, and we expect that future research extending our approach will yield major performance gains and novel insights.



Discretization & Classification | Voting as (Transposed) Convolution | Backprojection as Filtering

Figure 1: An overview of PCV. Left: A large region around each pixel is discretized into spatial cells according to a *Voting Filter*, with cells whose size increases the farther they are from the pixel. A classification convnet casts votes for the location of the instance to which the pixel belongs, in the form of a probability of a cell containing the instance centroid; a pixel can also vote for "abstaining" (not belonging to any instance). Middle: The votes from each pixel are aggregated into a voting heatmap, using an efficient dilated deconvolution (transposed conv.) mechanism. *Peak Regions* of the heatmap are treated as initial instance detection hypotheses. Right: A *Query Filter*, the spatial inversion of the *Voting Filter*, is convolved with each peak region, backprojecting from the vote map to produce a pixel-wise instance mask for that peak. Not shown: the semantic segmentation branch assigns categories to things and stuff, and filters out spurious detection on the left. See also Fig. 7.

2. Related work

Sliding window detection Over the last two decades, most approaches to object detection and instance segmentation have followed the general methodology of sliding window classification. Early work on densely evaluating windows was later refined to enumerating and evaluating select proposals. This is by far the most prevalent approach today. In a typical pipeline, a large number of candidate regions are sampled from an input image and a predictor (usually a convolutional network) scores each region's likelihood to intersect with objects. For highly ranked proposals, the network also predicts their categories, bounding box coordinates, and optionally generates instance masks. Two stage methods, such as Faster/Mask R-CNN [48, 20] use regional feature pooling as an attention mechanism to enhance prediction accuracy, while single stage methods, including YOLO [46], RetinaNet [34] and SSD [37], combine all network decisions in a single feedforward pass.

Instances from pixels Many attempts have been made to establish a direct connection between image pixels and instance segmentations. A natural idea to group pixels into instances is to obtain some measure of pixel affinity for clustering. In [10, 42], a network is trained to produce pixel embeddings that are similar within and different across instances, and an off-the-shelf clustering algorithm is used for grouping. RPE [26] integrates the clustering step into the learning process by formulating mean-shift clustering as a recurrent neural network. AdaptIS [50] further improves the training of discriminative embeddings by a novel scheme that provides end-to-end supervision. In addition, exploiting the sparsity in pairwise pixel affinity makes it possible to instantiate the clustering step as a graph-cut. Learning the sparse pixel affinity can be formulated as boundary detection [25], or a richer multi-hop dilated connectivity prediction [16, 38]. Alternatively, instances may be generated sequentially using a recurrent neural network [47, 49], or treated as watershed basins through a learned boundary distance transform [2].

Detections via Generalized Hough transform The Hough transform [12] frames the task of detecting analytical shapes as identifying peaks in a dual parametric space; this idea can be generalized [3] to arbitrary objects. The gist of the Generalized Hough transform is to collect local evidence as *votes* cast for the likely location, scale, and pose of potential instances. Works in this vein such as Implicit Shape Models [27] rely on memorized mapping from image patches to offsets, and is later improved by the use of more advanced learning techniques [40, 15]. It has been applied to a variety of problems including pose estimation [18, 4], tracking [15] and 3D object detection [45]. Our work can be seen as a descendant of these earlier efforts.

Some recent work follows broadly similar philosophy, but differs from ours in many ways. Most [41, 44, 55] treat learning to vote for object centroids as a regression task, followed by clustering votes represented as points in parameter space. Our work avoids potential limitations of offset regression and uses discretized region classification to capture pixel-level uncertainty. We design convolutional mechanisms for efficient vote aggregation and backprojec-

tion under this classification setting.

To our knowledge, the only prior work using transposed convolution *i.e*. deconv for classification-based pixel voting is [32], applied to single person pose estimation. We take inspiration from their work, but differ in motivation and implementation.

Panoptic Segmentation Most existing work address panoptic segmentation by merging the outputs from specialized components designed for instance [20] and semantic segmentation [57, 6] with greedy heuristics [24]. PFPN [23] establishes a strong single network baseline by sharing the FPN [33] feature for Mask R-CNN [20] and FCN [39] sub-branches. [54, 29, 30, 36] improve the segment overlap resolution with learned modules. [53, 11] trade off the performance for speed by using single-stage object detectors. Proposal-free methods [50, 16, 55, 8] provide novel perspectives that directly model instances from pixels, but in general lag behind the performance of those leveraging mature engineering solutions, with an especially large gap on the challenging COCO [35] benchmark.

3. Pixel Consensus Voting

Given an input image, PCV starts with a convolutional neural network extracting a shared representation (feature tensor) that is fed to two independent sub-branches (Fig. 2). The semantic segmentation branch predicts the category label for every pixel. The instance voting branch predicts for every pixel whether the pixel is part of an instance mask, and if so, the relative location of the instance mask centroid. This prediction is framed as classification over a set of grid cells around a pixel according to the *Voting Filter*. Both branches are trained with standard cross-entropy loss.

