JPPF: Multi-task Fusion for Consistent Panoptic-Part Segmentation

Shishir Muralidhara[1], Sravan Kumar Jagadeesh[1], René Schuster[1*], Didier Stricker[1]

[1]Augmented Vision, German Research Center for Artificial Intelligence – DFKI, Trippstadter Straße 122, Kaiserslautern, 67663, Germany.

*Corresponding author(s). E-mail(s): rene.schuster@dfki.de;
Contributing authors: shishir.muralidhara@dfki.de; sravan.jagadeesh@dfki.de; didier.stricker@dfki.de;

Abstract

Part-aware panoptic segmentation is a problem of computer vision that aims to provide a semantic understanding of the scene at multiple levels of granularity. More precisely, semantic areas, object instances, and semantic parts are predicted simultaneously. In this paper, we present our Joint Panoptic Part Fusion (JPPF) that combines the three individual segmentations effectively to obtain a panoptic-part segmentation. Two aspects are of utmost importance for this: First, a unified model for the three problems is desired that allows for mutually improved and consistent representation learning. Second, balancing the combination so that it gives equal importance to all individual results during fusion. Our proposed JPPF is parameter-free and dynamically balances its input. The method is evaluated and compared on the Cityscapes Panoptic Parts (CPP) and Pascal Panoptic Parts (PPP) datasets in terms of PartPQ and Part-Whole Quality (PWQ). In extensive experiments, we verify the importance of our fair fusion, highlight its most significant impact for areas that can be further segmented into parts, and demonstrate the generalization capabilities of our design without fine-tuning on 5 additional datasets.

Keywords: Semantic, Panoptic-Part, Segmentation, Fusion

1 Introduction

Humans are able to perceive various levels of detail and abstraction of a scene. We can not only understand different semantic categories such as bus, car, and sky, but we can also distinguish between individual entities (instances) and their components (parts), such as windows or wheels. In computer vision, the estimation of these parallel layers of abstraction has recently been introduced as panoptic-part segmentation [10]. Yet, there exists no completely unified and joint approach for this problem.

According to [6], the two pieces that make up a scene are *stuff* and *things*. Things are countable objects such as persons, cars, or buses, whereas *stuff*, like the sky or road, is usually amorphous and innumerable. Those two categories are identified in the well studied tasks of semantic segmentation and instance segmentation. However, both tasks are incapable of describing the entirety of the scene. To fill this gap, panoptic segmentation [21] was presented, which recognizes and segments both, *stuff* and *things*. After this, several

approaches for panoptic segmentation have been proposed [5, 20, 27, 42, 46, 57].

Part segmentation, or part parsing, on the other hand, seeks to semantically analyze the image based on part-level. There has been some effort in this area, where part segmentation is often treated as a semantic segmentation problem [12, 18, 19, 25, 35, 38]. A few methods are instance-aware [11, 25, 63] and even fewer handle multi-class part objects [41, 64].

With the release of datasets for panoptic-part segmentation [10, 40], the first methods for this problem have been proposed [17, 28, 29]. In [10], a baseline approach is presented in which two networks for panoptic and part segmentation are used. These two networks are trained independently and the results of both are combined using a uni-directional (top-down) merging strategy. This technique of independent training has significant drawbacks. Due to the use of two different networks, there is a computational overhead. As the authors employ different networks, there will be no consistency in their predictions, making the merging process ineffective. Also, the independent training strategy leads to learning redundancy since they could potentially share semantic information between segmentation heads.

Afterwards, Panoptic-PartFormer (PPF) [28] has been proposed, in which the authors present a unified, combined transformer for *things*, *stuff*, and parts that iteratively refines the individual segmentations to achieve consistency. In this design, redundancies are avoided and similarities between tasks are exploited, but we argue that an explicit modeling of multi-task fusion can produce more accurate results.

To this end, and to overcome the limitations of the top-down merging, we have presented a Joint Panoptic-Part Fusion (JPPF) for panoptic-part segmentation in [17], in which each sub-task is treated equally to allow for mutual benefits and maximal consistency (c.f. Fig. 1). By sharing a backbone for all three tasks, the joint fusion is outperforming the top-down baseline, while being more efficient at the same time.

In this work, we re-present our JPPF [17] and extend the experiments, validation, and discussion. In short,

- we present a single neural network that uses a shared encoder to perform semantic, instance, and part segmentation and fuses them efficiently to produce panoptic-part segmentation.
- we propose a parameter-free Joint Panoptic-Part Fusion (JPPF) module that dynamically considers the logits from the semantic, instance, and part head and consistently integrates the three predictions.
- we conduct a thorough analysis of our approach and demonstrate the efficacy, accuracy, and consistency of the joint fusion strategy.
- we obtain state-of-the-art results for panoptic-part segmentation on various datasets and metrics, surpassing our previous work [17], the top-down baseline [10], and the transformer-based competitor PPF [28].
- we demonstrate that our approach generalizes to many other datasets without fine-tuning.

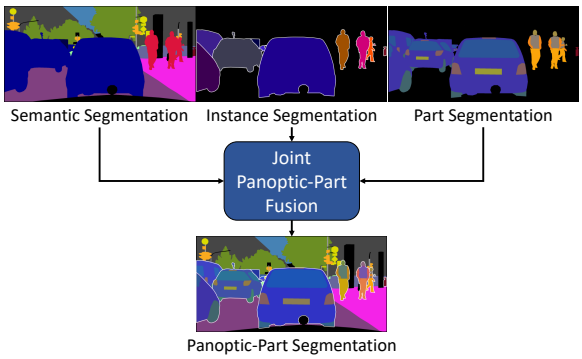


Fig. 1: Our Joint Panoptic-Part Fusion (JPPF) combines individual predictions into a consistent panoptic-part segmentation

2 Related Work

2.1 Towards Panoptic-Part Segmentation

Part-aware panoptic segmentation [10] is a recently introduced problem that brings semantic, instance, and part segmentation together. There have been several methods proposed for these individual tasks, including panoptic segmentation, which is a blend of semantic and instance segmentation.

Semantic Segmentation

PSPnet [62] introduced the pyramid pooling module, which focuses on the importance of multi-scale features by learning them at many scales, then concatenating and up-sampling them. Chen et al. [2] proposed Atrous Spatial Pyramid Pooling (ASPP), which is based on spatial pyramid pooling and combines features from several parallel atrous convolutions with varying dilation rates, as well as global average pooling. The incorporation of multi-scale characteristics and the capturing of global context increases computational complexity. So, Chen et al. [3] introduced the Dense Prediction Cell (DPC) and Valada et al. [54] suggested multi-scale residual units with changing dilation rates to compute high-resolution features at various spatial densities, as well as an efficient atrous spatial pyramid pooling module called eASPP to learn multi-scale representation with fewer parameters and a broader receptive field. In the encoder-decoder architecture, a lot of effort has been advocated for improving the decoder's upsampling layer. Chen et al. [4] extend DeepLabV3 [2] by adding an efficient decoder module to enhance segmentation results at object boundaries. Later, Tian et al. [53] suggest replacing it with data-dependent up-sampling (DUpsampling), which can recover pixel-wise prediction from low-resolution CNN outputs and take advantage of the redundant label space in semantic segmentation.

Instance Segmentation

Here, we mainly concentrate on proposal based approaches. Hariharan et al. [13] proposed a simultaneous object recognition and segmentation technique that uses Multi-scale Combinatorial Grouping (MCG) [45] to generate proposals and then run them through a CNN for feature extraction. In addition, Hariharan et al. [14] presented a hyper-column pixel descriptor that captures feature representations of all layers in a CNN with a strong correlation for simultaneous object detection and segmentation. O Pinheiro et al. [44] proposed the DeepMask network, which employs a CNN to predict the segmentation mask of each object as well as the likelihood of the object being in the patch. FCIS [30] employs position sensitive inside/outside score maps to simultaneously predict object detection and segmentation. Later, one of the most popular networks for instance segmentation, Mask-RCNN [16], was introduced. It extends Faster-RCNN [48] with an extra network that segments each of the detected objects. RoI-align, which preserves exact spatial position, replaces RoI-pool, which performs coarse spatial quantization for feature encoding.

Part Segmentation

Dense part-level segmentation, on the other hand, is instance agnostic and is regarded as a semantic segmentation problem [12, 18, 19, 25, 35, 39, 41, 64]. Most of the research has been conducted to perform human part parsing [7, 11, 22, 24, 31, 32, 49, 58, 63], and only little work has addressed multi-part segmentation tasks [41, 64].

Panoptic Segmentation

The authors of [21] combined the output of two independent networks for semantic and instance segmentation and coined the term panoptic segmentation. Panoptic segmentation approaches can be divided into top-down methods [23, 26, 34, 46, 51, 57] that prioritize semantic segmentation prediction and bottom-up methods [5, 8, 59] that prioritize instance prediction. Our previous in [17] builds on EfficientPS [42] and extends this model to obtain panoptic-part segmentation. This work, builds on our previous design of a joint architecture and exploits its modularity to replace individual components.

