Task-aligned Part-aware Panoptic Segmentation
through Joint Object-Part Representations

Daan de Geus   Gijs Dubbelman

Eindhoven University of Technology

{d.c.d.geus, g.dubbelman}@tue.nl

Abstract

Part-aware panoptic segmentation (PPS) requires (a) that each foreground object and background region in an image is segmented and classified, and (b) that all parts within foreground objects are segmented, classified and linked to their parent object. Existing methods approach PPS by separately conducting object-level and part-level segmentation. However, their part-level predictions are not linked to individual parent objects. Therefore, their learning objective is not aligned with the PPS task objective, which harms the PPS performance. To solve this, and make more accurate PPS predictions, we propose Task-Aligned Part-aware Panoptic Segmentation (TAPPS). This method uses a set of shared queries to jointly predict (a) object-level segments, and (b) the part-level segments within those same objects. As a result, TAPPS learns to predict part-level segments that are linked to individual parent objects, aligning the learning objective with the task objective, and allowing TAPPS to leverage joint object-part representations. With experiments, we show that TAPPS considerably outperforms methods that predict objects and parts separately, and achieves new state-of-the-art PPS results.

1. Introduction

To fully understand what is depicted in an image, it is important to consider concepts at different levels of abstraction. For rich scene understanding, we should not only recognize foreground objects (*e.g.*, *car*, *human*) and background regions (*e.g.*, *sky*, *ocean*), but also simultaneously identify the parts that constitute the objects (*e.g.*, *car-wheel*, *human-arm*). In a step towards such comprehensive scene understanding, De Geus *et al.* introduced part-aware panoptic segmentation (PPS) [5]. The objective of this computer vision task is (1) to output a segmentation mask and class label for all thing objects and stuff regions in an image like for panoptic segmentation [13] – we call these *object-level segments* – and (2) to simultaneously segment and classify
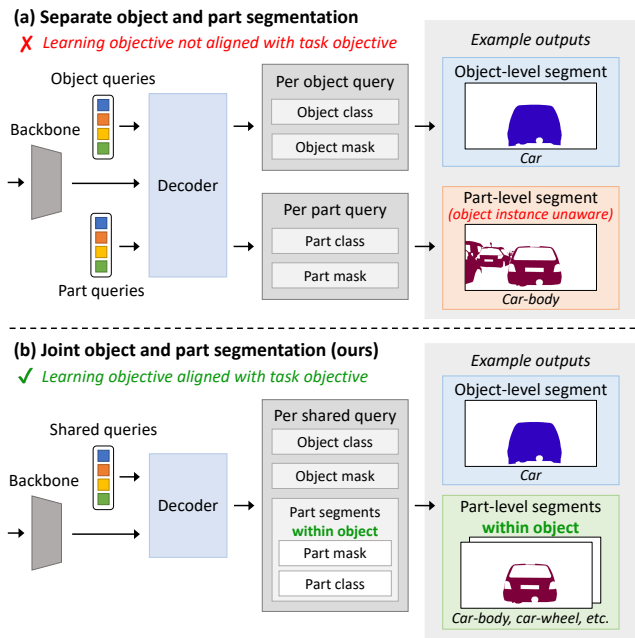

Figure 1. **Task-aligned part-aware panoptic segmentation.** **(a)** Existing works separately predict object-level segments and *object-instance-unaware* part-level segments. **(b)** In this work, we predict objects and parts jointly, using a set of shared queries. This allows our method to predict parts within individual object segments, aligning its learning objective with the PPS task objective.

all parts *within* each identified object. These are called *part-level segments* and should be explicitly linked to an object-level segment, establishing a part-whole relation.

Current state-of-the-art methods [18, 19] address PPS by using two different sets of learnable queries to separately predict object-level and part-level segments, see Fig. 1a. While these methods outperform earlier baselines [5], they have one main limitation: their learning objective is not aligned with the task objective. Where the PPS task objective is to predict parts within each individual object-level segment, these existing works conduct part-level semantic segmentation, predicting part-level masks that cover multiple objects (see Fig. 1a). In other words, these networks

are not optimized for the PPS task, but instead solve the surrogate subtasks of object-level panoptic segmentation and part-level semantic segmentation. As a result, to assign parts to individual objects, these methods require post-processing. Additionally, we hypothesize that there are several other negative consequences: (1) These networks learn a conflicting feature representation. They learn that object-level thing instances should be separated, but also that the parts of these object-level instances should be grouped together. We expect that this harms the ability of the networks to separate instances. (2) Predicting objects and parts separately may cause incompatible predictions (*e.g.*, a *car-window* part with a *bicycle* object), which makes it unclear which prediction to trust, and requires further post-processing. (3) The networks encode information about objects and their parts separately, whereas this information is potentially complementary. See Sec. 3.2.2 for more details on these limitations.

In this work, we aim to design a **simple** network for PPS that overcomes these limitations and thereby makes more accurate PPS predictions. To achieve this, we propose the Task-Aligned Part-aware Panoptic Segmentation (TAPPS) method. Instead of using separate queries for objects and parts, TAPPS uses one set of shared queries to jointly represent objects and the parts they contain. Specifically, each of these queries learns to represent at most one object-level segment, for which it predicts (1) a segmentation mask and object-level class, and (2) the segmentation masks and classes for all part-level segments within this object. This is visualized in Fig. 1b and explained in Sec. 3.

With this approach, TAPPS explicitly predicts part-level segments per individual object. Therefore, the network's learning objective is now aligned with the PPS task objective, and TAPPS is directly optimized for the PPS task. As a result, both object- and part-level segmentation are learned in an 'object-instance-aware' manner. This removes the conflict in the learned feature representations, allowing for better instance separability. Additionally, we hypothesize that predicting objects and parts from the same query reduces incompatibilities between object and part predictions, as they are made using the same information. Moreover, we use TAPPS to go even further, and explicitly only predict part segments compatible with predicted object segments. This enforces full object-part compatibility and simplifies the part segmentation task. Finally, the network can now encode complementary object-level and part-level information in a shared query, giving it a richer representation for making object- and part-level segmentation predictions.

With experiments, detailed in Sec. 4, we show that TAPPS outperforms the baseline with separate object and part queries in multiple aspects: (1) The part segmentation quality within identified objects is significantly better, which shows the positive impact of using joint object-part

representations and simplifying the part segmentation task. (2) The object instance segmentation performance is considerably improved, demonstrating the benefits of learning parts in an object-instance-aware manner. Together, this yields a large overall improvement with respect to the baseline, and causes TAPPS to considerably outperform existing works across different benchmarks, achieving new state-of-the-art results. See Sec. 5 for more extensive results.

