Combinatorial Optimization for Panoptic Segmentation: An End-to-End Trainable Approach

Anonymous Author(s)
Affiliation
Address
email

Abstract

We propose an end-to-end trainable architecture for simultaneous semantic and instance segmentation (a.k.a. panoptic segmentation) consisting of a convolutional neural network and an asymmetric multiway cut problem solver. The latter solves a combinatorial optimization problem that elegantly incorporates semantic and boundary predictions to produce a panoptic labeling. Our formulation allows to directly maximize a smooth surrogate of the panoptic quality metric by backpropagating the gradient through the optimization problem. Experimental evaluation shows improvement of end-to-end learning w.r.t. comparable approaches on CityScapes and COCO datasets. Overall, our approach shows the utility of using combinatorial optimization in tandem with deep learning in a challenging large scale real-world problem and showcases benefits and insights into training such an architecture end-to-end.

1 Introduction

Panoptic segmentation is the task of simultaneously segmenting different semantic classes and instances of the same class [32]. Panoptic segmentation is challenging since neural networks (NN) may produce conflicting predictions (i.e. boundaries separating instances that are not closed contours, instance voting schemes with multiple maxima per instance etc.). Therefore most approaches combine NNs with a post-processing step to compute a final panoptic segmentation that resolves the conflicting evidence produced by NNs. In general, joint training of NNs with post-processing algorithms is an active research area. In our work we propose end-to-end training for panoptic segmentation, our post-processing being a combinatorial optimization problem.

In this work we pursue the bottom-up approach building segmentations directly from pixels and combine CNNs with the asymmetric multiway cut problem (AMWC) [36]. The latter is an elegant combinatorial optimization problem that combines semantic and affinity predictions and directly produces a panoptic labeling. We train CNN and AMWC jointly so that the supervisory signal for training the CNN is influenced by the computations of the combinatorial optimization stage. The loss we propose to use for this training differs from common lower-level CNN losses and is a smooth surrogate closely corresponding to the final panoptic quality metric [32]. We show in this work how our conceptual contributions, i.e. using AMWC and training on surrogate panoptic loss end-to-end, can be made to work together and yield performance improvements.

The general idea of combining optimization and neural networks and train them end-to-end has recently enjoyed resurgent interest. The fundamental problem for the specific task of combinatorial optimization is that the output of combinatorial problem is 0–1 valued, hence the loss landscape becomes piecewise constant and simply differentiating through a solver is not possible anymore. Several methods have been proposed to address this problem [8, 20, 22, 27, 50, 52]. To our knowledge our work is the first to utilize the perturbation techniques [20, 52] on a large-scale setting with scalable

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

but suboptimal heuristic solvers. We give evidence that end-to-end training works in this setting and gives performance benefits. To this end we propose a robust extension of the backpropagation technique [52] that gives better empirical convergence.

Our architecture is inspired by [11, 13] and consists of a ResNet-50 backbone, a semantic segmentation branch for computing class costs and an affinity branch for boundary predictions. Semantic and affinity costs are taken as input by the AMWC solver that returns a panoptical labeling. We first pre-train semantic and affinity branches with simple cross-entropy losses obtaining a strong baseline that achieves a performance similar or better than other bottom-up approaches [13, 23, 63]. We fine-tune subsequently end-to-end with the AMWC solver and the panoptic surrogate loss via our new robust backpropagation and show further performance improvements.

Current state-of-the-art approaches use very large networks (e.g. Max-DeepLab [59] uses transformers containing more parameters than a ResNet-101). This might lead to the impression that advances in panoptic segmentation require deeper and more sophisticated architecture. We show that our simpler model can be significantly improved by end-to-end learning and argue that simpler models have not yet reached their full potential. Also, our simpler architecture allows for a more controlled setting and makes it easier to identify crucial components in end-to-end training and measure to which extent performance improvements can be achieved.

Contribution.

Optimization for Segmentation: We propose AMWC [36] as an expressive and tractable combinatorial optimization formulation to be used in an end-to-end trainable architecture for panoptic segmentation. We also propose a scalable heuristic for its solution.

Panoptic loss surrogate: We propose a surrogate loss function that approximates the panoptic loss metric and can be used in our end-to-end training.