2.2 Panoptic-Part Segmentation

2.2.1 Datasets and Baselines

In recent years, Part-Aware Panoptic Segmentation [10] was introduced, which aims at a unified scene and part-parsing. Also, de Geus et al. [10] introduced a baseline model using a state-of-the-art panoptic segmentation network and a part segmentation network, merging them using heuristics. The panoptic and part segmentation is merged in top-down or bottom-up manner. In the top-down merge, the prediction from panoptic segmentation is re-used for scene-level semantic classes that do not consist of parts. Then for partitionable semantic classes, the corresponding segment of the part prediction is extracted. In case of conflicting predictions, a void label will be assigned. According to de Geus et al. [10],

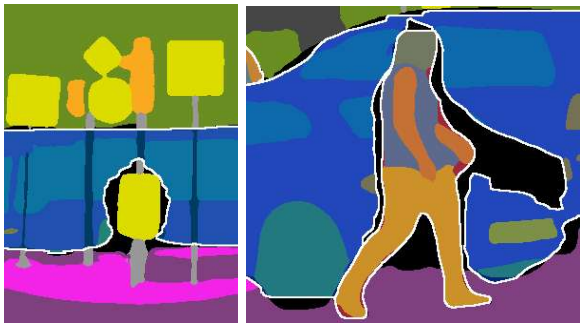


Fig. 2: A typical issue with the top-down merging approach of [10] are the gaps around the contours of objects due to inconsistencies and difficulties in distinguishing between *stuff* and *things*

top-down merge produces better results than the bottom-up approach. In addition, their paper has released two datasets with panoptic-part annotations: Cityscapes Panoptic Part (CPP) dataset and Pascal Panoptic Part (PPP) dataset [40]. Along with the drawbacks of employing independent networks as mentioned in Section 1, there are concerns with the usage of top-down merge. Due to inconsistencies, top-down merging may result in undefined regions around the contours of objects. Due to some imbalance between *stuff* and *things*, it also has trouble separating them. These issues are highlighted in Fig. 2. Furthermore, the uni-directional merge accounts higher importance to one of the predictions, neglecting the potential of mutual refinement during fusion. With our unified fusion for semantics, instances, and parts, we resolve these issues, giving equal priority to all individual predictions.

2.2.2 Unified Models

Panoptic-PartFormer (PPF) [28] was developed in parallel to [17] and follows a similar goal as our line of work: To unify panoptic-part segmentation. However, the authors of [28] approach the unification from the other side. While we suggest a unified fusion module to combine individual results in a well balanced manner, they propose a shared encoder and transformer-based decoder to predict *stuff*, *things*, and parts together via a single model. This way, they achieve remarkable consistency and results, however though the prediction of the individual tasks is fully unified in a single architecture that uses task-specific queries, it is followed by the same uni-directional top-down merging as in [10], leading to *void* labels where inconsistencies remain.

Li et al. [29] propose a second version of their Panoptic-PartFormer (PPF++), in which they also introduce a new metric, called Part-Whole Quality (PWQ). Compared to the PartPQ of [10], PWQ is supposed to resolve the bias towards the PQ metric of panoptic segmentation. In our experiments, we will consider both these metrics for thorough comparisons.

3 Unified Panoptic-Part Segmentation

The main contribution of our work is the Joint Panoptic-Part Fusion (JPPF) that produces highly dense and consistent panoptic-part segmentations in an efficient manner. To obtain individual predictions for our fusion, in theory any method could be applied. However, we argue that a combined network for all three segmentation tasks produces better results through mutual learning and reasoning. Therefore in this section, we first formalize the problem of panoptic-part segmentation, then explain our unified network architecture presented in [17], and lastly describe the inner workings of JPPF.

3.1 Panoptic-Part Segmentation

The goal of panoptic-part segmentation is to predict a panoptic-part label (s, id, p) for each pixel of an image I. Here, s represents semantic scene level class, p represents the part-level class and id indicates the instance identifier for each object. It is important to note, that not all pixels in an image may represent all components of panoptic-part segmentation, e.g. *stuff* is not instantiable, and there are many semantic classes for which it does not make sense to further subdivide them into parts, e.g. the sky. Anyhow, the three labels can be obtained independently, however a valid panoptic-part label must be consistent, i.e. free from contradiction. E.g. a car can not share the object identifier of a bicycle or consist of human body parts. Achieving this consistency is the fundamental challenge in panoptic-part segmentation. To obtain this goal, different strategies can be followed, including naive merging [10], joint prediction [28, 29], or – as in our case – fusion [17].

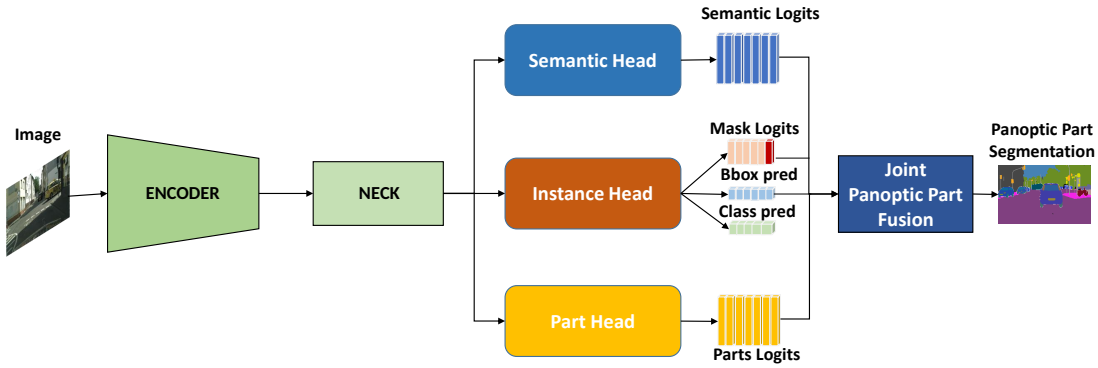


Fig. 3: Our overall architecture for panoptic-part segmentation features a shared encoder, three specialized prediction heads, and the unified joint fusion module. Its modular structure allows to easily replace the feature backbone or use intermediate results from other approaches to perform a consistent fusion

3.2 Overall Architecture

To obtain individual predictions for semantics, instances, and parts, our previous work extends EfficientPS [42] by incorporating a part segmentation head. We reuse the backbone, semantic head, and instance head of EfficientPS. As part segmentation can be regarded as a semantic segmentation problem, we are replicating the architecture of the semantic branch of EfficientPS and train it for part-level segmentation. All three resulting heads share a common backbone – in our case EfficientNet [52] – which helps to ensure that the predictions made by the heads are consistent with one another. Sharing a single representation for all three tasks improves efficiency and is beneficial during learning, as shown by our experiments in Section 4.2. An overview of the architecture of our proposed model is shown in Fig. 3.

3.2.1 Part Segmentation Head

According to previous work [10], the grouping of parts yields better results. We have verified this finding for our architecture in [17] and consequently follow the same principle and group semantically identical parts, e.g. the windows of cars and buses are grouped into a single window class. The grouping of elements allows the network to learn without ambiguity and provides more data per class for training. Additionally, we represent all non-partitionable semantic classes as a single background class within our part head. This avoids redundant predictions across different heads and further balances the learning of parts versus other classes. Both groupings of classes (semantic grouping of parts, as well as grouping of the background) can later be reverted into class-specific parts by the additional information of the other prediction heads to obtain a fine-grained panoptic-part segmentation.

3.3 Joint Panoptic-Part Fusion

The mutual combination of the predictions for semantic segmentation, instance segmentation, and part segmentation are the core of our work. Inspired by the panoptic fusion module of EfficientPS [42], we propose a module that jointly fuses the individual results of the three heads by giving each prediction equal priority and thoroughly exploiting coherent predictions. Given the definition of panoptic-part segmentation, we identified four possible cases for fusion: Partitionable and non-partitionable *stuff*, and partitionable and non-partitionable *things*. In the following, we will first describe the required input for our fusion module and then describe the three combinations which actually occur in the existing datasets (partitionable *stuff* is not included). However, our approach generalized to the missing case as well. Fig. 5 depicts our JPPF module.