The predictions from the voting branch are aggregated into a voting heatmap (*accumulator array* in the Hough transform terminology). A key technical innovation in PCV is a dilated convolution mechanism that implements this efficiently. Local maxima of the heatmap are detection candidates. At each *peak region*, we convolve a *Query Filter* to backproject the pixels that favor this particular peak above all others. These pixels together form a category-agnostic instance segmentation mask. Finally, we merge the instance and semantic segmentation masks using a simple greedy strategy, yielding a complete panoptic segmentation output.

3.1. Backbone and Feature Extraction

Our work develops a meta architecture to model and segment instances. To this end, PCV reduces the training of instance recognition to pixel labelling, which can be tackled by various descendants of Fully Convolutional Networks [39]. We follow the design of UPSNet [54], which repurposes a Feature Pyramid Network (FPN) [33] with a ResNet backbone [21] for semantic segmentation. Features from each stage of FPN, respectively at $1/32, 1/16, 1/8$ and $1/4$ of input resolution, first go through a shared deformable convolution module before being upsampled to a uniform size of $1/4$ of the input scale. Channel dimensions of the feature maps are reduced from 256 to 128 with 1×1 conv before channel-wise concatenation. On top of this, we apply a 1×1 conv, softmax and $4\times$ nearest neighbor upsampling to generate per-pixel labels. Note that we apply softmax first, before upsampling, since it is faster to produce instance masks at a lower resolution. The semantic segmentation branch predicts the labels for all categories, and is different from PFPN [23], which lumps all "thing" classes into a single category.

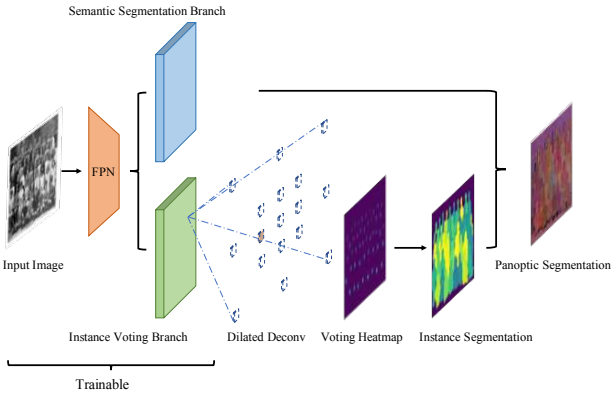


Figure 2: Network architecture for PCV. FPN serves as a shared extractor for the semantic segmentation branch and instance voting branch. Each branch predicts output at every pixel, and is trained with a per-pixel cross entropy loss.

3.2. Region Discretization

Consider an instance mask consisting of a set of pixels $\{p_i \,|\, p_i \in \mathbb{R}^2\}_{i=1}^N$, and the instance center of mass $c = \frac{1}{N}\sum p_i$. Predicting the relative offset $\delta_i = c - p_i$ from a pixel is typically treated as offset regression [31, 41, 55, 43]. But a direct regression limits the ability of the system to represent uncertainty, and suffers from the typical problem of "regressing to the mean". A pixel unsure about its instance centroid location might hedge by pointing between multiple candidates, creating spurious peaks and false positives. Moreover, it is impossible to attribute such pixels to object hypotheses during backprojection. We instead frame voting as classification among possible spatial cells where the centroid may reside. The probability histogram produces an explicit distribution for downstream reasoning.

Voting Filter Unlike YOLO [46], which divides up the entire image into a grid of regularly tiled patches, we consider a discretization of the region centered around each pixel. See Fig. 1 (left) for a visual illustration. Consider-

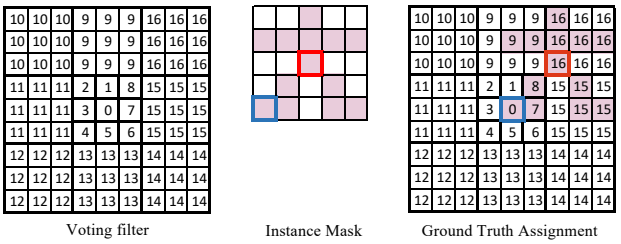


Figure 3: *Voting Filter* and ground truth assignment. Left: a toy *Voting Filter* mapping a $M \times M$, $M = 9$ region around a pixel to $K = 17$ indices. The discretization is coarser on the periphery, using 3×3 cells. Middle: an instance mask where the red pixel is the mask centroid. We need to discretize the offset from the blue pixel to the centroid. Right: overlaying the *Voting Filter* on top of the blue pixel, one sees that the ground truth voting index for the blue pixel is 16.

ing a particular pixel p_i, we map each of the $M \times M$ pixels centered around p_i to K discrete indices. This mapping can be naturally recorded with a translation invariant lookup table of size $M \times M$. By overlaying the lookup table on top of p_i, the ground truth index for classification can be directly read off from the spatial cell into which the instance centroid falls. We refer to this lookup table as the *Voting Filter*. Fig. 3 shows a toy example. For stuff pixels that do not belong to any instances, we create an "abstention" label as an extra class, and hence there are in total $K+1$ classes for the instance voting branch. If the instance centroid falls outside the extent of the *Voting Filter*, *i.e.* the pixel is too far away from the centroid, we ignore it during training.