Backpropagation: We give an extension of the perturbation technique [52] for backpropagating gradients through combinatorial solvers, improving training with suboptimal heuristic solvers.

Experimental validation: We conduct experiments on CityScapes [15] and COCO [41] and show the benefits of end-to-end training against comparable approaches.

We will make our code available upon acceptance of the paper.

2 Related Work

2.1 Panoptic segmentation

We categorize panoptic segmentation approaches into three categories: (i) top-down methods predict information on the pixel-level and then use post-processing to produce a segmentation, (ii) bottom-up methods proceed by first identifying regions of interest (ROI) and subsequently basing segmentation on them and (iii) hybrid methods combine bottom-up and top-down ideas. For a general overview of recent segmentation methods we refer to [45]. Here we will restrict to panoptic segmentation tasks.

Top-down. Recent works include [10, 31, 32, 37, 46, 53, 54, 67, 69]. This principle has also been used with weak supervision [39]. As a drawback, top-down approaches use ROIs which are mostly axis-aligned and so they can be in-efficient for scenarios containing deformable objects [58].

Bottom-up. Panoptic-DeepLab [13] based on [68] proposes a single-stage neural network architecture which combines instance center of mass scores with semantic segmentation to compute panoptic segmentation. They use post-processing similar to Hough-voting [5], obtaining great results and reducing the gap to top-down approaches. Subsequently, Axial-DeepLab [60] made improvements using an attention mechanism to enlarge the receptive field using the post-processing scheme of [68].

The methods SSAP [23] and SMW [63] are most similar to our as they also use semantic and affinity scores with a graph partitioning algorithm where SMW [63] additionally uses Mask-RCNN [25] and SSAP solves multiple graph partitioning problems in coarse-to-fine manner. Older works such as [33, 42] use graph partitioning schemes but only for the instance segmentation task.

Hybrid. The approaches [40, 63] use both bottom-up (affinity scores) and top-down (bounding boxes) sources of information. Conditional convolution [58] was used in [59]. Transformers are used in [10] and combined with Max-DeepLab in a sophisticated architecture, achieving remarkable results. They used a surrogate for the panoptic quality metric along with an instance discrimination

loss similar to [65]. However, Max-DeepLab imposes an upper bound on the maximum number of instances in an image and requires thresholding low confidence predictions.

In summary, bottom-up methods are generally simpler than top-down ones and require fewer hyperparameters. However, they lack global context and are generally outperformed by top-down approaches. As a solution Axial-DeepLab [60] reduce this gap by incorporating long range context.

Almost all of the above-mentioned approaches use multiple loss functions (see [29] for a possible solution), need thresholds for getting rid of low confidence predictions or assume an upper bound on the number of instances and therefore require hyperparameter tuning. To achieve end-to-end training, approaches of [10, 40, 59] design mechanisms embedded in the NNs which can compute panoptic segmentations directly but require more complicated architectures.. Except for the above works, other approaches delegate this task to a post-processing module which does not participate in training. The motivation of our work is based on prioritizing ease-of-use and simplicity. Therefore we have chosen a bottom-up approach and propose a method for end-to-end training with only one loss and no ad-hoc downstream refinements of the segmentation.

2.2 End-to-end Training

Recently there has been great interest in training neural networks end-to-end with additional layers for problem-specific constraints and prior knowledge. The works [24, 35] provide an extensive survey and insights. [21] gives an overview of multiple approaches for learning graphical model parameters.

The focus of our work is on using an optimization problem as a layer in neural networks. Hence, we will mainly cover approaches for this scenario. They can be categorized as follows:

Unrolling. For training NNs together with cheap and differentiable iterative algorithms (or for algorithms that can be made differentiable e.g. by smoothing), straightforwardly computing gradients is the most simple approach. This has been done for K-means [62] bipartite matching [70], conditional random fields [3, 20, 57, 71], non-linear diffusion for image restoration [12] and ranking and sorting [17]. The interesting study [14] shows that under some stability conditions backpropagation through the last few steps of iterative procedures is enough to get good estimates of gradients.