Input and Pre-Processing

The input for our fusion are the individual dense predictions for semantics, instances, and parts. In our complete architecture, these are obtained from the three prediction heads using the shared backbone, but it could be any other source that satisfies the preconditions. More precisely, we require three input components:

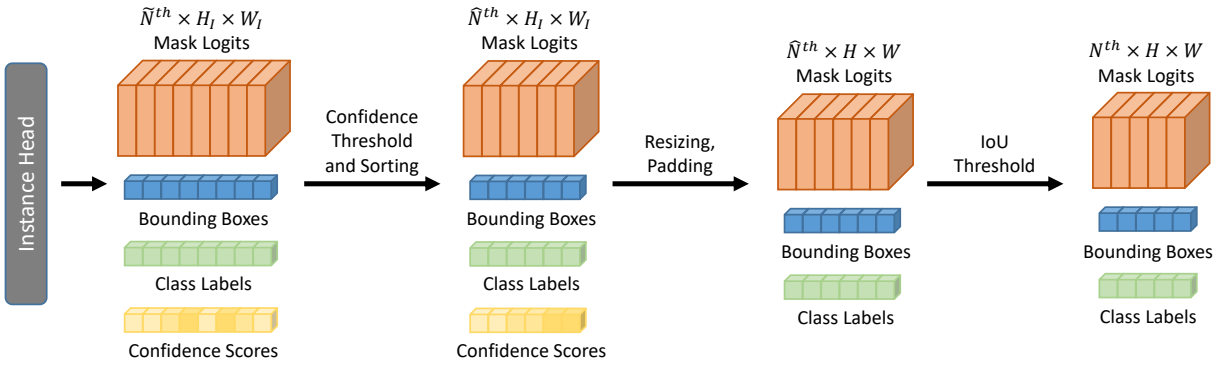


Fig. 4: Illustration of the pre-processing steps in [42] for predictions from the instance head. The remaining instances serve as input for our fusion

1. A map of semantic logits $S \in \mathrm{R}^3$ of shape $C_{st,th} \times H \times W$ in the interval $[0, 1]$ (e.g. via softmax activation), in which H and W are spatial dimensions of the input image (potentially resized) and $C_{st,th} = C_{st} + C_{th}$ is the total number of semantic classes.
2. A set of instance predictions for the *things* classes, each consisting of:
   (a) A softmax-activated map of logits M of shape $H_I \times W_I$ representing the object mask.
   (b) An axis-aligned 2D bounding box.
   (c) A class label c for this object.
   (d) A confidence score in the interval $[0, 1]$.
3. A map of part logits $P \in \mathrm{R}^3$ of shape $(C_p + 1) \times H \times W$ in the interval $[0, 1]$, in which C_p is the number of (grouped) part classes.

Before actual fusion, the instance objects are pre-processed, following the steps in [42]. This includes confidence thresholding, confidence based sorting, spatial resizing and padding of the instance-specific mask logits and box coordinates to the relevant input size, i.e. from $H_I \times W_I$ to $H \times W$, and a non-maximum suppression based on the overlap and confidence of boxes. After filtering, there remain N^{th} instances. The pre-processing is illustrated in Fig. 4.

Fusion for Things

For the fusion of *things*, all three input components are considered, even if the specific class is not further partitionable. In this case, the generic background class of the part head, can still support this hypothesis during fusion. The individual instance objects guide our fusion process, however during actual fusion, all three predictions are treated equally.

Given a single one out of the $N^{th} = N^{th}_{np} + N^{th}_{p}$ pre-processed *things* instance of class c with its mask logits MLI, we first use the resized bounding box to mask the corresponding prediction from the semantic head. Precisely, class c is sliced out of the semantic logits S and all values outside of the bounding box are set to zero to obtain the masked semantic logits MLS.

In case class c is partitionable, then the corresponding subset of size $C_{p,c}$ of the part logits P is selected from the part segmentation head, e.g. for an instance of class $c = person$, the part logits for head, torso, legs, and arms ($C_{p,human} = 4$) are selected. These logits are again masked by the corresponding bounding box to produce the third masked logits for parts MLP. If class c can not be segmented into parts, the background class from the part logits is selected instead and masked likewise. In order to make the fusion operation feasible, we replicate MLS and MLI to match the number of channels in MLP. E.g., a person instance contains four parts (head, arms, torso, legs), thus MLP is of shape $4 \times W \times H$. Therefore, MLS and MLI are replicated 4 times to match the shape of MLP. If the instance is not partitionable, MLP consists of the background class only and therefore MLS and MLI are not replicated.

At this point, we have obtained three sets of masked logits. We are now fusing these individual logits to obtain the fused logits for classes with parts FLP and class without parts $FLNP$.

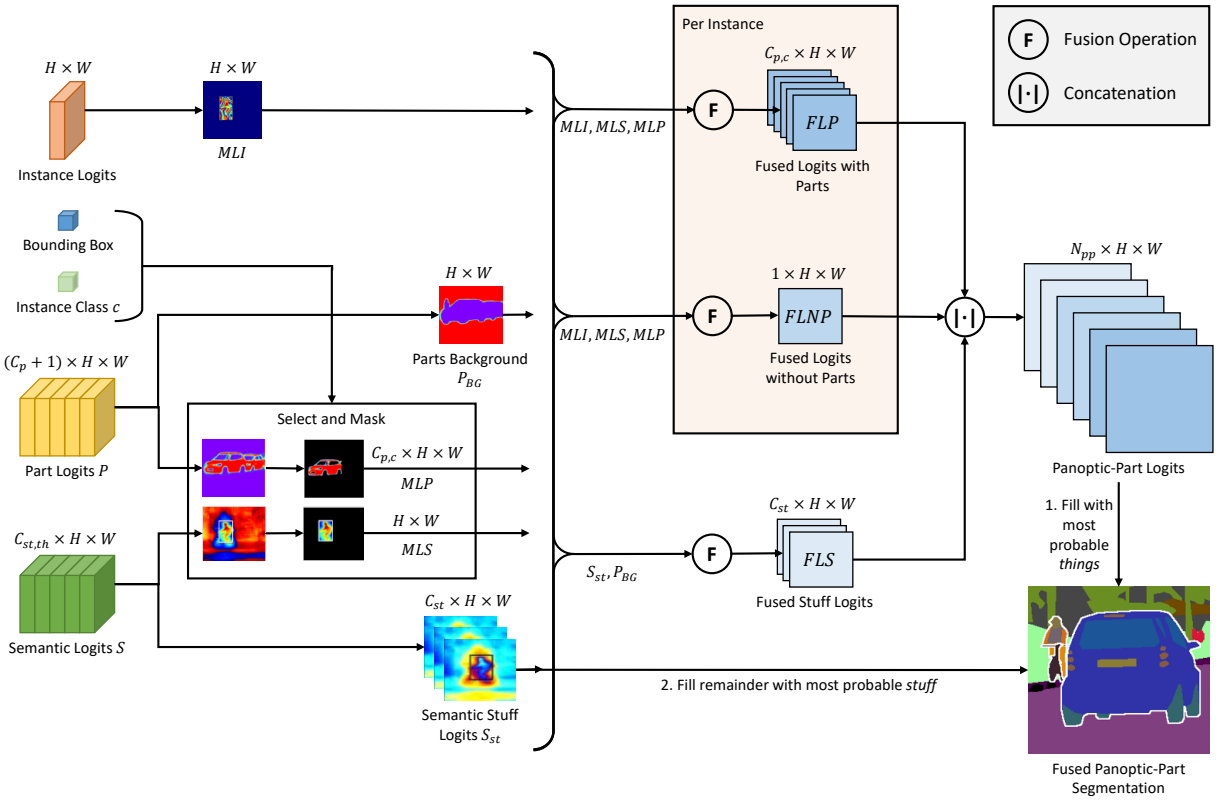


Fig. 5: Illustration of our proposed joint fusion module. For simplicity, we illustrate the process for a single instance object. Semantic, instance, and part predictions are equally balanced and combined

Fusion Operation

To compute the fused logits for any of the cases, we propose a uniformed fusion operation. This operation computes the sum of the sigmoid of the masked logits and the sum of the masked logits and calculates the Hadamard product of both. The procedure is formalized in Eq. 1:

$$FL\,(MLL) = \left(\sum_{l \in MLL} \sigma(l)\right) \odot \left(\sum_{l \in MLL} l\right) \quad (1)$$

In this equation, $\sigma(\cdot)$ denotes the sigmoid function, $\odot$ denotes the Hadamard product, and MLL is a set of equally shaped masked logits which are supposed to be fused, e.g. $MLL = \{MLS, MLI, MLP\}$. This equation describes a generalized version of the fusion proposed by Mohan and Valada [42] that handles arbitrarily many logits.

Fusion for Stuff

To generate the fused logits FLS for the *stuff* classes, each of the C_{st} channels from the semantic head are selected and fused with the background channel of the part head in the same manner, i.e. according to Eq. 1, but this time with only two sets of logits (no instance information). As mentioned, the same concept would also apply for *stuff* that is partitionable, i.e. selecting the corresponding parts, replicating the *stuff* logits, followed by pair-wise fusion.