To summarize, we make the following contributions:
- We propose TAPPS, a simple approach for PPS that aligns the learning objective with the task objective, facilitating object instance separability and enabling joint object-part representations for more accurate PPS predictions.
- We use the shared object-part queries to constrain TAPPS to only predict part segments that are compatible with the object class, enforcing object-part compatibility and simplifying the part segmentation task.
- We experimentally show the effectiveness of TAPPS across a range of datasets and network configurations.

The code for TAPPS is made publicly available through https://tue-mps.github.io/tapps/.

2. Related work

Part-aware panoptic segmentation. Part-aware panoptic segmentation [5] is an image segmentation task introduced for scene understanding at multiple abstraction levels. It extends panoptic segmentation [13] by requiring (a) object-level panoptic segmentation, and (b) part-level segmentation *within* object-level segments. Most existing works do not predict parts per object-level segment, but instead make separate predictions for panoptic segmentation and the surrogate task of part-level semantic segmentation – or *part segmentation* [10, 18, 19]. JPPF [10] proposes a single-network approach with a shared encoder followed by separate heads for semantic, instance, and part segmentation. To merge the predictions by these heads to the PPS format, JPPF proposes a new rule-based fusion strategy that outperforms the originally introduced merging strategy for PPS [5]. Panoptic-PartFormer and its extension Panoptic-PartFormer++ [18, 19] tackle PPS with a Transformer-based approach that independently makes panoptic segmentation and part segmentation predictions using separate sets of learnable queries for thing, stuff and part segments. Alternatively, ViRReq [39] is a paradigm in which PPS is predicted in a cascading fashion, by first segmenting objects and then segmenting the parts within these objects 'by request', but it requires multiple networks to achieve this.

In contrast to these existing approaches, we propose a method that jointly predicts object-level segments and the part-level segments within these objects. This aligns the learning objective with the task objective, and yields improved PPS performance. For more details, see Sec. 3.

Part-level image segmentation. Part-level segmentation is also widely studied beyond PPS. Most works address part segmentation, *i.e.*, semantic segmentation for parts [7, 8, 14–17, 20, 23, 27, 32, 34, 35, 37, 42, 43, 51, 52]. For more complete scene understanding, other works also take into account object instances, like PPS does, with instance-aware part segmentation [16, 30, 32, 46–49]. However, these works do not consider background stuff classes, which PPS does consider. Alternatively, UPerNet [44] separately predicts part-level and object-level semantic segmentation, including background classes, but without any instance awareness. Another work has extended PPS by also predicting the relations between objects [31], training a model on annotations from different datasets. In this work, we address the PPS task as introduced by De Geus *et al.* [5], and we compare our approach to state-of-the-art alternatives.

3. Method

3.1. PPS task definition

Part-aware panoptic segmentation (PPS) [5] requires consistent image segmentation across two abstraction levels: the object level and part level. For object-level segmentation, following panoptic segmentation [13], an image should be divided into N object-level segments, where N varies per image. Each segment s_i consists of a binary mask $\mathbf{M}_i^{\text{obj}}$ and an object class label c_i^{obj}. Thing classes require a segment per object instance, and stuff classes require a single segment per class. Next, per segment s_i, PPS requires a set of K_i part-level segments. Each part-level segment $s_{i,j}^{\text{pt}}$ consists of a binary part-level mask $\mathbf{M}_{i,j}^{\text{pt}}$ and a part-level class $c_{i,j}^{\text{pt}}$ that are compatible with the object-level mask and class. Specifically, the part-level mask must be a subset of the object-level mask, and the part-level class should be one of the part-level classes defined for the object-level class. For instance, if c_i^{obj} is *car*, then $c_{i,j}^{\text{pt}}$ cannot be *human-head* but it can be *car-window*. For all parts, a single part-level segment is required for each part class within an object.

To summarize, each image should be divided into a set of segments $\mathcal{S} = \{s_i\}_{i=1}^N = \{(\mathbf{M}_i^{\text{obj}}, c_i^{\text{obj}}, \mathcal{P}_i)\}_{i=1}^N$, where $\mathcal{P}_i = \{s_{i,j}^{\text{pt}}\}_{j=1}^{K_i} = \{(\mathbf{M}_{i,j}^{\text{pt}}, c_{i,j}^{\text{pt}})\}_{j=1}^{K_i}$ is a set of part-level segments compatible with the object-level segment in s_i. In practice, part-level classes are only defined for some object-level classes. When segment s_i has an object class that does not have parts, then $K_i = 0$ and $s_i = (\mathbf{M}_i^{\text{obj}}, c_i^{\text{obj}})$.

3.2. Preliminaries

Like existing state-of-the-art PPS approaches [18, 19], our work uses the powerful *mask classification* meta-architecture that is the foundation of many state-of-the-art segmentation models [2, 3, 11, 41, 50]. Here, we first explain this mask classification paradigm in more detail.

Then, we explain how existing works use it for PPS, and what the limitations are of these approaches.

3.2.1 Mask classification framework

The concept of mask classification is to predict a set of N^{q} object-level segments, *i.e.*, a set of N^{q} pixel-level masks $\hat{\mathbf{M}}^{\text{obj}}$ and corresponding class labels $\hat{c}^{\text{obj}}$. To make these predictions, mask classification networks output two components: high-resolution features $\mathbf{F} \in \mathbb{R}^{E \times H \times W}$ and queries $\mathbf{Q} \in \mathbb{R}^{N^{\text{q}} \times E}$, where H and W are the height and width of the features, and E is the feature and query dimensionality. Each of the N^{q} queries is used to predict the class label and segmentation mask of at most one object-level segment. The class predictions are a function of only the queries, and the mask predictions are a function of both the queries $\mathbf{Q}$ and the high-resolution features $\mathbf{F}$. To output these high-resolution features, the input image is first fed into a *backbone* to extract multi-scale features, *e.g.*, ResNet [9] or Swin [24]. Subsequently, these features are further refined and upsampled to high-resolution features by a *pixel decoder*, *e.g.*, Semantic FPN [12, 22]. In the other part of the network, the queries $\mathbf{Q}$ are generated by processing learnable queries $\mathbf{Q}^0$ using a *transformer decoder*, which applies self-attention across queries and cross-attention with image features. Through bipartite matching between N^{q} predicted and N ground-truth segments, each query is assigned to at most one ground-truth segment. As N^{q} is not always equal to N, some queries may not be assigned to a ground-truth segment. If so, they do not receive any supervision for segmentation, and will learn a 'no object' class. In this work, we build upon the Mask2Former [3] instantiation of this meta-architecture.