Implicit Function Theorem. In case solutions satisfy fixed point conditions (e.g. KKT conditions) the implicit function theorem can be used to compute gradients. This was done for quadratic programs in [2], embedding MaxSAT in neural networks [61], a large class of convex optimization problems [1], smoothed top-k selection via optimal transport [66] and deep equilibrium models [4].

Problem-specific methods. Specialized approaches for backpropagating for specific problems were investigated for submodularity [19] (e.g. using a graph-cut layer), belief propagation [34], dynamic programming [44] and nearest neighbor selection [51].

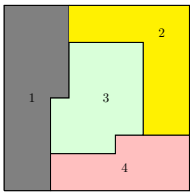
Gradient Estimation by Perturbation. Perturbing the objective of an optimization problem for learning has been proposed in [7, 38, 47] for graphical model parameters. In [8, 16, 49] perturbation is used in the forward pass to get a differentiable estimate of the solution. Perturbing the objective in the direction of loss decrease has been proposed in [20] for backpropagating through graphical model inference, in [43] to estimate gradients through a structured loss and in [52] to backpropagate through combinatorial optimization problems. The latter was used for ranking [55] and graph matching [56].

3 Method

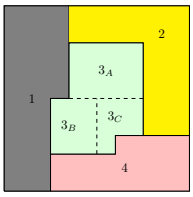
Our architecture shown in Figure 2 is comprised of two stages: (i) a CNN to compute semantic class and affinities for boundary predictions followed by (ii) an AMWC optimization layer producing the final panoptic labeling. We describe below our CNN architecture, the AMWC problem and finally the end-to-end training w.r.t. panoptic surrogate loss.

3.1 CNN Architecture

Our CNN architecture (see Figure 2) is comprised of the following parts: a shared ResNet-50 backbone pre-trained on ImageNet [18] producing feature maps for the subsequent semantic and affinity branch. Our CNN architecture corresponds to Panoptic-Deeplab [13] with the exception of a



(a) MWC



(b) AMWC

Figure 1: Exemplary MWC and AMWC problems with 4 classes ($K = 4$). MWC is a special case of AMWC when $P = \varnothing$. For $P = \{3\}$ we get an AMWC problem where class 3 is partitioned into subclusters (instances) 3_A, 3_B and 3_C.

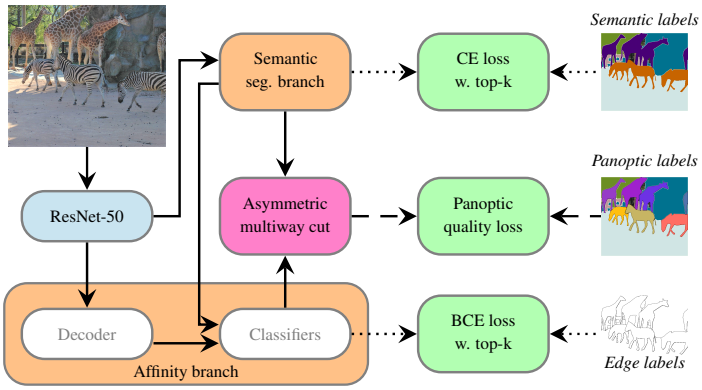


Figure 2: Overview of our architecture: The input is first given to the ResNet-50 [26] backbone. Computed features are fed into a semantic segmentation branch to predict class scores and to a affinity branch to predict object boundaries. Costs from both branches are used in the AMWC solver for computing a panoptic labeling. Pre-training of the semantic and affinity branch is done with top-k cross-entropy losses [68] (dotted arrows). End-to-end training uses the panoptic quality loss back-propagated through the AMWC solver (dashed arrows). The computation flow marked by solid lines is for panoptic segmentation, dottes arrow for pre-training and dashed arrows for end-to-end training.

modified instance segmentation branch due to different post-processing (Hough voting for [13] vs. AMWC in our work). This allows for a fair comparison with [13].

Affinity Predictor. The affinity branch predicts for given pairs of pixels whether they belong to the same instance. It takes two sources of inputs: (i) features from the affinity decoder and (ii) semantic segmentation costs which makes finding boundaries between different classes easier. Gradients of segmentation costs computed from affinity predictors are not backpropagated during training to preclude the affinity branch from influencing the semantic branch.