Overall Fusion

All three fused logits, FLP, $FLNP$, and FLS, are concatenated along the channel dimension to obtain the intermediate logits, in which each of the N_{pp} channels represents a valid panoptic-part label (see Section 3.1. The total number of N_{pp} label candidates depends on the number of things N^{th} predicted by the instance head and the number of parts of their classes $C_{p,c}$. We

produce an intermediate panoptic-part prediction by taking the *argmax* of these intermediate logits. Precisely, during fusion there will be $N_{pp} = C_{st} + N^{th}_{np} + \sum_{c \in N^{th}_{p}} C_{p,c}$ candidate logits. Finally, we fill an empty canvas with the most probable panoptic-part label for all *things* and the remaining areas are filled with the prediction for *stuff* classes extracted from the semantic segmentation head. During fusion, the fused score increases if the predictions of all three heads are consistent, and likewise it is decreased if the predictions do not match with each other.

Post-Processing

Areas of *stuff* classes below a minimum threshold $min_{st} = 2048$ pixels are filtered out, as in [42].

4 Experiments and Results

In this section, we will first introduce the relevant datasets and then provide more details on the implementation and training of our model. Afterwards, we compare our JPPF to previous work and investigate our design choices in ablative experiments.

Datsets

For most of our experiments, we use the recently introduced Cityscapes Panoptic Parts (CPP) and Pascal Panoptic Parts (PPP) datasets [10, 40]. CPP provides pixel-level annotations for 19 semantic categories, of which 11 are *stuff* and 8 are *things* classes. Out of the 8 *things*, 5 classes include annotations at the part level. There are 2975 images for training and 500 for validation in this finely annotated dataset. PPP consists of 20 *things* and 80 *stuff* classes. Part-level annotations are provided for 16 of the 20 *things*. As in previous work [40], we only consider a subset of 59 object classes for training and evaluation, including 20 *things* 39 *stuff* classes, and 58 part classes. These parts are detailed by Michieli et al. [41] and Zhao et al. [64]. PPP consists of a total of 10103 images which are divided into 4998 images for training and 5105 for validation. Next to CPP and PPP, we perform some experiments on a variety of other datasets to demonstrate how our method generalizes across domains.

Metrics

For the evaluation of individual semantic and part segmentations, we apply the mean Intersection-over-Union (mIoU), and the mean Average Precision (mAP) for instance segmentations. For the complete evaluation of combined panoptic-part segmentation, we use the Part Panoptic Quality (PartPQ) [10], which is an extension of the Panoptic Quality (PQ) that was proposed by Kirillov et al. [20]. Because the authors of [29] identified limitations in the expressiveness and interpretability of the PartPQ metric, they have introduced the Part-Whole Quality (PWQ), which we will also consider in our experiments.

Training and Implementation Details

For the Cityscapes data, we use images of the original resolution, i.e. 1024×2048 pixels, and resize the input images of PPP to 384×512 pixels for training. We perform data augmentation, scaling and hyperparameter initialization as in EfficientPS [42]. We use a multi-step learning rate (lr) and train our network by Stochastic Gradient Descent (SGD) with a momentum of 0.9. For the CPP and PPP, we use an initial lr of 0.07 and 0.01, respectively. We begin the training with a warm-up phase in which the lr is increased linearly from $\frac{1}{3} \cdot lr$ up to lr within 200 iterations. The weights of all InPlace-ABN layers [1] are frozen, and we train the model for 10 additional epochs with a fixed learning rate of 10^{-4}. Finally, we unfreeze the weights of the InPlace-ABN layers and train the model for $50k$ iterations beginning with lr of 0.07 (CPP) and 0.01 (PPP), and reduce lr after $32k$ and $44k$ iteration by a factor of 10. Four GPUs are used for the training with a batch size of 2 per GPU for CPP and 8 per GPU for PPP. Our feature backbone is the most recent version of EfficientNet – EfficientNet-L2 [56]. This is in contrast to our previous work [17], in which we have used the preliminary EfficientNet-B5 [52]. We initialize the backbone with weights pre-trained on COCO [33]. The impact of this initialization is quantified in Table 1.

4.1 Comparison to State-of-the-Art

Our comparison to previous work and state-of-the-art considers the initially introduced baseline in [10] and the more recent unified transformer-based architecture of Li et al. [28, 29]. The baseline by

Table 1: Comparison of our updated model on CPP with and without pre-trained weights

Pretrained	PartPQ	PWQ
no	61.4	66.7
COCO [33]	61.4	67.3

Table 2: Comparison of EfficientPS trained on Panoptic Cityscapes and on Cityscapes Panoptic Parts (CPP)

Network	Data	PQ
EfficientPS [42]	Panoptic CS [6]	63.9
	CPP [10]	62.2

de Geus et al. [10] uses the panoptic labels of the Cityscapes dataset [6] to train a panoptic segmentation network. Since this data is slightly different from the actual panoptic-part dataset (CPP), a direct, fair comparison is not possible. This deviation is indicated in Table 2 that shows the results of EfficientPS [42] trained on Cityscapes vs. CPP. To make the baseline comparable in terms of data, we re-implement the baseline and train it on the same data. The re-implementation consists of EfficientPS [42] for panoptic segmentation, and our part segmentation network with a separate backbone (c.f. Section 3.2.1). Top-down merging is then used to combine the two independent results into a panoptic-part segmentation. The re-implementation and results are in line with our previous work in [17].

Finally, we compare our previous and updated model with JPPF to the reproduced baseline, the official baselines of de Geus et al. [10], and multiple variants of both versions of the Panoptic-PartFormer (PPF) [28, 29]. The official baseline consists of EfficientPS [42] and BSANet [64] with top-down merging. The results of this comparison on CPP and PPP are shown in Table 3 for single-scale and multi-scale inference.

On CPP with single-scale testing, JPPF improves the accuracy significantly compared to the reproduced baseline. We surpass the reproduced baseline by 3.7 percentage points (pp) in overall PartPQ and by 5.3 pp in PartPQ_P with our updated backbone. Similarly for multi-scale testing, our updated model outperforms the baseline by 3.1 pp and 6.1 pp in PartPQ and PartPQ_P, respectively. Our JPFF even outperforms the strong transformer-based competitor PPF and the non-peer reviewed extension PPF++ in terms of PartPQ and PWQ by a small margin. Especially for areas that can be segmented into parts, we achieve more accurate results, indicating the increased consistency after our fusion and leading to a higher PWQ metric. Interestingly, for these two metrics (PartPQ_P and PWQ), even our single-scale results better over the multi-scale results of any competitor.

For PPP, our model outperforms the top-down combination of DeepLabV3+ [4] and Mask RCNN [16] (*Baseline-1*), even though this baseline was trained with the original Pascal parts and Pascal panoptic segmentation datasets, which provide more annotations. *Baseline-2* (top-down merging of DeepLabV3-ResNeSt269 [2, 61], DetectoRS [47], and BSANet [64]) yields even better results because of the more advanced backbones, and hence has a higher representational capacity. Similarly, the more sophisticated transformer of PPF (and PPF++) together with powerful backbone models achieves the best results for PartPQ and PWQ. However, in partionable areas ($PartPQ_P$), we significantly outperform the baselines on PPP. We believe that this advantage can be attributed to the balanced integration of parts in our fusion module. In comparison to top-down merging, our design is also slightly favorable in terms of density, as presented in Table 5.

From Fig. 6, we can see that our proposed fusion is able to segment the parts of very small and distant object classes reliably. Also, our proposed fusion solves some typical problems of top-down merging, which are the bifurcation of *things* by *stuff* and the inconsistent parts within *things*. As illustrated in Fig. 6, our fusion gets rid of unknown regions within objects by giving equal priority to all three individual predictions. In Figs. A1 and A2 we provide more examples and a visual comparison to PPF [28]. There, we also present failure cases of our model to provide insights into its limitations. In some cases, especially on PPP, PPF produces finer details compared to our approach. In cluttered areas where small objects occlude each other, our JPPF seems to perform favorably.