Scale *vs.* Accuracy Discretization implies a loss of spatial accuracy, but we argue that knowing the exact location of the centroid is not necessary for accurate instance segmentations. What matters is the consensus among pixels that enables backprojection. Large instances can naturally tolerate coarser predictions than small objects, as seen in Fig. 5. We construct the *Voting Filter* so that the farther the distance from the instance centroid, the larger the spatial cell. A naive evenly spaced grid would have to either be too fine, introducing too many classes for the network to learn and converge, or too coarse to allow accurate predictions for smaller objects. Based on these considerations, we propose a grid of square cells whose size expands radially outward, as shown in Fig. 4. It involves $K = 233$ cells over a region of $M = 243$ pixels applied to images at $1/4$ input resolution, thus covering up to 972×972 at full resolution.

3.3. Voting as Transposed Convolution

The instance voting branch yields a tensor of size $[H, W, K+1]$, where $K+1$ is the number of distinct voting possibilities, including abstention by stuff pixels. We

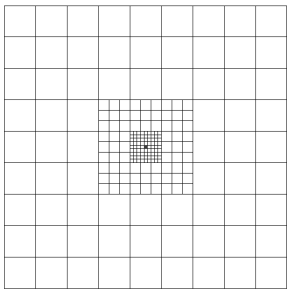


Figure 4: The grid structure of our voting and query filters. It covers 243×243 pixel area and consists of 233 bins ranging in size of $1, 3, 9, 27$ from center to the periphery.

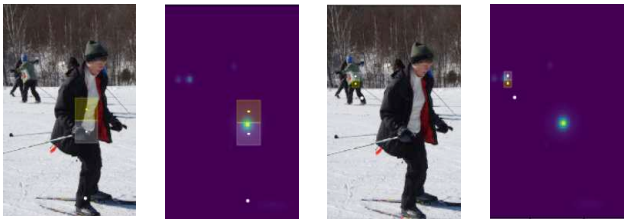


Figure 5: Illustration of voting behavior. The white dot indicates a pixel whose votes we inspect. The cell receiving its highest vote is shown as a white box, second highest as a yellow box. Left: pixels distant from the object center (like the front skier's foot) can afford more slack/spatial uncertainty, voting for larger cells near grid perimeter. Right: pixels closer to object center (and in particular any pixel of a small object, like the far away skier) need more spatial accuracy, voting for small cells near the grid center.

use dilated deconvolution and average pooling to aggregate the probabilistic votes to their intended spatial locations.

Recall our toy example in Fig. 3. Say the blue pixel predicts a probability of 0.9 for its instance centroid to fall into cell 16, which consists of 9 pixels. Voting involves two steps: 1) transfer the probability 0.9 to cell 16, and 2) share the vote evenly among the 9 constituent pixels, with each pixel receiving 0.1. We implement step 1 with dilated deconvolution (deconv), and step 2 with average pooling.

Transposed convolution, or deconvolution by convention, can be understood as the backward pass of convolution. A conv kernel aggregates spatial information to a single point, whereas a deconv kernel spreads a point signal across multiple spatial locations. It is most often used for feature upsampling, where the parameters of the kernel are learned. For the purpose of vote aggregation, however, we fix the deconv kernel parameters to 1-hot across each channel that marks the target location. Dilation in this case enables a pixel to cast its vote to faraway points.

Our toy *Voting Filter* in Fig. 3 discretizes the 9×9 region into inner 3×3 cells of side length 1, encircled by outer 3×3 cells of side length 3, hence $K = 9 + 8 = 17$ voting classes. At step 1, after discarding abstention votes, we split the $[H, W, 17]$ tensor along channels into two components of size $[H, W, 9]$ and $[H, W, 8]$, and apply two deconv kernels of size $[C_{\text{in}}{=}9, C_{\text{out}}{=}1, H{=}3, W{=}3]$ with dilation 1

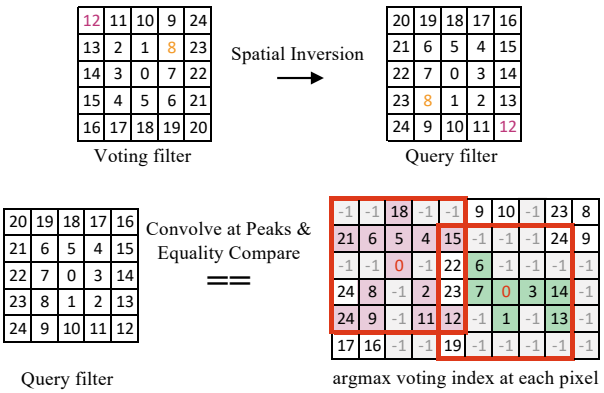


Figure 6: *Query Filter* and backprojection. Top: The *Query Filter* is a spatial inversion of the *Voting Filter*, and the indices of the two filters are symmetric about the center (highlighted here for a few corresponding pairs of cells in the two filters). The *Voting Filter* captures the spatial relationship between a pixel and the surrounding centroid, whereas the *Query Filter* represents the dual relationship between a centroid and surrounding pixels; Bottom: The *Query Filter* is convolved within each *peak region* to produce instance masks. For simplicity a *peak region* here is a single red point, but is in general a connected component of pixels, and hence the need for convolving the *Query Filter*. -1 denotes pixels whose argmax voting decision is abstention *i.e.* "stuff" pixels. Votes that are not cast for a peak region (where the votes and the *Query Filter* disagree) are ignored.