3.2.2 Mask classification for PPS

With the aforementioned meta-architecture, we can conduct object-level segmentation, and thereby solve one aspect of the PPS task definition. To solve the full task, the object-level segments should also be further segmented into parts.

Current state-of-the-art approaches [18, 19] achieve this by introducing an additional set of queries, the part-level queries depicted in Fig. 1a. Each of these part-level queries learns to represent a part-level segment. However, in contrast to the PPS task definition, these part-level segments are not explicitly linked to an object-level segment. Instead, these part-level segments represent an entire part-level class, *i.e.*, their masks contain all pixels belonging to one part-level class across multiple object segments.

This means that the learning objective of these networks is not aligned with the PPS task objective. As a result, they are not directly optimized for PPS, but instead for two surrogate tasks: object-level panoptic segmentation and part-level semantic segmentation. In addition to necessitating
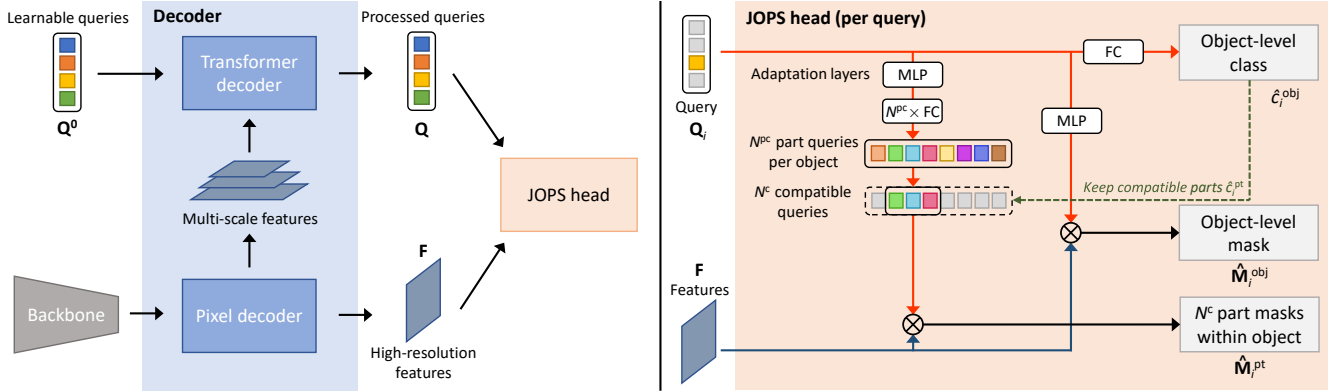
Figure 2. **Network architecture**. Left: The overall TAPPS architecture. A set of learnable queries and features from a backbone are fed into a pixel decoder and transformer decoder to generate high-resolution features and processed queries. Right: These queries and features are fed into the JOPS head, which predicts for each shared query (a) an object-level class, (b) an object-level segmentation mask, and (c) a set of part-level masks for the part-level classes compatible with the object-level class. Operator $\otimes$ denotes a matrix multiplication.

post-processing, we hypothesize that this has several negative consequences: (1) By learning parts in an 'object-instance-unaware' manner, the network learns conflicting representations in the features, **F**. On the one hand, different thing instances belong to different object-level segments, requiring distinct features. On the other hand, the parts of these instances belong to the same part-level segment, requiring similar features. We hypothesize that this conflict harms the ability of the network to learn to separate object-level thing instances. (2) Separately predicting objects and parts in this way may also cause incompatible predictions. These incompatibilities make it unclear for downstream processes which prediction to trust. To obtain predictions compliant with the PPS definition, rule-based post-processing is required, which discards incompatible predictions [5]. This is undesired because the discarded information is potentially correct. (3) By representing object- and part-level information with separate queries, the network cannot encode complementary information about an individual object and its parts, which it could use to make more informed predictions at these abstraction levels.

3.3. Task-aligned PPS

To overcome the limitations of state-of-the-art methods and achieve a better PPS performance, the objective of this work is to design a **simple** PPS approach that uses the powerful mask classification framework, but that aligns its learning objective with the PPS task objective, and explicitly predicts part-level segments within individual objects. To achieve this, we should (a) generate a unique query for each object-part combination, *i.e.*, for each part-level segment within an object-level segment, and (b) explicitly link these part-level segments to their parent objects, as PPS requires. To accomplish this, we propose Task-Aligned Part-aware Panoptic Segmentation (TAPPS), a method that jointly represents

object-level segments and their parts with a set of shared queries, and generates 'per-object' part queries from each shared query. Each of these queries can then be used to predict one part-level segment within an object-level segment, and is automatically linked to its parent object segment.

3.3.1 Overall architecture

In Fig. 2, we visualize the network architecture of TAPPS. First, we initialize a set of N^q learnable queries, $\mathbf{Q^0} \in \mathbb{R}^{N^q \times E}$. Each of these queries learns to represent one object-level segment, but also the part-level segments belonging to this object (*e.g.*, *car3* and also *car3-wheels*, *car3-windows*, etc.). Following the mask classification framework described in Sec. 3.2, these learnable queries $\mathbf{Q^0}$ and the features from a backbone are fed into a decoder, which generates a set of processed queries $\mathbf{Q}$ and high-resolution features $\mathbf{F}$. Subsequently, these processed queries and features enter the Joint Object and Part Segmentation (JOPS) head. For each query $\mathbf{Q}_i \in \mathbb{R}^E$, this head predicts (a) an object-level class, (b) an object-level segmentation mask, and (c) a set of part-level segmentation masks and classes for the parts within this object-level segment.

With this approach, we solve the limitations of existing works. Following the PPS task definition, we predict part-level segments per individual object-level segment. (1) As a result, we learn object-instance-aware representations for both parts and objects, improving the ability of the network to separate instances. (2) By predicting objects and parts jointly, the compatibility between both sets of predictions greatly improves. Moreover, in the JOPS head, we ensure that we only predict compatible parts, ensuring full object-part compatibility (see Sec. 3.3.2). (3) The joint object-part representations allow the network to fully leverage the complementary information from both abstraction levels.