Table 3: Comparison of results for panoptic-part segmentation on Cityscapes and Pascal Panoptic Parts [40]. *P* and *NP* refer to areas with and without part labels, respectively. The best result per data setting and metric is highlighted in bold. *indicates our reproduced baseline (details in Section 4.1)

Method	Backbone	PartPQ			PWQ
		All	P	NP	
Cityscapes Panoptic Parts, Single-Scale					
Baseline*	EfficientNet-B5 [52]	57.7	44.2	62.5	–
JPPF (Ours)	EfficientNet-B5 [52]	59.6	47.7	63.8	66.1
	EfficientNet-L2 [56]	**61.4**	**49.5**	**65.7**	**67.3**
Cityscapes Panoptic Parts, Multi-Scale					
Baseline [10]	EfficientNet [52], ResNet101 [15]	60.2	46.1	65.2	–
PPF [28]	ResNet50 [15]	57.4	43.9	62.2	60.5
	Swin [36]	61.9	45.6	68.0	65.3
PPF++ [29]	ResNet50 [15]	59.2	42.5	65.1	62.1
	Swin [36]	62.3	46.0	68.2	65.7
	ConvNext [37]	63.1	46.4	**69.1**	66.5
JPPF (Ours)	EfficientNet-B5 [52]	61.8	50.8	65.7	**68.2**
	EfficientNet-L2 [56]	**63.3**	**52.2**	67.2	**68.2**
Pascal Panoptic Parts, Single-Scale					
Baseline-1 [10]	ResNet50 [15]	31.4	47.2	26.0	–
Baseline-2 [10]	ResNeSt269 [61]	38.3	51.6	33.8	–
PPF [28]	ResNet50 [15]	37.8	–	–	40.2
	ResNet101 [15]	39.3	–	–	41.3
PPF++ [29]	ResNet50 [15]	42.2	–	–	45.2
	ResNet101 [15]	42.4	–	–	46.0
	Swin [36]	**49.3**	–	–	52.7
	ConvNext [37]	48.6	–	–	**54.2**
JPPF (Ours)	EfficientNet-B5 [52]	32.3	48.3	26.9	45.6
	EfficientNet-L2 [56]	40.5	**58.5**	**34.4**	53.7

Table 4: Comparison of three independent encoders to our design with a shared feature encoder with different backbones on Cityscapes Panoptic Parts

Method	Backbone	Semantic mIoU	Instance AP	Part mIoU
Independent Networks	EfficientNet-B5 [52]	78.1	37.3	74.5
Shared Features (Ours)	EfficientNet-B5 [52]	80.5	37.9	77.0
	EfficientNet-L2 [56]	81.7	40.7	76.6

4.2 A Single Shared Encoder

As part of our contribution, we aim to unify semantic, instance, and part segmentation and jointly learn all three in a single, unified model. We are validating that these three tasks benefit from a shared feature representation by comparing the individual predictions before fusion to three separate equivalent networks that have been trained individually with different encoders. As shown in Table 4, both models with a single, shared encoder

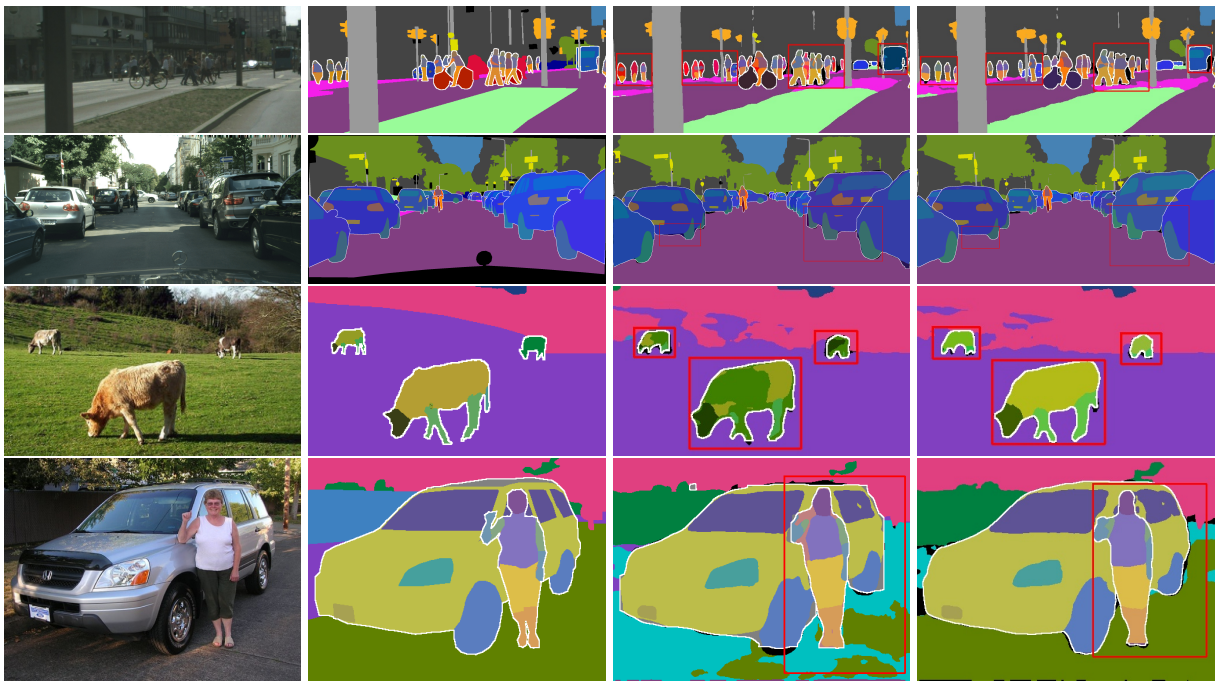


Fig. 6: Qualitative results of our proposed model on Citscapes Panoptic Parts (first two rows) and Pascal Panoptic Parts (last two rows) compared to our reproduced baseline, ground-truth and the reference image. *indicates the reproduced baseline which is detailed in Section 4.1. The results for our JPPF are obtained with the backbone of the previous version. The graphic is adopted from [17]. More visual examples with our updated backbone for both datasets are provided in the appendix in Figs. A1 and A2

surpass the individual models in all three tasks. To no surprise, the more recent version of EfficientNet [52] produces already better initial results for our fusion. This experiment clearly indicates that using a shared encoder enables the learning of a common feature representation, resulting in more accurate individual outcomes of each head, which are also more consistent by design due to the shared representation.

4.3 Joint Fusion

Next, we compare our joint fusion module to the previously presented top-down merging strategy [10] in Table 5. It is important to note, that even the recently published state-of-the-art method PPF [28, 29] uses this merging strategy. The proposed fusion module surpasses the top-down merge in terms of PartPQ, PartPQ_P, PartPQ_NP, and PWQ on all datasets and settings. Even though our proposed fusion is admittedly only slightly better in some cases, the joint fusion produces also denser results than the uni-directional merge, indicating the improved consistency. The advantages of our fusion are mainly reflected for the results in areas that are partitionable. Since the *things* with part labels are limited in CPP, the impact is best observed on the PPP dataset. On this data, our proposed fusion module is significantly better. Specifically for our design, PartPQ_P is improved by 10.5 pp and 14.9 pp for the different backbones. That is a relative improvement of about 28 % and 44 %.

4.4 Density, Run Time, and Model Size

Our JPPF produces results, which are at least as dense as the top-down merging (see Table 5). We further assessed the inference time of our proposed model with JPPF, and the results are displayed in Table 6. It is evident that the top-down merging

Table 5: Comparison between the uni-directional top-down merge [10] and our proposed joint fusion module using various input sources on Cityscapes and Pascal Panoptic Parts [40]

Model	Backbone	Merging/ Fusion	Before Merge/Fusion Sem. mIoU	Inst. AP	Part mIoU	After Merge/Fusion PartPQ All	P	NP	PWQ	Density [%]
Cityscapes Panoptic Parts, Single-Scale										
JPPF	EfficientNet-B5 [52]	Top-Down	80.5	37.9	77.0	59.5	47.5	63.7	66.0	99.1
		JPPF				59.6	47.7	63.8	66.1	99.3
	EfficientNet-L2 [56]	Top-Down	81.7	40.7	76.6	61.1	48.7	65.5	67.1	99.3
		JPPF				61.4	49.5	65.7	67.3	99.5
Cityscapes Panoptic Parts, Multi-Scale										
JPPF	EfficientNet-B5 [52]	Top-Down	81.8	41.3	78.5	61.6	50.7	65.5	68.2	99.2
		JPPF				61.8	50.8	65.7	68.2	99.5
	EfficientNet-L2 [56]	Top-Down	80.0	40.3	76.3	62.7	50.7	67.0	67.8	99.2
		JPPF				63.3	52.2	67.2	68.2	99.5
Pascal Panoptic Parts, Single-Scale										
JPPF	EfficientNet-B5 [52]	Top-Down	46.0	39.1	54.4	29.0	37.8	26.0	42.3	89.6
		JPPF				32.3	48.3	26.9	45.6	92.1
	EfficientNet-L2 [56]	Top-Down	52.3	47.3	62.2	30.6	43.6	26.2	49.2	92.7
		JPPF				40.5	58.5	34.4	53.7	92.7

Table 6: Detailed run-time analysis of our JPPF and the baseline on full resolution images of Cityscapes Panoptic Parts usisng a Nvidia A100 GPU. *indicates the reproduced baseline which is detailed in Section 4.1

Method	Backbone	Feature Extraction [ms]	Individual Predictions [ms]	Fuse/Merge [ms] Panoptic Fusion	Merge	JPPF	Total Inference [ms]
Baseline*	EfficientNet-B5 [52]	52	202	118	484	–	856
JPPF (Ours)	EfficientNet-B5 [52]	26	202	–	–	161	389
	EfficientNet-L2 [56]	59	202	–	–	161	422

Table 7: Comparison of model complexity for an input of size 1200×800 pixels

Method	Backbone	Params [M]	FLOPs [G]
PPF [28]	ResNet50 [15]	37.4	185.8
	Swin [36]	100.3	408.5
PPF++ [29]	ResNet50 [15]	45.6	215.4
	ConvNext [37]	120.2	519.5
JPPF (Ours)	Eff.Net-B5 [52]	44.2	211.6
	Eff.Net-L2 [56]	406.2	889.7

requires more than twice the time compared to our proposed fusion. To obtain panoptic-part segmentation as proposed by de Geus et al. [10], one must first perform a panoptic fusion and then combine it with the part segmentation, which adds an extra overhead. Table 7 shows that our approach, in terms of model size and number of floating point operations (FLOPs), is comparable to previous work for the smaller backbones (ResNet50 [15] and EffificentNet-B5 [52]). Our updated backbone (EfficientNet-L2 [56]) is significantly larger and more complex than the initial version, but adds only little overhead in terms of run time (see Table 6).