and $[C_{\text{in}}{=}8, C_{\text{out}}{=}1, H{=}3, W{=}3]$ with dilation 3 to produce two heatmaps $H_{dilate1}, H_{dilate3}$, both of size $[H, W, 1]$.

After step 1, all the votes have been sent to the center of each spatial cell. At step 2, we smooth out the votes evenly within each cell. Smoothing in this particular case is exactly equivalent to average pooling. We apply 3×3 average pooling on $H_{dilate3}$, and 1×1 average pooling on $H_{dilate1}$ (an identity operation). The two heatmaps are then summed together to complete the final voting heatmap. The voting process for other instantiations of a *Voting Filter* can be done analogously.

Peaks in the voting heatmap correspond to consensus detections, and we use a simple strategy of thresholding followed by connected components to locate the peaks. We define a *peak region*, which identifies a hypothesized instance, as a connected component of pixels that survive after thresholding the voting heatmap. We set the threshold value to 4.0 for both COCO and Cityscapes. See Fig. 7.

3.4. Backprojection as Filtering

Backprojection aims to determine for every *peak region* the pixels that favor that particular maximum above all others. To do this we make use of the *Query Filter*. Recall that the *Voting Filter* records the class label a pixel aligned at the center of the filter should predict given possible centroid locations in the surrounding region. The query filter is the spatial inversion of the *Voting Filter*. It records the class labels the *surrounding* pixels should have predicted in order to have voted for the instance centroid being at the center of the filter placement. This dual relationship is shown in the top row of Fig. 6.

During backprojection, we first obtain the argmax voting index at each pixel. This is a tensor of size $[H, W, 1]$. Then, within a *peak region*, we convolve the *Query Filter* and perform equality comparison against the argmax voting indices to pick up all the pixels whose strongest vote falls within this *peak region*. See Fig. 6, bottom row. This operation is parallelizable and can be implemented to run on a GPU. In practice, we extend the equality comparison to top 3 votes rather than just the argmax vote, so that a pixel whose argmax decision is wrong is not completely abandoned. If a single pixel is contested by multiple peaks, the pixel is assigned to the *peak region* whose total vote count is the highest. In the edge case where multiple peaks are within the same spatial cell with respect to the pixel, the pixel goes to the spatially nearest peak (this distance is measured from the center of the enclosing bounding box of the *peak region* to the pixel).

3.5. Segment Loss Normalization

Training a network to solve pixel labeling problems such as ours usually involves averaging the per-pixel cross entropy loss over an image [39]. Each pixel contributes equally to the training and the notion of instance is absent. This is often the case for semantic segmentation, since the annotations specify only the categories. For Panoptic Segmentation, however, each instance segment is given equal weight during evaluation and training with the default pixel-averaged loss would put emphasis primarily on large instances, neglecting small objects that are numerous and critical. Therefore, we need to design an objective function that balances the loss across instances. Let a_i denote the area of the mask segment to which a pixel p_i belongs. The training losses for both semantic and voting branches are normalized to be

$$L = \frac{1}{\sum_i w_i} \sum_i w_i \log p(y_i|p_i) \qquad (1)$$

where y_i is the ground truth semantic/voting label, and $w_i = \frac{1}{a_i^\lambda}$. λ controls the strength of normalization. When $\lambda = 0$, $w_i = 1$, we get back the default pixel-averaged loss. When $\lambda = 1$, we divide the summed loss from each segment by the segment area, so that all segments would contribute equally to the training. $\lambda = 0.5$ could be interpreted as a length based normalization that strikes a middle ground between pixel-averaged loss and full segment normalized loss. Note that stuff and thing segments are treated identically. The final loss is the sum of semantic segmentation loss and voting loss $L_{\text{total}} = L_{\text{sem}} + L_{\text{vote}}$. In Sec. 4, we demonstrate through ablation experiments that segment

loss normalization significantly improves performance on both COCO and Cityscapes.

3.6. Determining Object Categories

Once an instance mask is obtained from backprojection, we predict its category by taking the majority decision made by the semantic segmentation branch in the masked region. This strategy is similar to the one used by [55].

4. Experiments

We report results on COCO [35] and Cityscapes [9] Panoptic Segmentation. Since PCV formulates centroid prediction as region classification rather than offset regression, it trades off the upper bound on prediction accuracy for a richer representation. We first conduct oracle experiments to understand the potential of our system. Then we compare our model performance on COCO and Cityscapes validation sets against prior and concurrent works. Ablations focus on the use of different discretization schemes and segment loss normalizations.