3.3.2 JOPS head

Within the JOPS head, we predict an object-level segment and compatible part-level segments for each shared query. For the object-level predictions, we follow Mask2Former [3]: (1) For the object-level class, we feed each processed query $\mathbf{Q}_i$ through a single fully connected layer to predict a score for each possible class, and obtain the predicted class $\hat{c}_i^{\text{obj}}$ by picking the highest-scoring class. (2) For the object-level mask, we feed the query $\mathbf{Q}_i$ through a 3-layer MLP and generate a mask by taking the product of the resulting mask queries $\mathbf{Q}_i^{\text{m}}$ and the features $\mathbf{F}$, and applying a sigmoid activation, yielding $\hat{\mathbf{M}}_i^{\text{obj}} \in [0, 1]^{H \times W}$.

To predict part-level segments using the same features $\mathbf{F}$, we need to generate queries for each part class within an object-level segment. To do so, we first apply an MLP to each query $\mathbf{Q}_i$ to *adapt* it for part-level segmentation, and then apply N^{pc} different fully-connected layers, where N^{pc} is the total number of part-level classes in the dataset. This results in $\mathbf{Q}_i^{\text{pt}} \in \mathbb{R}^{N^{\text{pc}} \times E}$, a set of 'per-object' part-level queries, where each query always corresponds to a fixed, predetermined part class. We could then take the product of $\mathbf{Q}_i^{\text{pt}}$ and the features $\mathbf{F}$ to generate a segmentation mask for all part-level classes. However, we already have an object-level class prediction for each query, and we know that only a subset of all N^{pc} part classes is compatible with a certain object class. Therefore, we propose to only predict and supervise part-level masks for those compatible part-level classes. This simplifies the part segmentation task that the network needs to learn, as it no longer has to learn to predict 'empty' segmentation masks for incompatible part classes. Concretely, as visualized in Fig. 2, we identify the object-level class $\hat{c}_i^{\text{obj}}$ for each query $\mathbf{Q}_i$, and keep only the part-level queries $\mathbf{Q}_i^{\text{pt,c}} \in \mathbb{R}^{N^{\text{c}} \times E}$ that correspond to the N^{c} part classes that are compatible with the object-level class. Then, we compute the product of these remaining queries $\mathbf{Q}_i^{\text{pt,c}}$ and the features $\mathbf{F}$, and apply a sigmoid activation to generate compatible part segmentation masks $\hat{\mathbf{M}}_i^{\text{pt}} \in [0, 1]^{N^{\text{c}} \times H \times W}$. As each per-object part query corresponds to a fixed part class, we also know the part classes $\hat{c}_i^{\text{pt}}$. Note that the number N^{c} depends on the object-level class and will therefore vary per query $\mathbf{Q}_i$.

3.3.3 Training

To assign each query to at most one ground-truth object-level segment with corresponding part-level segments during training, we apply bipartite matching based on the predicted and ground-truth object-level classes and masks.

To supervise the object-level segments, we use the cross-entropy loss for the classes, and both the Dice [28] and cross-entropy loss for the segmentation masks. Together, these losses form the object-level loss L_{obj}. For the part-level segmentation masks, we also use the Dice and cross-

entropy loss, together forming part-level loss L_{pt}. We balance the losses with weights λ_{obj} and λ_{pt} to calculate the total loss

$$L = \lambda_{\text{obj}} L_{\text{obj}} + \lambda_{\text{pt}} L_{\text{pt}}. \tag{1}$$

We find that using $\lambda_{\text{obj}} = \lambda_{\text{pt}} = 1$ provides the best balance; see the supplementary material for more details.

4. Experimental setup

Datasets. We use the two PPS benchmarks for evaluation.

Cityscapes Panoptic Parts (Cityscapes-PP) [5] extends the original Cityscapes dataset [4] with part-level labels. It consists of street scene images from several cities. It has labels for 19 object classes (11 stuff; 8 thing). Part classes are defined and labeled for 5 object-level thing classes; there are 23 part classes in total. We train on the *train* split (2975 images), and evaluate on the *val* split (500 images).

Pascal Panoptic Parts (Pascal-PP) [5], which combines existing labels for Pascal VOC [1, 6, 29], consists of a wide range of scenes and classes. There are 59 object-level classes (39 stuff; 20 thing), and part-level classes are defined for 15 thing classes. Unless otherwise indicated, we use the default part class definition with 57 part classes in total [5]. To evaluate a more challenging setting, we additionally evaluate on Pascal-PP-107, which uses the 107 non-background classes from the Pascal-Part-108 definition introduced by Michieli *et al*. [27] for part segmentation. For both class definitions, we train on the *training* split (4998 images), and evaluate on the *validation* split (5105 images).

Baseline. Our baseline is a version of TAPPS that uses the same network architecture, but uses a separate set of 100 additional queries for part-level segmentation (see also Fig. 1a). Comparing the results of the baseline in Tab. 1 to existing methods in Tab. 2, we find that our baseline outperforms state-of-the-art approaches that also use separate part-level queries [18, 19], indicating that it is a strong baseline. See the supplementary material for more details.

Evaluation metrics. The part-aware panoptic segmentation performance is evaluated using the default Part-aware Panoptic Quality (PartPQ) metric [5]. It captures both the ability to recognize and segment object-level segments (*i.e.*, stuff regions and thing instances), and the ability to further segment the identified object-level segments into part-level masks. The PartPQ per object-level class c is given by

$$\text{PartPQ}_c = \frac{\sum_{(p,g) \in TP_c} \text{IoU}_{\text{p}}(p, g)}{|TP_c| + \frac{1}{2}|FP_c| + \frac{1}{2}|FN_c|}, \tag{2}$$

where TP_c, FP_c, and FN_c are the sets of true positive, false positive and false negative segments, respectively. A pair of predicted and ground-truth segments (p, g) of the same

Method	PartPQ			PartSQ	PQ		
	Pt	No pt	All	Pt	Th	St	All
ImageNet pre-training							
Baseline	55.2	38.8	42.9	71.5	62.0	37.3	45.7
TAPPS (ours)	**59.6**	**39.4**	**44.6**	**74.3**	**65.6**	**37.6**	**47.1**
COCO pre-training							
Baseline	64.8	49.9	53.7	73.1	74.8	48.1	57.1
TAPPS (ours)	**67.2**	**50.4**	**54.7**	**75.1**	**75.7**	**48.5**	**57.7**

(a) **Pascal-PP** *validation* [1, 5, 6, 29].

Method	PartPQ			PartSQ	PQ		
	Pt	No pt	All	Pt	Th	St	All
ImageNet pre-training							
Baseline	46.6	62.6	58.4	66.1	53.8	67.1	61.5
TAPPS (ours)	**48.7**	**63.1**	**59.3**	**66.8**	**55.6**	**67.3**	**62.4**
COCO pre-training							
Baseline	48.2	64.8	60.4	66.7	56.1	69.0	63.6
TAPPS (ours)	**48.9**	**65.7**	**61.3**	**66.9**	**57.2**	**69.7**	**64.4**

(b) **Cityscapes-PP** *val* [4, 5].