4.5 Generalization

Since our fusion module is free of learned parameters (i.e. there are no trainable layers involved), it is independent of its input and supposed to exhibit a good generalization to unseen domains.

Input Images Results of JPPF

Fig. 7: Visual results of our JPPF on the Indian Driving Dataset (IDD) [55] without fine-tuning

However, the entire model (including the backbone and individual prediction heads) is restricted by the typical rigidity of deep neural networks and their sensitivity to shifts in the distribution. Yet, our complete model generalizes well to other datasets, as demonstrated in Fig. 7 even though it has only been trained on CPP for this experiment. We show the generalization for a more extensive set of various other datasets without fine-tuning in Figs. A3 to A6, including a typical failure case. A qualitative comparison in terms of generalization between PPF++ [29] and our method is provided in Fig. A7. For all our results, we have resized the input images to fit the size of the original CPP dataset, i.e. 1024×2048 pixels.

5 Limitations

During the thorough evaluation of our approach, we have identified some remaining limitations, which we discuss here. Though, our fusion operation treats the logits of the three prediction heads equally, the overall process is mainly guided by the prediction of the instance branch. I.e. the detected *things* and their classes and bounding boxes control the information flow during fusion, mostly. With respect to balance and importance of the individual predictions, this is a limitation. Additionally, as a side effect of this fact, the fusion of *things* is limited to the area within each bounding box. Thus, for very large objects that are not fully covered by the bounding box, the fusion can not compensate the initially too small estimated area of these objects. Furthermore, we have identified a theoretical limitation in the fusion operation in Eq. 1 itself. Our generalized version is indeed able to handle an arbitrarily sized set of input logits, however there is no explicit mechanism to balance (normalize) the fused output for different numbers of inputs. In practice, highly confident inputs produce similar highly confident outputs when they are consistent, independent of the number of inputs (e.g. 2 or 3). For less confident areas, the imbalance between the fused *stuff* (two input logits) and the fused *things* (three input logits) might be an issue. Finally, we notice that post-processing step (filtering out small *stuff* areas) is the remaining factor that hinders fully dense predictions, i.e. a valid (not necessarily correct)

panoptic-part label for every pixel of the input image.

6 Conclusion

JPPF is a versatile fusion operation that combines semantic, instance, and part segmentation effectively into a consistent panoptic-part segmentation. It consistently outperforms uni-directional top-down merging for various input sources, e.g. our previous and updated model. Our design with the updated backbone and joint fusion module surpasses the baseline on all datasets, achieves state-of-the-art results on Cityscapes Panoptic Parts, and ranks in between the first and second versions of the Panoptic-PartFormer on Pascal Panoptic Parts. The advantages of our proposed approach become most visible for partitionable areas. The increased consistency in the prediction of our model is highlighted by its increased density. We leave it for future work to find suitable solutions for the limitations that have been discussed.

Acknowledgments. This work was partially funded by the Federal Ministry of Education and Research Germany under the project DECODE (01IW21001).

References

[1] Bulo SR, Porzi L, Kontschieder P (2018) In-place activated batchnorm for memory-optimized training of dnns. In: Conference on Computer Vision and Pattern Recognition (CVPR)

[2] Chen L, Papandreou G, Schroff F, et al. (2017) Rethinking atrous convolution for semantic image segmentation. arXiv preprint arXiv:170605587

[3] Chen L, Collins MD, Zhu Y, et al. (2018) Searching for efficient multi-scale architectures for dense image prediction. Advances in Neural Information Processing Systems (NeurIPS)

[4] Chen LC, Zhu Y, Papandreou G, et al. (2018) Encoder-decoder with atrous separable convolution for semantic image segmentation. In: European conference on computer vision (ECCV)

[5] Cheng B, Collins MD, Zhu Y, et al. (2020) Panoptic-deeplab: A simple, strong, and fast baseline for bottom-up panoptic segmentation. In: Conference on Computer Vision and Pattern Recognition (CVPR)

[6] Cordts M, Omran M, Ramos S, et al. (2016) The cityscapes dataset for semantic urban scene understanding. In: Conference on Computer Vision and Pattern Recognition (CVPR)

[7] Dong J, Chen Q, Xia W, et al. (2013) A deformable mixture parsing model with parselets. In: International Conference on Computer Vision (ICCV)

[8] Gao N, Shan Y, Wang Y, et al. (2019) Ssap: Single-shot instance segmentation with affinity pyramid. In: International Conference on Computer Vision (ICCV)

[9] Geiger A, Lenz P, Stiller C, et al. (2013) Vision meets robotics: The kitti dataset. The International Journal of Robotics Research (IJRR)

[10] de Geus D, Meletis P, Lu C, et al. (2021) Part-aware panoptic segmentation. In: Conference on Computer Vision and Pattern Recognition (CVPR)

[11] Gong K, Liang X, Li Y, et al. (2018) Instance-level human parsing via part grouping network. In: European Conference on Computer Vision (ECCV)

[12] Gong K, Gao Y, Liang X, et al. (2019) Graphonomy: Universal human parsing via graph transfer learning. In: Conference on Computer Vision and Pattern Recognition (CVPR)

[13] Hariharan B, Arbeláez P, Girshick R, et al. (2014) Simultaneous detection and segmentation. In: European Conference on Computer Vision (ECCV)

[14] Hariharan B, Arbeláez P, Girshick R, et al. (2015) Hypercolumns for object segmentation and fine-grained localization. In: Conference on Computer Vision and Pattern Recognition (CVPR)

[15] He K, Zhang X, Ren S, et al. (2016) Deep residual learning for image recognition. In: Conference on Computer Vision and Pattern Recognition (CVPR)

[16] He K, Gkioxari G, Dollár P, et al. (2017) Mask r-cnn. In: International Conference on Computer Vision (ICCV)

[17] Jagadeesh SK, Schuster R, Stricker D (2023) Multi-task fusion for efficient panoptic-part segmentation. In: International Conference on Pattern Recognition Applications and Methods (ICPRAM)

[18] Jiang Y, Chi Z (2018) A cnn model for semantic person part segmentation with capacity optimization. Transactions on Image Processing (T-IP)

[19] Jiang Y, Chi Z (2019) A cnn model for human parsing based on capacity optimization. Applied Sciences

[20] Kirillov A, Girshick R, He K, et al. (2019) Panoptic feature pyramid networks. In: Conference on Computer Vision and Pattern Recognition (CVPR)

[21] Kirillov A, He K, Girshick R, et al. (2019) Panoptic segmentation. In: Conference on Computer Vision and Pattern Recognition (CVPR)

[22] Ladicky L, Torr PH, Zisserman A (2013) Human pose estimation using a joint pixel-wise and part-wise formulation. In: Conference on Computer Vision and Pattern Recognition (CVPR)

[23] Li J, Raventos A, Bhargava A, et al. (2018) Learning to fuse things and stuff. arXiv preprint arXiv:181201192

[24] Li P, Xu Y, Wei Y, et al. (2020) Self-correction for human parsing. Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)

[25] Li Q, Arnab A, Torr PH (2017) Holistic, instance-level human parsing. British Machine Vision Conference (BMVC)

[26] Li Q, Arnab A, Torr PH (2018) Weakly- and semi-supervised panoptic segmentation. In: European conference on computer vision (ECCV)

[27] Li Q, Qi X, Torr PH (2020) Unifying training and inference for panoptic segmentation. In: Conference on Computer Vision and Pattern Recognition (CVPR)

[28] Li X, Xu S, Yang Y, et al. (2022) PanopticPartFormer: Learning a Unified Model for Panoptic Part Segmentation. In: European Conference on Computer Vision (ECCV)

[29] Li X, Xu S, Yang Y, et al. (2023) Panopticpartformer++: A unified and decoupled view for panoptic part segmentation. arXiv preprint arXiv:230100954

[30] Li Y, Qi H, Dai J, et al. (2017) Fully convolutional instance-aware semantic segmentation. In: Conference on Computer Vision and Pattern Recognition (CVPR)