We take horizontal and vertical edges at varying distances d. For COCO we use $d \in \{1, 4, 16, 32, 64\}$ and for CityScapes $d \in \{1, 4, 16, 32, 64, 128\}$. For each d all corresponding edges are sampled and affinity scores are computed by a dedicated predictor for each distance. For long range edges with $d > 1$ we compute edge features by taking the difference of affinity features of the edge endpoints before sending them to the predictor. This helps in capturing long-range context. Additional architectural details can be found in the appendix.

3.2 (Asymmetric) Multiway Cut

Multiway cut (MWC) [9] is a combinatorial optimization problem for graph partitioning defined on a graph. In MWC a pre-defined number of classes is given and each node is assigned to one. The cost of a class assignment is given by node and edge affinity costs that give the preference of a node belonging to a certain class and endpoints of the edge to belong to the same class respectively. Hence, the multiway cut can be straightforwardly used to formulate semantic segmentation, each MWC class corresponding to a semantic class.

The AMWC problem was introduced in [36] as an extension of MWC. AMWC additionally allows to subdivide some classes into an arbitrary number of sub-clusters. This allows to model segmenting a given semantic class into multiple instances for panoptic segmentation.

Mathematically, MWC and AMWC are defined on a graph $G = (V, E)$ together with edge weights $c_E : E \to \mathbb{R}$ and node costs $c_V : V \times \{1, \dots, K\} \to \mathbb{R}$, where K is the number of classes. The edge affinities c_E indicate preference of edge endpoints to belong to the same cluster, while the node costs c_V indicate preference of assigning nodes to classes. A set of classes $P \subseteq [K]$ is given indicating

classes that can be partitioned into multiple clusters. For MWC we have $P = \varnothing$ while for AMWC $P \subseteq [K]$. Let B be the set of valid boundaries, i.e. edge indicator vectors of partitions of V

$$\mathrm{B} = \{y : E \to \{0,1\} : \exists C_1 \dot\cup \ldots \dot\cup C_l \text{ s.t. } y(ij) = 1 \Leftrightarrow \exists l \neq l' \text{ and } i \in C_l, j \in C_{l'}\}\,, \tag{1}$$

where the number of clusters is arbitrary and $\dot\cup$ is the disjoint union. The MWC and AMWC optimization problems can be written as

$$\begin{aligned} \min_{x:V\to\{1,\ldots,K\}, y\in\mathrm{B}} \quad & \textstyle\sum_{i\in V} c_V(x(i)) + \sum_{ij\in E} c_E(ij)\cdot y(ij) \\ \text{s.t.} \quad & y(ij) = 0,\ \text{if } x(i) = x(j) \notin P \\ & y(ij) = 1,\ \text{if } x(i) \neq x(j) \end{aligned} \tag{2}$$

The above constraints stipulate that y produces a valid clustering of the graph compatible with the node labeling x, i.e. boundaries implied by y align with class boundaries defined by x and non-partitionable classes not in P do not possess internal boundaries. The AMWC can be though of as a special case of InstanceCut [33] that has class-dependent edge affinities, which, however, makes it less scalable. Illustrations of MWC and AMWC are given in Figure 1.

Given a feasible solution (x, y) satisfying the constraints in (2), the panoptic labeling $z : V \to \{1, \ldots, L\}$ is computed by connected components w.r.t. y, i.e. $z(i) = z(j) \Leftrightarrow y(ij) = 0,\ \forall ij \in E$.

Optimization algorithms for efficiently computing possibly suboptimal solutions for (2) are given in the appendix. Note that, contrary to other approaches for panoptic segmentation such as [58, 59, 67] AMWC neither has an upper bound on the number of instances nor suffers from computational bottlenecks in this regard. It also does not require thresholding to get rid of low confidence predictions.

3.3 End-to-end training

To train our whole architecture with the AMWC solver end-to-end we first introduce a new robust variant of the perturbation technique for backpropagation [52] which works well for our setting of a large-scale problem and suboptimal solver. Second, we introduce a smooth panoptic loss surrogate. Last, we show how to backpropagate gradients for the panoptic loss surrogate through a MWC layer.