4.1. Setup

COCO Panoptic Segmentation consists of 80 thing and 53 stuff categories. We use the 2017 split with 118k training images and report results on `val` and `test-dev`. Cityscapes includes images of urban street scenes. We use standard `train`/`val` split, including 2975 and 500 images, respectively. There are 19 categories, 11 stuff and 8 things. We measure performance by Panoptic Quality (PQ) [24]. PQ can be interpreted as a generalized F1-score that reflects both recognition quality RQ and segmentation quality SQ. In addition to the overall PQ, we include PQ^{th} and PQ^{st} and focus in particular on thing category performance.

4.2. Oracles

There are no learnable parameters in the vote aggregation and backprojection steps of PCV, and so once the pixel-wise classification decisions are made by the backbone network, the subsequent inference is deterministic. Therefore we perform oracle experiments that feed into the inference pipeline ground truth classification labels for both the voting and semantic branches. As seen in Table 1, given our default discretization scheme, PCV oracle achieves performance close to the upper bound on both COCO and Cityscapes validation sets. The remaining gaps in PQ are mostly due to small instances of extremely high occlusion and instances with colliding centroids. We also show 2 more oracle results: a simple radially expanding grid with 41 voting classes performs worse than the default grid with 233 voting classes. A uniform grid with evenly-spaced bins of size 15 and total *Voting Filter* side length 225 does the worst. Even though it has roughly the same number of voting classes as our default grid, the even spacing severely degrades performance on small instances.

		PQ	SQ	RQ	PQ^{th}	SQ^{th}	RQ^{th}	PQ^{st}
COCO	1/4 gt	**92.5**	93.2	**99.2**	**90.6**	91.6	**98.8**	95.3
	default grid	90.1	93.0	96.8	86.6	91.3	94.8	95.3
	simple grid	79.2	92.6	85.1	68.6	90.7	75.4	95.3
	uniform grid	67.1	**95.8**	70.1	49.4	**96.0**	51.5	93.8
Cityscapes	1/4 gt	**89.4**	89.8	**99.6**	**87.1**	87.7	**99.4**	91.0
	default grid	88.6	89.7	98.8	85.4	87.4	97.6	91.0
	simple grid	83.0	89.3	92.8	72.1	86.6	83.4	91.0
	uniform grid	66.1	**92.6**	71.8	31.8	**94.3**	33.4	91.0

Table 1: Oracle inference on COCO and Cityscapes `val` using ground truth voting and semantic classification label. '1/4 gt' is the performance upper bound when the output is at 1/4 of input resolution, and the default discretization is behind by a small gap.

4.3. Main Results and Ablations

For Cityscapes training we use batch size 16 over 8 GPUs and crop input images to a uniform size of 1536×1024. We apply random horizontal flipping and randomly scale the size of the crops from 0.5 to 2. The model is trained for 65 epochs (12k iterations) with learning rate set initially at 0.01, dropped by 10x at 9000 iterations. We use SGD with momentum 0.9 and weight decay set to 1e-4.

For COCO, we use standard Mask R-CNN $1\times$ training schedule and hyperparameters. Input images are resized to have length 800 on the shorter side and length not exceeding 1333 on the longer. Resizing is consistent for both training and testing. Left-right flipping is the only data augmentation used. We use SGD with momentum 0.9 and set the initial learning rate at 0.0025, weight decay at 0.0001. The model is trained on 8 GPUs with batch size of 16 for a total of around 13 epochs (90k iterations). Learning rate decays by a factor of 10 at 60k and 80k iterations. BatchNorm [22] layers in ResNet are frozen in our current setup.

Following [23, 54], for stuff predictions we filter out small predicted segments to reduce false positives. The thresholds are set at areas of 4096 pixels for COCO and 2048 pixels for Cityscapes.

Main Results We compare the performance of PCV against representative methods using different approaches. On both COCO and Cityscapes, PCV still lags behind leading methods that leverage Mask RCNN for instance segmentation. On the challenging COCO benchmark, PCV outperforms all other proposal free methods as well as [53] which uses RetinaNet for object detection. Results on Cityscapes are shown in Table 4. Qualitative results are displayed in Fig 8 and 9.

Ablation: discretization We explore the influence of discretization by comparing a model using a simple 40-cell

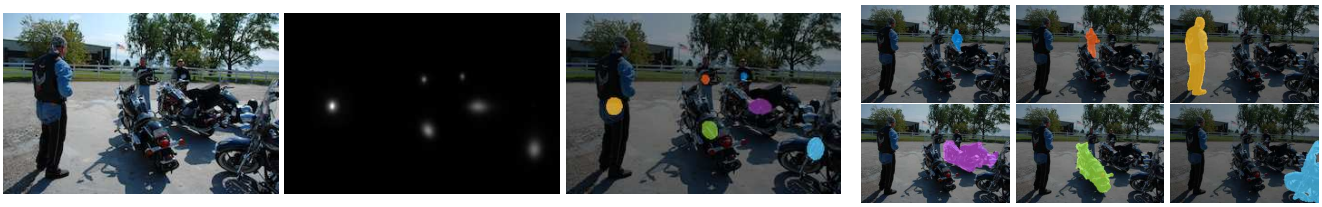


Figure 7: Illustration of instance mask inference in PCV. Left to right: input image, voting heatmap, detected peak regions (random colors assigned to each peak); six masks resulting from backprojection from color-matching regions.