Table 1. **Main results.** We compare TAPPS to a strong baseline that uses separate sets of queries to predict object- and part-level segments, instead of predicting object- and part-level segments jointly like TAPPS (see Sec. 4).

class c is part of TP_c if the Intersection-over-Union (IoU) of their object-level masks is larger than 0.5. If a ground-truth segment is not identified, it is part of FN_c; if a prediction is incorrect, it is part of FP_c. The IoU_p term captures the segmentation performance within identified object-level segments. For object-level classes for which part classes are defined, it calculates the part-level mIoU. Otherwise, it uses the object-level IoU. We report the mean PartPQ over all classes, but also separately for object-level classes with parts (PartPQ[Pt]) and without parts (PartPQ[NoPt]).

To individually assess the ability of methods to conduct part-level segmentation within identified objects, we report the Part Segmentation Quality for object-level classes that have parts, PartSQ[Pt]. Following De Geus *et al.* [5], per object-level class c, the PartSQ[Pt] calculates the average part segmentation mIoU within object-level segments:

$$\text{PartSQ}_c^{\text{Pt}} = \frac{\sum_{(p,g) \in TP_c} \text{IoU}_\text{p}(p, g)}{|TP_c|}. \tag{3}$$

To evaluate the ability of networks to conduct object-level panoptic segmentation, we report the Panoptic Quality (PQ) metric [13]. Similarly to the PartPQ, we report the average PQ over all classes, but also over all thing classes (PQ[Th]) and stuff classes (PQ[St]) separately.

Implementation details. TAPPS is built on top of the publicly available code of state-of-the-art panoptic segmentation network Mask2Former [3]. For all datasets, we use a batch size of 16 images, and train on 4 Nvidia A100 GPUs. To optimize TAPPS, we use AdamW [26], a weight decay of 0.05, and a polynomial learning rate decay schedule with an initial learning rate of 10^{-4} and a power of 0.9. For Cityscapes-PP, we train for 90k iterations and apply conventional data augmentation steps [3, 18]: random flip, random resize with a factor between 0.5 and 2.0, and finally a random crop of 512×1024 pixels. For Pascal-PP, we train for 60k iterations in case of ImageNet pre-training, but for only 10k iterations in case of COCO pre-training, to prevent overfitting. Following state-of-the-art panoptic segmenta-

tion implementations on COCO [3, 21], we apply large-scale jittering with a scale between 0.1 and 2.0 followed by a random crop of 1024×1024 pixels. During inference, we resize the image such that the shortest side is 800 pixels. Note that we use exactly the same training and testing settings for both TAPPS and the baseline. For more implementation details, see the supplementary material.

5. Results

5.1. Main results

First, we compare TAPPS to the strong baseline that uses separate sets of queries for object-level panoptic segmentation and part-level semantic segmentation (see Sec. 4). The results in Tab. 1a demonstrate that TAPPS significantly outperforms the baseline on Pascal-PP in several aspects. Most importantly, the PartPQ for object-level classes with parts (PartPQ[Pt]) is +4.4 or +2.4 higher, depending on the pre-training strategy. Looking in more detail, we find that this increase is caused by two individual improvements: (1) The part-level segmentation quality (PartSQ[Pt]) is considerably higher than for the baseline. This shows the positive impact of having a joint representation for objects and parts, and simplifying the part segmentation task by only allowing compatible predictions (see also Sec. 5.3). (2) The panoptic quality for thing classes (PQ[Th]) also sees a substantial increase. Here, it should be noted that (a) all the object-level classes with parts are thing classes, and (b) thing classes require instance separation. Thus, this improvement does not only show the benefit of learning objects and parts jointly, but also indicates that learning parts in an object-instance-aware manner indeed improves the ability of the network to separate object instances (see also Sec. 5.4).

For Cityscapes-PP, we see similar results in Tab. 1b. Again, we observe improvements on the PartPQ[Pt], PartSQ[Pt] and PQ[Th] metrics. In this case, the absolute improvements are slightly smaller. This is expected because this dataset contains significantly fewer classes and very similar images, making it easier to learn the PPS task and limiting the potential gains that can still be obtained by TAPPS.

Method	Backbone	Pre-training	PartPQ			PartSQ	PQ
			Pt	No Pt	All	Pt	All
Pascal-PP validation							
JPPF [10]	EfficientNet-B5	I	48.3	26.9	32.2	-	-
TAPPS (ours)	RN-50	I	**59.6**	**39.4**	**44.6**	**74.3**	**47.1**
Panoptic-PartFormer† [18]	RN-50	I,C	56.1	38.8	43.2	66.8	47.6
Panoptic-PartFormer++† [19]	RN-50	I,C	52.6	42.6	45.1	60.4	51.6
TAPPS (ours)	RN-50	I,C	**67.2**	**50.4**	**54.7**	**75.1**	**57.7**
Panoptic-PartFormer† [18]	Swin-B	I,C	64.3	50.6	54.1	70.8	58.1
Panoptic-PartFormer++† [19]	Swin-B	I,C	48.9	52.1	51.3	53.0	59.8
TAPPS (ours)	Swin-B	I,C	**72.2**	**56.3**	**60.4**	**78.1**	**63.0**
Cityscapes-PP val							
Panoptic-PartFormer [18]	RN-50	I	-	-	54.5	-	57.8
Panoptic-PartFormer++ [19]	RN-50	I	-	-	57.5	-	61.6
JPPF [10]	EfficientNet-B5	I	47.7	**63.8**	**59.6**	-	-
TAPPS (ours)	RN-50	I	**48.7**	63.1	59.3	**66.8**	**62.4**
Panoptic-PartFormer [18]	RN-50	I,C	43.9	62.4	57.5	60.1	61.6
Panoptic-PartFormer++ [19]	RN-50	I,C	42.5	65.1	59.2	-	63.6
TAPPS (ours)	RN-50	I,C	**48.9**	**65.7**	**61.3**	**66.9**	**64.4**
Panoptic-PartFormer [18]	Swin-B	I,C	45.6	67.8	62.0	59.0	66.6
Panoptic-PartFormer++ [19]	Swin-B	I,C	46.0	68.2	62.3	-	68.0
Panoptic-PartFormer++ [19]	ConvNeXt-B	I,C	46.4	**69.1**	63.1	-	**68.2**
SegFormer-B5 + CondInst + BPR [38–40, 45]	MiT-B5 + RN-50	I,C*	48.6	67.5	62.5	-	-
TAPPS (ours)	Swin-B	I,C	**53.0**	69.0	**64.8**	**68.0**	68.0

Table 2. **Comparison with state of the art**. Evaluation on the Cityscapes-PP and Pascal-PP benchmarks [4–6]. RN-50 is ResNet-50 [9]. Other backbones are EfficientNet-B5 [36], MiT-B5 [45], Swin-B [24] and ConvNeXt-B [25]. I = ImageNet [33], C = COCO panoptic [21], C* = COCO pre-training for instance segmentation. †PartPQ scores for these existing methods have been re-evaluated using official code [5] and are higher than originally reported on Pascal-PP [18, 19], see supplementary material for more details.