Robust Perturbation for Backpropagation. The fundamental difficulty of backpropagating through a combinatorial optimization problem is that the loss landscape is piecewise constant, since the output of the combinatorial problem is 0–1 valued. To handle this difficulty, generally applicable perturbation techniques [6, 20, 27, 43, 52] have been proposed. They work by taking finite differences of solutions with perturbations of the original problem. The work [52] interprets this as creating a continuous interpolation of the non-continuous original loss landscape.

The second difficulty is that, due to large size and NP-hardness of AMWC, we use a heuristic suboptimal solver that does not in general deliver optimal solutions. Therefore, we propose a multi-scale extension of [52] for increased robustness that works well in our setting.

Assuming an optimization layer $\mathcal{W} : \mathbb{R}^n \to \{0,1\}^n, c \mapsto \arg\min_{x\in\mathcal{X}} \langle c, x\rangle$ where $\mathcal{X} \subset \{0,1\}^n$ is some constraint set and a loss function $L : \mathbb{R}^n \mapsto \mathbb{R}$, the gradient $\frac{\partial(L\circ\mathcal{W})}{\partial c}$, where $L \circ \mathcal{W}$ is the composition of L and $\mathcal{W}$, is zero almost everywhere. In [52] a continuous interpolation $(L \circ \mathcal{W})_\lambda$ is proposed where $\lambda > 0$ is an interpolation range. The gradient w.r.t. the interpolation is computed by

$$\frac{\partial(L\circ\mathcal{W})_\lambda}{\partial c} = \frac{1}{\lambda}\left(\mathcal{W}(c + \lambda\nabla L(\mathcal{W}(y))) - \mathcal{W}(c)\right)\,. \tag{3}$$

While [52] report that a large interval of interpolation ranges λ work well on their test problems with optimal solvers, we have not been able to confirm this for our suboptimal heuristic that only gives approximately good solutions to $\mathcal{W}$. Therefore, we propose to use a multi-scale loss and its gradient

$$(L\circ\mathcal{W})_{avg} := \frac{1}{N}\sum_{i=1}^{N}(L\circ W)_{\lambda_i}\,, \qquad \frac{\partial(L\circ\mathcal{W})_{avg}}{\partial c} = \frac{1}{N}\sum_{i=1}^{N}\frac{\partial(L\circ\mathcal{W})_{\lambda_i}}{\partial c}\,. \tag{4}$$

where λ_i are sampled uniformly on some interval. While the robust backpropagation formula (4) needs multiple calls to the optimization oracle $\mathcal{W}$, they can be computed in parallel. In practice the computation time for a backward pass will hence not increase.

Panoptic Quality Surrogate Loss. Panoptic quality (PQ) [32] is a size-invariant evaluation metric defined between a set of predicted masks and ground-truth masks for each semantic class $l \in [K]$. For

each class, it requires to match predicted and object masks to each other w.r.t intersection-over-union (IoU) since instance labels are permutation invariant. A pair of predicted and ground truth masks p and g of the same class l is matched (i.e. true-positive) if $IoU(p, g) \geq 0.5$. We write $(p, g) \in TP_l$. For the unmatched masks, each prediction (ground-truth) is marked as false positive FP_l (false negative FN_l). Since at most one match exists per ground truth mask, this matching process is well-defined [32]. The PQ metric is defined as the mean of class specific PQ scores

$$PQ_l = \frac{\sum_{(p,g)\in TP_l} IoU(p, g)}{|TP_l| + 0.5(|FP_l| + |FN_l|)} \,. \tag{5}$$

Note that the PQ score (5) can be arbitrarily low just by the presence of small sized false predictions [13, 53, 67]. A common practice to avoid such issue is to reject small predictions before computing the PQ score with some dataset specific size thresholds, before evaluation. However, this rejection mechanism is not incorporated during training.