	PQ	SQ	RQ	PQ^{th}	SQ^{th}	RQ^{th}	PQ^{st}
λ=0	32.9	77.2	40.5	33.0	78.0	40.3	32.6
λ=0.5	**37.5**	**77.7**	**47.2**	**40.0**	**78.4**	**50.0**	**33.7**
λ=1.0	31.8	74.7	41.3	33.9	75.1	44.0	28.6

(a) **Segment Loss Normalization:** results on COCO `val` with various λ that controls the normalization strength. $\lambda = 0.5$ improves PQ^{th} by 7 points over the commonly used pixel-averaged loss.

	PQ	SQ	RQ	PQ^{th}	SQ^{th}	RQ^{th}	PQ^{st}
default grid	37.5	77.7	47.2	40.0	78.4	50.0	33.7
simple grid	33.3	77.2	41.8	32.8	77.4	40.9	34.1

(b) **Impact of Discretization:** consistent with oracle results in Table 1, the simple grid is too coarse for accurate localizations and the default grid leads on PQ^{th} by 7.17 points on COCO `val`.

Table 2: Ablations on segment loss normalization and impact of discretization.

	Methods	Backbone	Split	PQ	SQ	RQ	PQ^{th}	SQ^{th}	RQ^{th}	PQ^{st}	SQ^{st}	RQ^{st}
Mask R-CNN	PFPN [23] (1x)	ResNet 50	val	39.4	77.8	48.3	45.9	80.9	55.4	29.6	73.3	37.7
	PFPN [23] (3x)	ResNet 50	val	41.5	79.1	50.5	48.3	82.2	57.9	31.2	74.4	39.4
	UPSNet [54] (1x)	ResNet 50	val	42.5	78.0	52.5	48.6	79.4	59.6	33.4	75.9	41.7
Single Stage Detection	SSPS [53]	ResNet 50	val	32.4	-	-	34.8	-	-	28.6	-	-
	SSPS [53]	ResNet 50	test-dev	32.6	74.3	42.0	35.0	74.8	44.8	29.0	73.6	37.7
Proposal-Free	AdaptIS [50]	ResNet 50	val	35.9	-	-	40.3	-	-	29.3	-	-
	DeeperLab [55]	Xception 71	val	33.8	-	-	-	-	-	-	-	-
	DeeperLab [55]	Xception 71	test-dev	34.3	77.1	43.1	37.5	77.5	46.8	29.6	76.4	37.4
	SSAP [16]	ResNet 101	val	36.5	-	-	-	-	-	-	-	-
	SSAP [16]	ResNet 101	test-dev	36.9	80.7	44.8	40.1	81.6	48.5	32.0	79.4	39.3
	Ours (1x)	ResNet 50	val	**37.5**	77.7	**47.2**	40.0	78.4	**50.0**	**33.7**	76.5	**42.9**
	Ours (1x)	ResNet 50	test-dev	**37.7**	77.8	**47.3**	**40.7**	78.7	**50.7**	**33.1**	76.3	**42.0**

Table 3: Comparisons on COCO. PCV outperforms proposal-free and single-state detection methods.

	PQ	PQ^{th}	PQ^{st}	mIoU
DIN [1]	53.8	42.5	62.1	80.1
UPSNet [54]	59.3	54.6	62.7	75.2
PFPN [23]	58.1	52.0	62.5	75.7
AdaptIS [50]	59.0	55.8	61.3	75.3
SSAP [16]	58.4	50.6	-	-
Ours	54.2	47.8	58.9	74.1

Table 4: Results on Cityscapes val using ResNet 50

	Input Size	Backbone	Voting	Backproj.	Total
COCO	800×1333	93.4	1.3	81.8	176.5
Cityscapes	1024×2048	115.6	2.8	64.4	182.8

Table 5: runtime benchmark using GTX 1080 Ti (unit: ms)

grid against the default model using a 233-cell grid. The results on COCO val set are presented in Table 2b. The full grid outperforms the simplistic grid and this agrees with our observation made earlier for the oracle experiments. The simple grid might make the learning easier but sacrifices the prediction accuracy due to coarse discretization.

Ablation: segment loss normalization We hypothesize that the contribution from each pixel to the final training loss should be normalized by a function of the segment area so that large instances would not eclipse the attention paid to small objects. We train PCV on COCO with λ set at $0, 0.5, 1$. As expected, pixel-averaged loss with $\lambda = 0$ dilutes the focus on small objects and drags down PQ things, while a full area based segment normalization with $\lambda = 1$ causes severe degradation on stuff PQ. Length-based normalization with λ set at 0.5 achieves the best performance on both things and stuff.

Timing Table 5 examines PCV runtime, benchmarked on a GTX 1080 Ti and averaged over Cityscapes and COCO `val`. Backprojection filtering relies on an unoptimized indexing implementation. PCV runs at a competitive 5fps.