[31] Liang X, Gong K, Shen X, et al. (2018) Look into person: Joint body parsing & pose estimation network and a new benchmark. Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)

[32] Lin K, Wang L, Luo K, et al. (2020) Cross-domain complementary learning using pose for multi-person part segmentation. Transactions on Circuits and Systems for Video Technology (T-CSVT)

[33] Lin TY, Maire M, Belongie S, et al. (2014) Microsoft coco: Common objects in context. In: European Conference on Computer Vision (ECCV)

[34] Liu H, Peng C, Yu C, et al. (2019) An end-to-end network for panoptic segmentation. In: Conference on Computer Vision and Pattern

Recognition (CVPR)

[35] Liu S, Sun Y, Zhu D, et al. (2018) Cross-domain human parsing via adversarial feature and label adaptation. In: Conference On Artificial Intelligence (AAAI)

[36] Liu Z, Lin Y, Cao Y, et al. (2021) Swin transformer: Hierarchical vision transformer using shifted windows. In: International Conference on Computer Vision (ICCV)

[37] Liu Z, Mao H, Wu CY, et al. (2022) A convnet for the 2020s. In: Conference on Computer Vision and Pattern Recognition (CVPR)

[38] Luo P, Wang X, Tang X (2013) Pedestrian parsing via deep decompositional network. In: International Conference on Computer Vision (ICCV)

[39] Luo X, Su Z, Guo J, et al. (2018) Trusted guidance pyramid network for human parsing. In: ACM International Conference on Multimedia (ACM-MM)

[40] Meletis P, Wen X, Lu C, et al. (2020) Cityscapes-panoptic-parts and pascal-panoptic-parts datasets for scene understanding. arXiv preprint arXiv:200407944

[41] Michieli U, Borsato E, Rossi L, et al. (2020) Gmnet: Graph matching network for large scale part semantic segmentation in the wild. In: European Conference on Computer Vision (ECCV)

[42] Mohan R, Valada A (2021) EfficientPS: Efficient Panoptic Segmentation. International Journal of Computer Vision (IJCV)

[43] Neuhold G, Ollmann T, Rota Bulo S, et al. (2017) The mapillary vistas dataset for semantic understanding of street scenes. In: International Conference on Computer Vision (ICCV)

[44] O Pinheiro PO, Collobert R, Dollár P (2015) Learning to segment object candidates. Advances in Neural Information Processing Systems (NeurIPS)

[45] Pont-Tuset J, Arbelaez P, Barron JT, et al. (2016) Multiscale combinatorial grouping for image segmentation and object proposal generation. Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)

[46] Porzi L, Bulo SR, Colovic A, et al. (2019) Seamless scene segmentation. In: Conference on Computer Vision and Pattern Recognition (CVPR)

[47] Qiao S, Chen LC, Yuille A (2020) Detectors: Detecting objects with recursive feature pyramid and switchable atrous convolution. arXiv preprint arXiv:200602334

[48] Ren S, He K, Girshick RB, et al. (2015) Faster R-CNN: towards real-time object detection with region proposal networks. Advances in Neural Information Processing Systems (NeurIPS)

[49] Ruan T, Liu T, Huang Z, et al. (2019) Devil in the details: Towards accurate single and multiple human parsing. In: Conference on Artificial Intelligence (AAAI)

[50] Sakaridis C, Dai D, Van Gool L (2021) Acdc: The adverse conditions dataset with correspondences for semantic driving scene understanding. In: International Conference on Computer Vision (ICCV)

[51] Sofiiuk K, Barinova O, Konushin A (2019) Adaptis: Adaptive instance selection network. In: International Conference on Computer Vision (ICCV)

[52] Tan M, Le Q (2019) Efficientnet: Rethinking model scaling for convolutional neural networks. In: International Conference on Machine Learning (ICML)

[53] Tian Z, He T, Shen C, et al. (2019) Decoders matter for semantic segmentation: Data-dependent decoding enables flexible feature aggregation. In: Conference on Computer

Vision and Pattern Recognition (CVPR)

[54] Valada A, Mohan R, Burgard W (2018) Self-supervised model adaptation for multimodal semantic segmentation. International Journal of Computer Vision (IJCV)

[55] Varma G, Subramanian A, Namboodiri A, et al. (2019) Idd: A dataset for exploring problems of autonomous navigation in unconstrained environments. In: Winter Conference on Applications of Computer Vision (WACV)

[56] Xie Q, Luong MT, Hovy E, et al. (2020) Self-training with noisy student improves imagenet classification. In: Conference on Computer Vision and Pattern Recognition (CVPR)

[57] Xiong Y, Liao R, Zhao H, et al. (2019) Upsnet: A unified panoptic segmentation network. In: Conference on Computer Vision and Pattern Recognition (CVPR)

[58] Yang L, Song Q, Wang Z, et al. (2019) Parsing r-cnn for instance-level human analysis. In: Conference on Computer Vision and Pattern Recognition (CVPR)

[59] Yang T, Collins MD, Zhu Y, et al. (2019) Deeperlab: Single-shot image parser. arXiv preprint arXiv:190205093

[60] Yu F, Chen H, Wang X, et al. (2020) Bdd100k: A diverse driving dataset for heterogeneous multitask learning. In: Conference on Computer Vision and Pattern Recognition (CVPR)

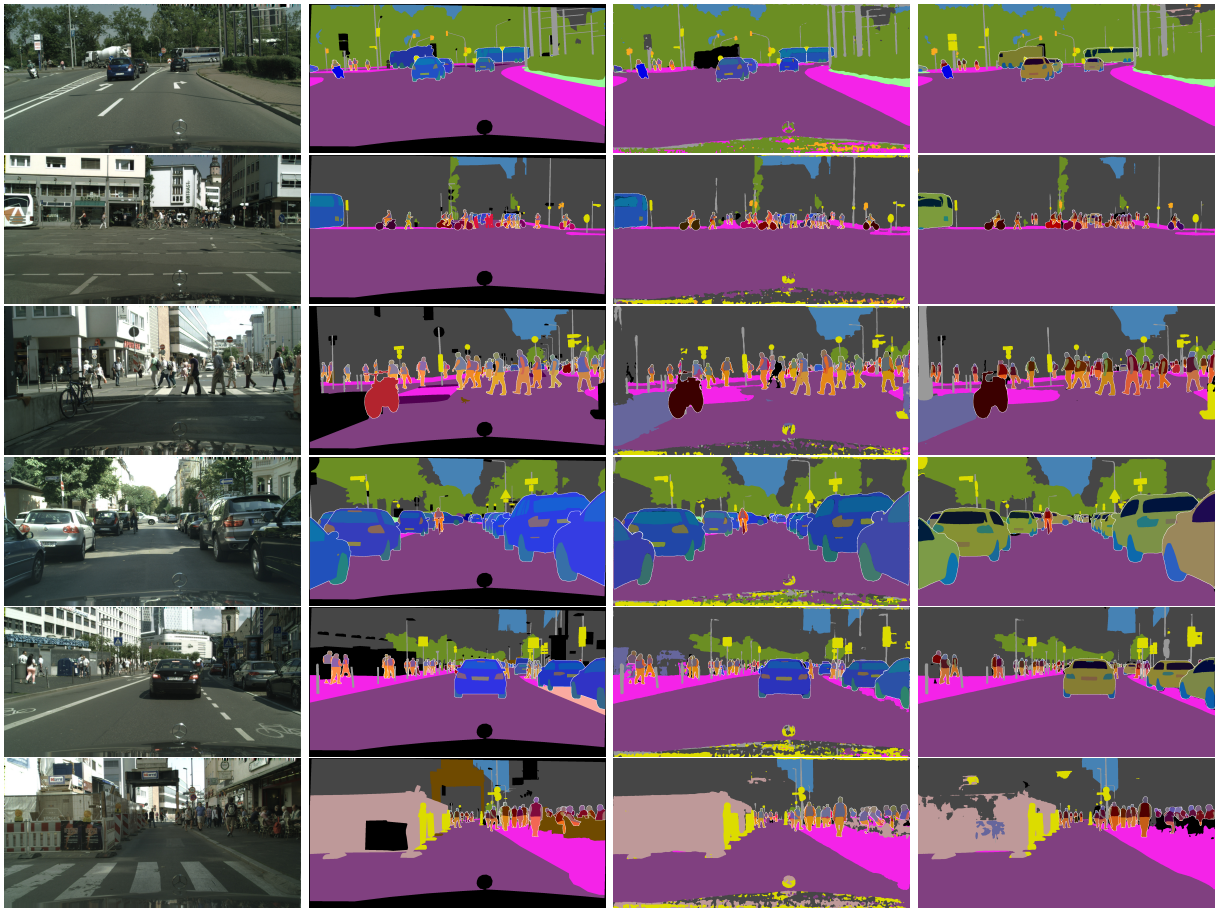
[61] Zhang H, Wu C, Zhang Z, et al. (2022) Resnest: Split-attention networks. In: Conference on Computer Vision and Pattern Recognition (CVPR)