The PQ metric (5) cannot be straightforwardly used for training due to the discontinuity of the hard threshold based matching and the rejection mechanism. Therefore we replace the hard threshold matching process for each class l by computing correspondences via a maximum weighted bipartite matching with IoU as weights. The corresponding matches are $\overline{TP}_l$, the unmatched prediction masks $\overline{FP}_l$ and the unmatched ground truth masks $\overline{FN}_l$. The hard thresholding is smoothed via the soft thresholding function $h(u) = \frac{u^4}{u^4+(1-u)^4}$ centered around 0.5. The small prediction rejection mechanism for mask p is smoothed via $\sigma_l(p) = [1 + \exp(-0.1(1^T p - t_l))]^{-1}$ centered at area threshold t_l for class l. The overall surrogate PQ for class l is

$$\overline{PQ}_l = \frac{\sum_{(p,g)\in\overline{TP}_l} h(IoU(p, g))\sigma_l(p)IoU(p, g)}{\sum_{(p,g)\in\overline{TP}_l} h(IoU(p, g))\sigma_l(p) + 0.5(\sigma_l(p)|\overline{FP}_l| + |\overline{FN}_l|)} \,. \tag{6}$$

Note that our approach also allows one to also learn the class-specific thresholds t_l.

Transformation to Multiway Cut. In order to directly train with the panoptic loss surrogate (6) via the backpropagation formula (4) our optimization oracle $\mathcal{W}$ must take the gradient of the loss function as perturbation. This is not the case for AMWC, since the gradient of (6) affects node costs of individual instances separately (i.e. they work on panoptic labels), but AMWC assumes node costs are equal for all instances of a semantic class (i.e. it works on class labels). Therefore we transform the AMWC problem into a lifted MWC problem that has a class for each panoptic label in the ground truth. This allows to separately optimize directly in panoptic label space and return a gradient to the semantic and affinity branch.

For the backward pass described in Algorithm 1 we define the following notation: Let J be the number of classes for the lifted MWC problem and $m : [J] \to [K]$ the mapping from panoptic labels onto the corresponding semantic class. Let x, y, z be semantic, edge and panoptic labels respectively, c_V and c_E class and edge costs for AMWC. Algorithm 1 computes the gradient w.r.t. the simple backpropagation formula (3). For the robust backprop (4) the algorithm has to be called multiple times with the corresponding interpolation ranges λ. An illustration of the gradient computation is given in Figure 3.

Algorithm 1: BACKWARD PASS

Input : $\frac{\partial L}{\partial z}$, c_V, c_E, solution x, y, m, λ
Output : $\frac{\partial L}{\partial c_V}$, $\frac{\partial L}{\partial c_E}$

1. Transform node costs to panoptic and perturb:
$$c'_V(l) = c_V(m(l)) + \lambda \frac{\partial L}{\partial z}(l),\ \forall l \in [J]$$
2. Multiway cut on panoptic label space:
$$(z_p, y_p) = \text{MWC}(c'_V, c_E)$$
3. Perturbed class labels:
$$x_p(i) = m(z_p(i)), \forall i \in V$$
4. Compute node cost gradients: $\frac{\partial L}{\partial c_V} = \frac{1}{\lambda}(x_p - x)$
5. Compute edge cost gradients: $\frac{\partial L}{\partial c_E} = \frac{1}{\lambda}(y_p - y)$
6. **return** $\frac{\partial L}{\partial c_V}$, $\frac{\partial L}{\partial c_E}$

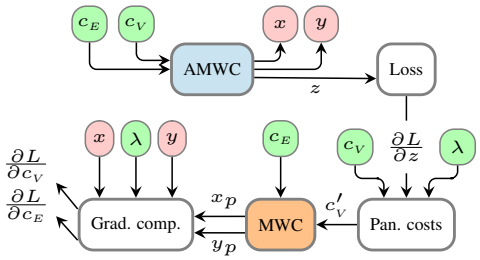


Figure 3: Gradient computation for c_V, c_E in end-to-end learning: AMWC produces semantic, edge, panoptic labels x, y, z resp. Perturbations of panoptic label costs c'_V are computed and sent to the MWC solver together with the original edge costs c_E. Results are used to compute and return the gradients.

Line 1 in Alg. 1 merges two sources of information i.e., preference of the loss L on panoptic labels and current class costs c_V. Note that the edge costs are not perturbed. Afterwards, the perturbed panoptic labels are converted back to class labels on line 3 before computing the gradients. Ablation study w.r.t using simpler losses on the output of AMWC solver is shown in the appendix.