5.2. Comparison with state of the art

In Tab. 2, we compare TAPPS to existing state-of-the-art methods across different datasets, backbones, and pre-training settings. On both Pascal-PP and Cityscapes-PP, TAPPS significantly outperforms existing work. Most importantly, it consistently scores higher on the PartPQ, PartPQ^{Pt}, and PartSQ^{Pt} metrics. TAPPS is only slightly outperformed by JPPF [10] on Cityscapes-PP with ImageNet pre-training, but we note that JPPF uses the EfficientNet-B5 [36] backbone, which is much more powerful than the ResNet-50 [9] used by TAPPS. Moreover, we do outperform JPPF by a large margin on the Pascal-PP dataset, showing the strength of TAPPS on more complex datasets. Overall, we achieve new state-of-the-art results on both datasets, obtaining PartPQ scores of 60.4 and 64.8 on Pascal-PP and Cityscapes-PP, improvements of +6.3 and +1.7, respectively. In the supplementary material, we compare qualitative examples of TAPPS, our baseline, and existing approaches, and we also show typical failure cases.

5.3. Ablations

Predicting only compatible parts. Using shared queries for object- and part-level segmentation enables TAPPS to only allow part segmentation predictions that

Predict compat. parts		PartPQ			PartSQ	PQ	Obj. w/o
Training	Testing	Pt	No Pt	All	Pt	All	conflicts
ImageNet pre-training							
Baseline		55.2	38.8	42.9	71.5	45.7	66.1%
✗	✗	57.9	39.2	43.9	72.9	46.6	99.7%
✗	✓	58.0	39.2	43.9	72.9	46.6	**100.0%**
✓	✓	**59.6**	**39.4**	**44.6**	**74.3**	**47.1**	**100.0%**
COCO pre-training							
Baseline		64.8	49.9	53.7	73.1	57.1	64.6%
✗	✗	66.3	**50.5**	54.6	74.0	**57.9**	99.5%
✗	✓	66.4	**50.5**	54.6	74.0	**57.9**	**100.0%**
✓	✓	**67.2**	50.4	**54.7**	**75.1**	57.7	**100.0%**

Table 3. **Predicting only compatible part classes during training and testing.** Evaluated on Pascal-PP [1, 5, 6, 29].

are compatible with the query's object-level class (see Sec. 3.3.2). In Tab. 3, we evaluate the effect of predicting only compatible parts during both training and testing. In addition to the main metrics, we also assess the percentage of predicted objects for which there are *no* conflicting part-level predictions. The results show that, even when allowing incompatible predictions, TAPPS has significantly fewer object-part conflicts than the baseline. Naturally, when predicting only compatible parts during testing, all object-part conflicts are removed, eliminating the need

Adaptation layers	PartPQ			PartSQ	PQ
	Pt	No Pt	All	Pt	All
-	67.3	49.6	54.1	75.2	57.1
1× FC	67.1	50.3	54.6	75.1	57.6
2-layer MLP	67.2	50.4	54.7	75.1	57.7
3-layer MLP	67.1	50.4	54.6	75.1	57.7

Table 4. **JOPS head architecture (Fig. 2).** Evaluated on Pascal-PP [1, 5, 6, 29], with pre-training on COCO panoptic [21].

Method	mIoU$^{\text{Th}}$	PQ$^{\text{Th}}$	Method	mIoU$^{\text{Th}}$	PQ$^{\text{Th}}$
Baseline	69.8	62.0	Baseline	77.6	53.8
TAPPS (ours)	69.9	65.6	TAPPS (ours)	78.2	55.6

(a) **Pascal-PP [1, 5, 6, 29].**   (b) **Cityscapes-PP [4, 5].**

Table 5. **Performance for *things*.** ImageNet pre-training [33].

Method	PartPQ			PartSQ	PQ		
	Pt	No Pt	All	Pt	Th	St	All
ImageNet pre-training							
Baseline	45.7	38.9	40.6	59.3	62.1	37.6	45.9
TAPPS (ours)	**49.8**	**39.3**	**42.0**	**61.7**	**65.8**	**37.7**	**47.2**
COCO pre-training							
Baseline	53.2	49.8	50.7	59.9	75.2	**48.1**	57.2
TAPPS (ours)	**55.4**	**50.0**	**51.4**	**62.1**	**75.6**	**48.1**	**57.4**

Table 6. **107 part-level categories.** Results on the more challenging Pascal-PP-107 dataset [1, 5, 6, 27, 29].

Method	PartPQ			PartSQ	PQ
	Pt	No Pt	All	Pt	All
Dynamic	65.1	**50.5**	54.2	73.1	**57.7**
Fixed	**67.2**	50.4	**54.7**	**75.1**	**57.7**

Table 7. **Fixed or dynamic part segmentation.** Evaluated on Pascal-PP [1, 5, 6, 29], with pre-training on COCO panoptic [21].

for post-processing. However, this does not yield a big improvement in terms of the segmentation quality. When we also apply this during training, we do observe a part segmentation quality (PartSQ$^{\text{Pt}}$) improvement. This shows that simplifying the part segmentation task during training leads to improved performance, as we hypothesized in Sec. 3.3.2.

JOPS head architecture. In Tab. 4, we evaluate the impact of using different numbers of adaptation layers in the JOPS head *before* applying the N^{pc} fully-connected (FC) layers to generate the part queries (see Fig. 2). We find that one or two adaptation layers are necessary to generate object- and part-level representations that are sufficiently distinct to perform their respective tasks accurately; adding any more layers does not yield further improvements.