Figure 8: Results of PCV on Cityscapes `val` and COCO `test-dev`.

Figure 9: Results of PCV on images from COCO `val2017`.

5. Conclusion

We propose a novel approach to panoptic segmentation that is entirely pixel-driven. Different from region-based top-down object detection approaches, Pixel Consensus Voting elevates pixels to a first-class role; each pixel provides evidence for presence and location of object(s) it may belong to. It affords efficient inference, thanks to our filtering-based mechanism for vote aggregation and back-projection. PCV is an alternative to region-based methods, that can be trained efficiently using only per-pixel classification losses. It is significantly simpler than current, highly engineered state-of-the-art panoptic segmentation models.

On Cityscapes and COCO panoptic segmentation benchmarks, PCV achieves competitive results, both qualitatively and on standard panoptic segmentation metrics. This is promising given the engineering effort that, since the introduction of R-CNN [17], has been poured into region-proposal driven systems over the past five years. Weighting the scientific potential of a new approach against a backdrop of a highly engineered and optimized established technique is not a task for which benchmarks alone suffice.

Our results demonstrate that the Generalized Hough transform, a historical competitor to the sliding window detection paradigm, is again viable once combined with deep neural networks. This should be a call for future research exploring new ways of interconnecting traditional computer vision techniques with deep learning. For PCV specifically, there is clear potential to explore improved voting and inference protocols. This includes voting in higher dimensions (*e.g.*, scale-space) and alternative models of interaction between instance detection and category assignments.

6. Acknowledgements

We would like to thank Deva Ramanan for discussions and feedback. The work was in part supported by the DARPA L2M award FA8750-18-2-0126, the DARPA GARD award HR00112020003, and AFOSR award FF9950-18-1-0166 (MADlab).

References

[1] Anurag Arnab and Philip HS Torr. Pixelwise instance segmentation with a dynamically instantiated network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 441–450, 2017.

[2] Min Bai and Raquel Urtasun. Deep watershed transform for instance segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5221–5229, 2017.

[3] Dana H Ballard. Generalizing the hough transform to detect arbitrary shapes. *Pattern recognition*, 13(2):111–122, 1981.

[4] Lubomir Bourdev and Jitendra Malik. Poselets: Body part detectors trained using 3d human pose annotations. In *2009 IEEE 12th International Conference on Computer Vision*, pages 1365–1372. IEEE, 2009.

[5] Holger Caesar, Jasper Uijlings, and Vittorio Ferrari. Cocostuff: Thing and stuff classes in context. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1209–1218, 2018.

[6] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2017.

[7] Xinlei Chen, Ross Girshick, Kaiming He, and Piotr Dollár. Tensormask: A foundation for dense object segmentation. *arXiv preprint arXiv:1903.12174*, 2019.

[8] Bowen Cheng, Maxwell D. Collins, Yukun Zhu, Ting Liu, Thomas S. Huang, Hartwig Adam, and Liang-Chieh Chen. Panoptic-deeplab, 2019.

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

[10] Bert De Brabandere, Davy Neven, and Luc Van Gool. Semantic instance segmentation with a discriminative loss function. *arXiv preprint arXiv:1708.02551*, 2017.

[11] Daan de Geus, Panagiotis Meletis, and Gijs Dubbelman. Fast panoptic segmentation network. *arXiv preprint arXiv:1910.03892*, 2019.

[12] Richard O Duda and Peter E Hart. Use of the hough transformation to detect lines and curves in pictures. Technical report, Sri International Menlo Park Ca Artificial Intelligence Center, 1971.

[13] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.

[14] Pedro F Felzenszwalb, Ross B Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *IEEE transactions on pattern analysis and machine intelligence*, 32(9):1627–1645, 2009.

[15] Juergen Gall, Angela Yao, Nima Razavi, Luc Van Gool, and Victor Lempitsky. Hough forests for object detection, tracking, and action recognition. *IEEE transactions on pattern analysis and machine intelligence*, 33(11):2188–2202, 2011.

[16] Naiyu Gao, Yanhu Shan, Yupei Wang, Xin Zhao, Yinan Yu, Ming Yang, and Kaiqi Huang. Ssap: Single-shot instance segmentation with affinity pyramid. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 642–651, 2019.

[17] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[18] Ross Girshick, Jamie Shotton, Pushmeet Kohli, Antonio Criminisi, and Andrew Fitzgibbon. Efficient regression of general-activity human poses from depth images. In *2011 International Conference on Computer Vision*, pages 415–422. IEEE, 2011.

[19] Bharath Hariharan, Pablo Arbeláez, Ross Girshick, and Jitendra Malik. Simultaneous detection and segmentation. In *European Conference on Computer Vision*, pages 297–312. Springer, 2014.

[20] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[21] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[22] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[23] Alexander Kirillov, Ross Girshick, Kaiming He, and Piotr Dollár. Panoptic feature pyramid networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6399–6408, 2019.