4 Experiments

All baselines are trained on NVIDIA Quadro RTX 8000 GPUs with $48GB$ memory each. For end-to-end training we use one Tesla P40 with $24GB$ memory and a 32 core CPU to solve all AMWC problems in the batch in parallel.

4.1 Datasets

We train and evaluate our method on the Cityscapes [15] and COCO [41] panoptic segmentation datasets. We cross-validate our method on the validation split and evaluate the same model on the testing split for which ground-truth is not publicly provided and requires evaluation via online servers.

Cityscapes. Contains traffic related images of resolution 1024×2048 where training, validation and testing splits have 2975, 500, and 1525 images for training, validation, and testing, respectively. It contains 8 'thing' and 11 'stuff' classes. During training we use random scale augmentation and crop to 512×1024 resolution as done in Panoptic-DeepLab. During evaluation the input images are sent at original resolution. The values of small segment rejection thresholds (used during both training and inference) are $200, 2048$ for 'thing'and 'stuff'class resp. Lastly, to handle larger occlusions we additionally use affinities at a distance of 128.

COCO. Is more diverse and contains $118k$, $5k$, and $20k$ images for training, validation, and testing, resp. The dataset has 80 'thing' and 53 'stuff' classes. During training random scale augmentation is also used with a crop size of 640×640 resolution as in [13]. The values of small segment rejection thresholds (used during both training and inference) are $200, 4096$ for 'thing'and 'stuff'class resp. During evaluation the input images are resized to 640×640 resolution.

4.2 Training

We closely follow the implementation of Panoptic-DeepLab in [64] (based on Pytorch [48]), use the provided ImageNet pre-trained ResNet-50 backbone and the same learning rate parameters for training our baseline model. The Adam optimizer [30] is used for all our experiments.

Resolution. The CNNs produce an output with $1/4$-th the resolution in every dimension w.r.t input images, similar to Panoptic-DeepLab. This reduced input size is maintained for AMWC (instead of upsampled) to reduce computation time during end-to-end training and evaluation. The panoptic labels computed by the AMWC solver are upsampled during evaluation. Since these labels are discrete, upsampling may misalign object boundaries and small ground-truth objects can potentially be missed as well. While this can put our method at a disadvantage, our end-to-end training offsets this by achieving panoptic quality even better than the performance at finest resolution of comparable methods.

Baseline Pre-training. We pre-train the CNN architecture as a baseline model and for achieving a good initialization for the subsequent end-to-end training. This also allows us to measure the additional gain by end-to-end training. In pre-training we apply the weighted top-k cross-entropy loss [68] to each affinity predictor separately and also to the semantic segmentation branch. Since the main objective of the affinity classifier should be to predict instance boundaries we increase the loss by a factor of 4 for edges where at least one endpoint belongs to a 'thing' class. Additionally, we also increase the semantic and affinity loss weights of small objects by a factor of 3 following [13].

We train CityScapes on one GPU with batch-size 12 for $250k$ iterations, with initial learning rate 0.001 and the decay strategies of Panoptic-DeepLab. Training takes around 8 days. COCO is trained on four GPUs with a total batch-size of 48 for $240k$ iterations using the same learning rate parameters as above. Training takes around 11 days.

End-to-end training. For end-to-end training we use only the panoptic quality surrogate loss (6) and fine-tune the classification layer of the semantic segmentation branch and the last two layers of all affinity classifiers. All batch normalization parameters [28] are frozen. We train with batch size of

Table 1: Results on Cityscapes (above) and COCO (below) on validation and testing splits. R-X: ResNet-X, X-71: Xception-71, †: Mask selection (e.g. by Mask-RCNN), *: Uses test-time augmentation. (-) Marks the results which are not reported for that setting.