5.4. Additional analyses

Instance separability. Tab. 1 showed that TAPPS improves the PQ$^{\text{Th}}$, *i.e.*, the ability to recognize, segment and classify object instances. To assess if this is due to better object recognition or improved instance separability, we group all thing instances of the same object-level class together and evaluate the instance-agnostic semantic segmentation performance with the mIoU$^{\text{Th}}$. If the PQ$^{\text{Th}}$ improvement were due to improved recognition, we expect the mIoU$^{\text{Th}}$ to improve too. However, the results in Tab. 5 show only a minor mIoU$^{\text{Th}}$ improvement. This indicates that the PQ$^{\text{Th}}$ gain is not due to better recognition, but mainly results from a better ability to separate instances, showing the benefit of learning parts in an object-instance-aware manner.

Performance on Pascal-PP-107. To assess the performance of TAPPS in a more complex setting, we evaluate it on Pascal-PP-107, which has 107 part-level classes instead of 57. The results on this dataset, reported in Tab. 6, show once more that TAPPS consistently improves the part segmentation and thing segmentation performance with respect to the baseline, in this case leading to a PartPQ$^{\text{Pt}}$ improvement of +4.1 or +2.2, depending on the pre-training strategy. This demonstrates that TAPPS is also effective when the PPS task becomes more complex.

Fixed vs. dynamic part segmentation. As explained in Sec. 3.3.2, TAPPS uses a fixed fully connected layer for each part class to generate the corresponding 'per-object' part query. Alternatively, it is possible to predict part masks and classes dynamically within each object segment, using a set of dynamic queries like we do for object-level segmentation. In Tab. 7, we compare our *fixed* part segmentation setting with this *dynamic* approach, which is explained in more detail in the supplementary material. We find that our fixed approach results in a better PartSQ$^{\text{Pt}}$ and therefore PartPQ$^{\text{Pt}}$ performance. We hypothesize that the dynamic approach performs worse because it makes the part segmentation task unnecessarily complex, as the network needs to learn to assign segments to queries dynamically, and additionally predict a class label.

6. Conclusion

With experiments, we have shown that TAPPS considerably outperforms methods that predict objects and parts separately, by improving the object instance separability, part segmentation quality, and object-part compatibility. Importantly, these improvements can be attributed to the fact that TAPPS is directly optimized for the PPS task, using a set of shared queries to jointly predict objects and corresponding parts. With our promising findings, we hope to inspire future research towards even more complete scene understanding, *e.g.*, image segmentation at even more abstraction levels, potentially with more flexible class hierarchies.

References

[1] Xianjie Chen, Roozbeh Mottaghi, Xiaobai Liu, Sanja Fidler, Raquel Urtasun, and Alan Yuille. Detect What You Can: Detecting and Representing Objects using Holistic Models and Body Parts. In *CVPR*, 2014. 5, 6, 7, 8

[2] Bowen Cheng, Alexander G. Schwing, and Alexander Kirillov. Per-Pixel Classification is Not All You Need for Semantic Segmentation. In *NeurIPS*, 2021. 3

[3] Bowen Cheng, Ishan Misra, Alexander G Schwing, Alexander Kirillov, and Rohit Girdhar. Masked-attention Mask Transformer for Universal Image Segmentation. In *CVPR*, 2022. 3, 5, 6

[4] 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 *CVPR*, 2016. 5, 6, 7, 8

[5] Daan de Geus, Panagiotis Meletis, Chenyang Lu, Xiaoxiao Wen, and Gijs Dubbelman. Part-aware Panoptic Segmentation. In *CVPR*, 2021. 1, 2, 3, 4, 5, 6, 7, 8

[6] Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John Winn, and Andrew Zisserman. The Pascal Visual Object Classes (VOC) Challenge. *IJCV*, 88(2):303–338, 2010. 5, 6, 7, 8

[7] Ke Gong, Xiaodan Liang, Yicheng Li, Yimin Chen, Ming Yang, and Liang Lin. Instance-level human parsing via part grouping network. In *ECCV*, 2018. 3

[8] Ke Gong, Yiming Gao, Xiaodan Liang, Xiaohui Shen, Meng Wang, and Liang Lin. Graphonomy: Universal human parsing via graph transfer learning. In *CVPR*, 2019. 3

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *CVPR*, 2016. 3, 7

[10] Sravan Kumar Jagadeesh, René Schuster, and Didier Stricker. Multi-task Fusion for Efficient Panoptic-Part Segmentation. In *ICPRAM*, 2022. 2, 7

[11] Jitesh Jain, Jiachen Li, MangTik Chiu, Ali Hassani, Nikita Orlov, and Humphrey Shi. OneFormer: One Transformer to Rule Universal Image Segmentation. In *CVPR*, 2023. 3

[12] Alexander Kirillov, Ross Girshick, Kaiming He, and Piotr Dollar. Panoptic Feature Pyramid Networks. In *CVPR*, 2019. 3

[13] Alexander Kirillov, Kaiming He, Ross Girshick, Carsten Rother, and Piotr Dollar. Panoptic Segmentation. In *CVPR*, 2019. 1, 2, 3, 6

[14] Jianshu Li, Jian Zhao, Yunpeng Chen, Sujoy Roy, Shuicheng Yan, Jiashi Feng, and Terence Sim. Multi-human parsing machines. In *ACM MM*, 2018. 3

[15] Liulei Li, Tianfei Zhou, Wenguan Wang, Jianwu Li, and Yi Yang. Deep Hierarchical Semantic Segmentation. In *CVPR*, 2022.

[16] Peike Li, Yunqiu Xu, Yunchao Wei, and Yi Yang. Self-Correction for Human Parsing. *IEEE TPAMI*, 44(6):3260–3271, 2022. 3

[17] Qizhu Li, Anurag Arnab, and Philip HS Torr. Holistic, instance-level human parsing. In *BMVC*, 2017. 3

[18] Xiangtai Li, Shilin Xu, Yibo Yang, Guangliang Cheng, Yunhai Tong, and Dacheng Tao. Panoptic-PartFormer: Learning a Unified Model for Panoptic Part Segmentation. In *ECCV*, 2022. 1, 2, 3, 5, 6, 7

[19] Xiangtai Li, Shilin Xu, Yibo Yang, Haobo Yuan, Guangliang Cheng, Yunhai Tong, Zhouchen Lin, and Dacheng Tao. PanopticPartFormer++: A Unified and Decoupled View for Panoptic Part Segmentation. *arXiv preprint arXiv:2301.00954*, 2023. 1, 2, 3, 5, 7

[20] Xiaodan Liang, Ke Gong, Xiaohui Shen, and Liang Lin. Look into Person: Joint Body Parsing & Pose Estimation Network and A New Benchmark. *IEEE TPAMI*, 41(4):871–885, 2018. 3