[62] Zhao H, Shi J, Qi X, et al. (2017) Pyramid scene parsing network. In: Conference on Computer Vision and Pattern Recognition (CVPR)

[63] Zhao J, Li J, Cheng Y, et al. (2018) Understanding humans in crowded scenes: Deep nested adversarial learning and a new benchmark for multi-human parsing. In: ACM International Conference on Multimedia (ACM-MM)

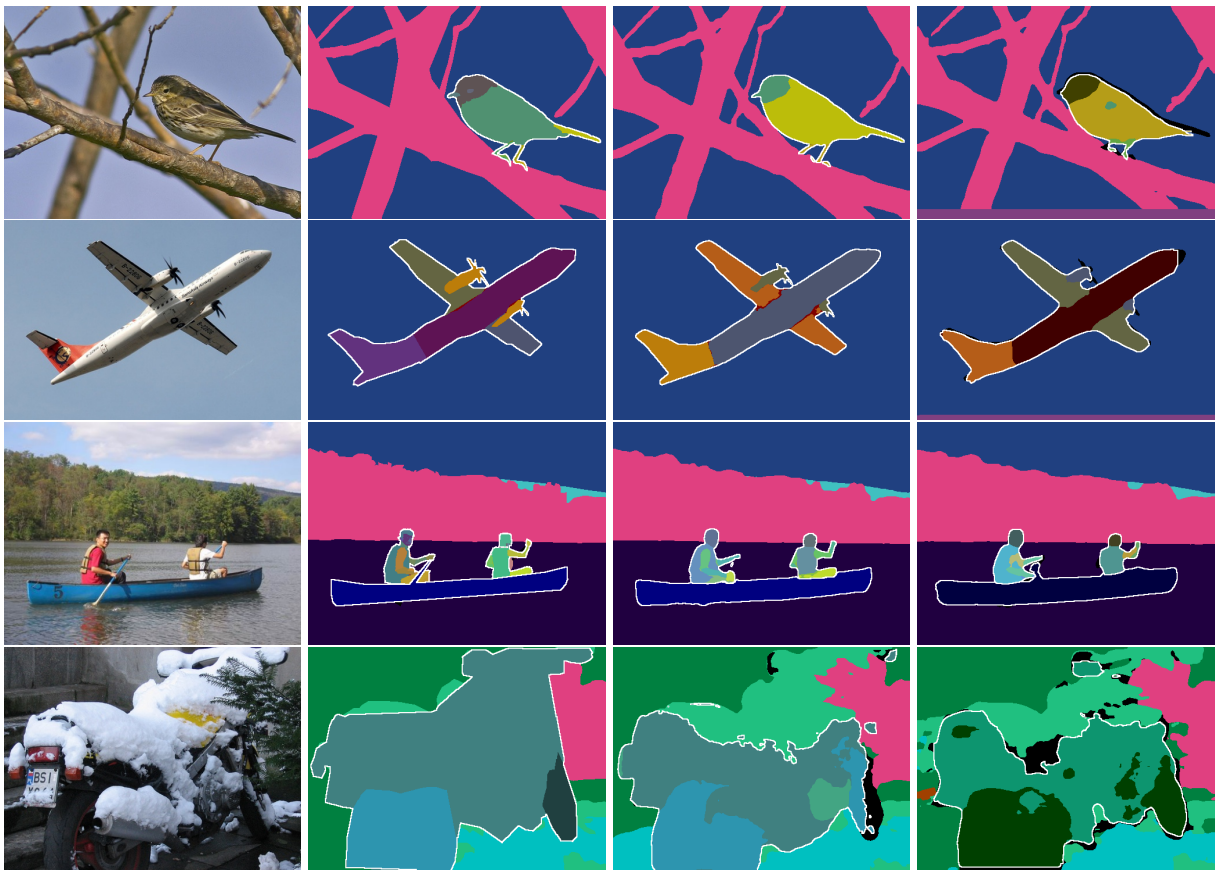
[64] Zhao Y, Li J, Zhang Y, et al. (2019) Multi-class part parsing with joint boundary-semantic awareness. In: International Conference on Computer Vision (ICCV)

Appendix A Additional Visualizations



Original Image | Ground-truth | PPF [28] | JPPF (Ours)

Fig. A1: Additional qualitative results of our proposed model and PPF [28] on CPP [40]



Original Image | Ground-truth | PPF [28] | JPPF (Ours)

Fig. A2: Additional qualitative results of our proposed model and PPF [28] on PPP [40]

Input Images Results of JPPF

Fig. A3: Visual results of our JPPF on the ACDC dataset [50] without fine-tuning

Input Images Results of JPPF

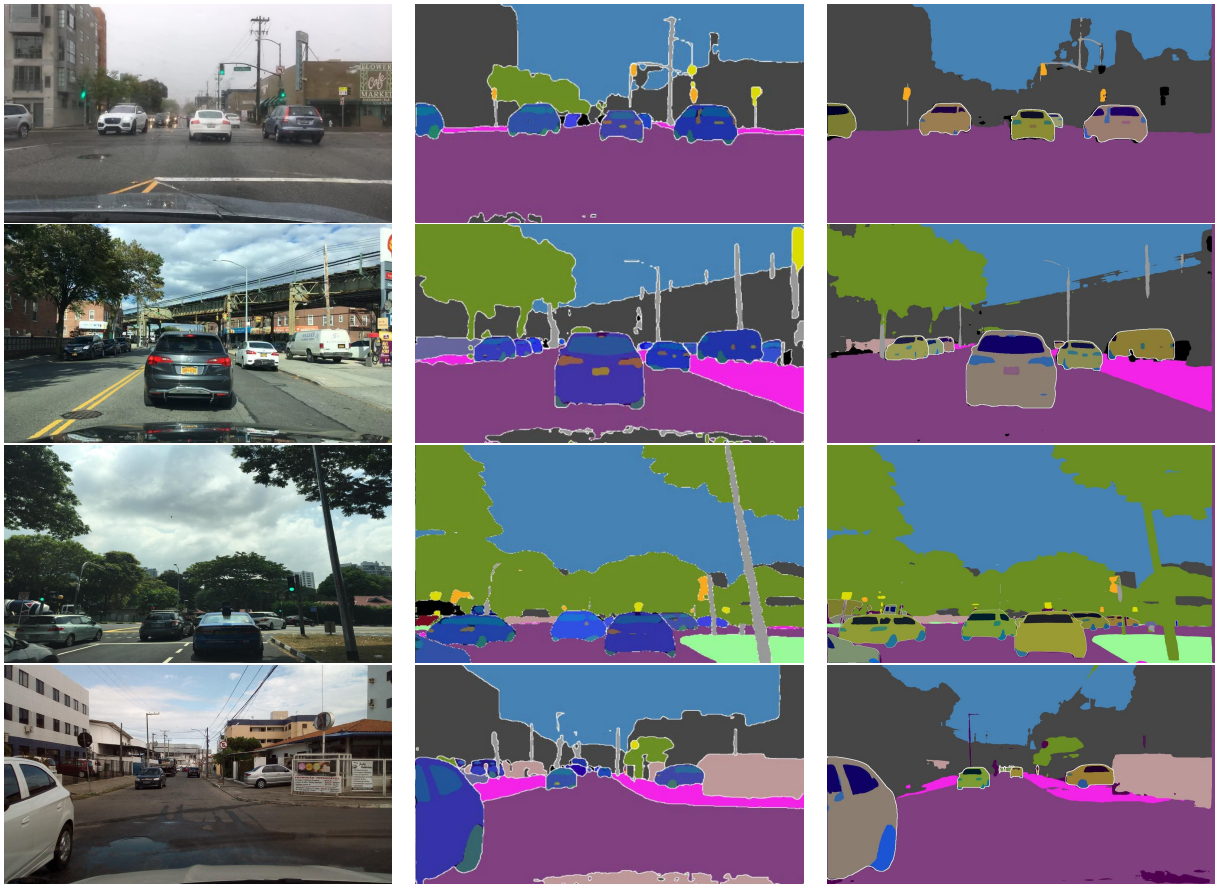
Fig. A4: Visual results of our JPPF on the BDD100K dataset [60] without fine-tuning. The second example shows a failure case, in which our model is not able to properly generalize to the unseen data

Input Images Results of JPPF

Fig. A5: Visual results of our JPPF on the KITTI dataset [9] without fine-tuning

Input Images Results of JPPF

Fig. A6: Visual results of our JPPF on the Mapillary Vistas dataset [43] without fine-tuning



Input Images PPF++ [29] Results of JPPF

Fig. A7: We visually compare the generalization capabilities of PPF++ [29] and our JPPF on BDD100K [60] (first two rows) and Mapillary Vistas [43] (last two rows) without fine-tuning