[24] Alexander Kirillov, Kaiming He, Ross Girshick, Carsten Rother, and Piotr Dollár. Panoptic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9404–9413, 2019.

[25] Alexander Kirillov, Evgeny Levinkov, Bjoern Andres, Bogdan Savchynskyy, and Carsten Rother. Instancecut: from edges to instances with multicut. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5008–5017, 2017.

[26] Shu Kong and Charless C Fowlkes. Recurrent pixel embedding for instance grouping. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9018–9028, 2018.

[27] Bastian Leibe, Ales Leonardis, and Bernt Schiele. Combined object categorization and segmentation with an implicit shape model. In *Workshop on statistical learning in computer vision, ECCV*, volume 2, page 7, 2004.

[28] Bastian Leibe, Aleš Leonardis, and Bernt Schiele. Robust object detection with interleaved categorization and segmentation. *International journal of computer vision*, 77(1-3):259–289, 2008.

[29] Jie Li, Allan Raventos, Arjun Bhargava, Takaaki Tagawa, and Adrien Gaidon. Learning to fuse things and stuff. *arXiv preprint arXiv:1812.01192*, 2018.

[30] Yanwei Li, Xinze Chen, Zheng Zhu, Lingxi Xie, Guan Huang, Dalong Du, and Xingang Wang. Attention-guided unified network for panoptic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7026–7035, 2019.

[31] Xiaodan Liang, Liang Lin, Yunchao Wei, Xiaohui Shen, Jianchao Yang, and Shuicheng Yan. Proposal-free network

for instance-level object segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2978–2991, 2017.

[32] Ita Lifshitz, Ethan Fetaya, and Shimon Ullman. Human pose estimation using deep consensus voting. In *European Conference on Computer Vision*, pages 246–260. Springer, 2016.

[33] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017.

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

[35] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[36] Huanyu Liu, Chao Peng, Changqian Yu, Jingbo Wang, Xu Liu, Gang Yu, and Wei Jiang. An end-to-end network for panoptic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6172–6181, 2019.

[37] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[38] Yiding Liu, Siyu Yang, Bin Li, Wengang Zhou, Jizheng Xu, Houqiang Li, and Yan Lu. Affinity derivation and graph merge for instance segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 686–703, 2018.

[39] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[40] Subhransu Maji and Jitendra Malik. Object detection using a max-margin hough transform. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1038–1045. IEEE, 2009.

[41] Davy Neven, Bert De Brabandere, Marc Proesmans, and Luc Van Gool. Instance segmentation by jointly optimizing spatial embeddings and clustering bandwidth. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8837–8845, 2019.

[42] Alejandro Newell, Zhiao Huang, and Jia Deng. Associative embedding: End-to-end learning for joint detection and grouping. In *Advances in Neural Information Processing Systems*, pages 2277–2287, 2017.

[43] David Novotny, Samuel Albanie, Diane Larlus, and Andrea Vedaldi. Semi-convolutional operators for instance segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 86–102, 2018.

[44] George Papandreou, Tyler Zhu, Liang-Chieh Chen, Spyros Gidaris, Jonathan Tompson, and Kevin Murphy. Personlab: Person pose estimation and instance segmentation with a bottom-up, part-based, geometric embedding model. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 269–286, 2018.

[45] Charles R Qi, Or Litany, Kaiming He, and Leonidas J Guibas. Deep hough voting for 3d object detection in point clouds. *arXiv preprint arXiv:1904.09664*, 2019.

[46] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271, 2017.

[47] Mengye Ren and Richard S Zemel. End-to-end instance segmentation with recurrent attention. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6656–6664, 2017.

[48] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[49] Bernardino Romera-Paredes and Philip Hilaire Sean Torr. Recurrent instance segmentation. In *European conference on computer vision*, pages 312–329. Springer, 2016.

[50] Konstantin Sofiiuk, Olga Barinova, and Anton Konushin. Adaptis: Adaptive instance selection network. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 7355–7363, 2019.

[51] Régis Vaillant, Christophe Monrocq, and Yann Le Cun. Original approach for the localisation of objects in images. *IEE Proceedings-Vision, Image and Signal Processing*, 141(4):245–250, 1994.

[52] Paul Viola, Michael Jones, et al. Rapid object detection using a boosted cascade of simple features. *CVPR (1)*, 1(511-518):3, 2001.

[53] Mark Weber, Jonathon Luiten, and Bastian Leibe. Single-shot panoptic segmentation. *arXiv preprint arXiv:1911.00764*, 2019.

[54] Yuwen Xiong, Renjie Liao, Hengshuang Zhao, Rui Hu, Min Bai, Ersin Yumer, and Raquel Urtasun. Upsnet: A unified panoptic segmentation network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8818–8826, 2019.

[55] Tien-Ju Yang, Maxwell D Collins, Yukun Zhu, Jyh-Jing Hwang, Ting Liu, Xiao Zhang, Vivienne Sze, George Papandreou, and Liang-Chieh Chen. Deeperlab: Single-shot image parser. *arXiv preprint arXiv:1902.05093*, 2019.

[56] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015.

[57] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2881–2890, 2017.