[21] 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 *ECCV*, 2014. 6, 7, 8

[22] Tsung-Yi Lin, Piotr Dollar, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature Pyramid Networks for Object Detection. In *CVPR*, 2017. 3

[23] Qing Liu, Adam Kortylewski, Zhishuai Zhang, Zizhang Li, Mengqi Guo, Qihao Liu, Xiaoding Yuan, Jiteng Mu, Weichao Qiu, and Alan Yuille. Learning Part Segmentation Through Unsupervised Domain Adaptation From Synthetic Vehicles. In *CVPR*, 2022. 3

[24] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin Transformer: Hierarchical Vision Transformer Using Shifted Windows. In *ICCV*, 2021. 3, 7

[25] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A ConvNet for the 2020s. In *CVPR*, 2022. 7

[26] Ilya Loshchilov and Frank Hutter. Decoupled Weight Decay Regularization. In *ICLR*, 2019. 6

[27] Umberto Michieli, Edoardo Borsato, Luca Rossi, and Pietro Zanuttigh. GMNet: Graph Matching Network for Large Scale Part Semantic Segmentation in the Wild. In *ECCV*, 2020. 3, 5, 8

[28] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. V-Net: Fully Convolutional Neural Networks for Volumetric Medical Image Segmentatio. In *3DV*, 2016. 5

[29] Roozbeh Mottaghi, Xianjie Chen, Xiaobai Liu, Nam-Gyu Cho, Seong-Whan Lee, Sanja Fidler, Raquel Urtasun, and Alan Yuille. The Role of Context for Object Detection and Semantic Segmentation in the Wild. In *CVPR*, 2014. 5, 6, 7, 8

[30] Tai-Yu Pan, Qing Liu, Wei-Lun Chao, and Brian Price. Towards Open-World Segmentation of Parts. In *CVPR*, 2023. 3

[31] Lu Qi, Jason Kuen, Weidong Guo, Jiuxiang Gu, Zhe Lin, Bo Du, Yu Xu, and Ming-Hsuan Yang. AIMS: All-Inclusive Multi-Level Segmentation for Anything. In *NeurIPS*, 2023. 3

[32] Tao Ruan, Ting Liu, Zilong Huang, Yunchao Wei, Shikui Wei, and Yao Zhao. Devil in the details: Towards accurate single and multiple human parsing. In *AAAI*, 2019. 3

[33] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *IJCV*, 115(3):211–252, 2015. 7, 8

[34] Rishubh Singh, Pranav Gupta, Pradeep Shenoy, and Ravikiran Sarvadevabhatla. FLOAT: Factorized Learning of Object Attributes for Improved Multi-Object Multi-Part Scene Parsing. In *CVPR*, 2022. 3

[35] Peize Sun, Shoufa Chen, Chenchen Zhu, Fanyi Xiao, Ping Luo, Saining Xie, and Zhicheng Yan. Going denser with open-vocabulary part segmentation. In *ICCV*, 2023. 3

[36] Mingxing Tan and Quoc Le. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. In *ICML*, 2019. 7

[37] Xin Tan, Jiachen Xu, Zhou Ye, Jinkun Hao, and Lizhuang Ma. Confident Semantic Ranking Loss for Part Parsing. In *ICME*, 2021. 3

[38] Chufeng Tang, Hang Chen, Xiao Li, Jianmin Li, Zhaoxiang Zhang, and Xiaolin Hu. Look Closer To Segment Better: Boundary Patch Refinement for Instance Segmentation. In *CVPR*, 2021. 7

[39] Chufeng Tang, Lingxi Xie, Xiaopeng Zhang, Xiaolin Hu, and Qi Tian. Visual Recognition by Request. In *CVPR*, 2023. 2

[40] Zhi Tian, Chunhua Shen, and Hao Chen. Conditional Convolutions for Instance Segmentation. In *ECCV*, 2020. 7

[41] Huiyu Wang, Yukun Zhu, Hartwig Adam, Alan Yuille, and Liang-Chieh Chen. MaX-DeepLab: End-to-End Panoptic Segmentation With Mask Transformers. In *CVPR*, 2021. 3

[42] Wenguan Wang, Hailong Zhu, Jifeng Dai, Yanwei Pang, Jianbing Shen, and Ling Shao. Hierarchical human parsing with typed part-relation reasoning. In *CVPR*, 2020. 3

[43] Meng Wei, Xiaoyu Yue, Wenwei Zhang, Shu Kong, Xihui Liu, and Jiangmiao Pang. OV-PARTS: Towards Open-Vocabulary Part Segmentation. In *NeurIPS*, 2023. 3

[44] Tete Xiao, Yingcheng Liu, Bolei Zhou, Yuning Jiang, and Jian Sun. Unified perceptual parsing for scene understanding. In *ECCV*, 2018. 3

[45] Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, Jose M. Alvarez, and Ping Luo. SegFormer: Simple and Efficient Design for Semantic Segmentation with Transformers. In *NeurIPS*, 2021. 7

[46] Lu Yang, Qing Song, Zhihui Wang, and Ming Jiang. Parsing R-CNN for Instance-Level Human Analysis. In *CVPR*, 2019. 3

[47] Lu Yang, Qing Song, Zhihui Wang, Mengjie Hu, Chun Liu, Xueshi Xin, Wenhe Jia, and Songcen Xu. Renovating Parsing R-CNN for Accurate Multiple Human Parsing. In *ECCV*, 2020.

[48] Lu Yang, Qing Song, Zhihui Wang, Mengjie Hu, and Chun Liu. Hier R-CNN: Instance-Level Human Parts Detection and A New Benchmark. *IEEE TIP*, 30:39–54, 2021.

[49] Sanyi Zhang, Xiaochun Cao, Guo-Jun Qi, Zhanjie Song, and Jie Zhou. AIParsing: Anchor-Free Instance-Level Human Parsing. *IEEE TIP*, 31:5599–5612, 2022. 3

[50] Wenwei Zhang, Jiangmiao Pang, Kai Chen, and Chen Change Loy. K-Net: Towards Unified Image Segmentation. In *NeurIPS*, 2021. 3

[51] Jian Zhao, Jianshu Li, Yu Cheng, Terence Sim, Shuicheng Yan, and Jiashi Feng. Understanding humans in crowded scenes: Deep nested adversarial learning and a new benchmark for multi-human parsing. In *ACM MM*, 2018. 3

[52] Yifan Zhao, Jia Li, Yu Zhang, and Yonghong Tian. Multi-Class Part Parsing With Joint Boundary-Semantic Awareness. In *ICCV*, 2019. 3