Method	Backbone	PQ^{test}	PQ^{test}_{th}	PQ^{test}_{st}	PQ^{val}	PQ^{val}_{th}	PQ^{val}_{st}
			Cityscapes				
EfficientPS [46]†	Custom	-	-	-	63.9	66.2	60.7
Panoptic-FPN [31]†	R-101	-	-	-	58.1	52.0	62.5
UPSNet [67]†	R-50	-	-	-	59.3	54.6	62.7
Unify. PS [40]†	R-50	61.0	52.7	67.1	61.4	54.7	66.3
Panoptic-DL [13]	X-71	60.7	-	-	63.0	-	-
Axial-DL [60]	Axial-L	62.7	53.4	69.5	63.0	-	-
Panoptic-Dl [13]	R-50	58.0	-	-	60.2	51.1	66.9
Unify. PS [40]	R-50	-	-	-	59.0	50.2	65.3
SMW [63]†	Multiple	-	-	-	59.3	50.6	65.7
SSAP [23]*	R-101	58.9	48.4	66.5	61.1	55.0	-
SSAP [23]	R-50	-	-	-	56.6	49.2	-
Our baseline	R-50	56.7	46.0	64.5	58.5	48.3	66.0
Our end-to-end	R-50	59.5	50.8	65.9	62.2	55.5	67.1
			COCO				
Max-DeepLab [59]	MaX-S	49	54	41.6	-	-	
Panoptic-FPN [31]†	R-101	40.9	48.3	29.7	40.3	47.5	29.5
UPSNet [67]†	R-50	-	-	-	42.5	48.5	33.4
Unify. PS [40]†	R-50	43.6	48.9	35.6	43.4	48.6	35.5
Panoptic-DL [13]	X-71	38.8	-	-	39.7	43.9	33.2
Axial-DL [60]	Axial-S	42.2	46.5	35.7	41.8	46.1	35.2
SSAP [23]*	R-101	36.9	40.1	32	36.5	-	-
Panoptic-DL [13]	R-50	35.2	-	-	35.1	-	-
Our baseline	R-50	34.2	35.2	32.8	34.3	34.9	33.4
Our end-to-end	R-50	36.6	40.1	31.4	36.3	39.3	31.7

24 for 12000 iterations for Cityscapes and 4000 iterations for COCO. To approximate the gradient (4) we use relatively large values of λ compared to [52] since in-exact optimization might not react to small perturbations correctly (for example the backward pass solution might not even be equal to the one from the forward pass for $\lambda \to 0$). We also observed more stable training curves for larger values of N. We use $N = 5$ in our experiments. We do not learn the class-specific area thresholds t_l in (6) as it leads to over-fitting on the training data.

4.3 Results

We compare panoptic quality (in terms of percentage) on both testing PQ^{test} and validation PQ^{val} splits of Cityscapes and COCO datasets, see Table 1. For the testing splits evaluation requires submission to an online server. We also show performance on 'thing' classes PQ_{th}, and stuff classes PQ_{st} separately. To allow a fair comparison, we restrict ourselves to results of competing approaches which are closest to our setting i.e., without test-time augmentation, similar number of parameters in the network, not utilizing other sources of training data etc. For a better overall comparison, we also consider at least one state-of-the-art work from each other type of method (top-down, hybrid etc.).

First, our end-to-end trained model improves by 3% for CityScapes and 2% for COCO in comparison to our baseline model. This is evidence our panoptic loss surrogate and end-to-end training works. Especially, performance on the 'thing' classes improves which have internal boundaries. We argue this is mainly due to better training of the affinity branch, which benefits more from the AMWC supervisory signal. A sample qualitative comparison between baseline and end-to-end trained model can be seen in Figure 5, where end-to-end training shows clear visible improvements. The methods

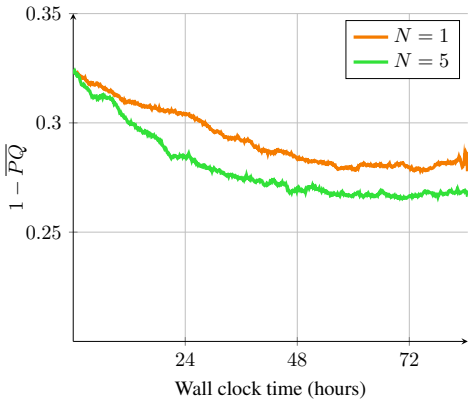
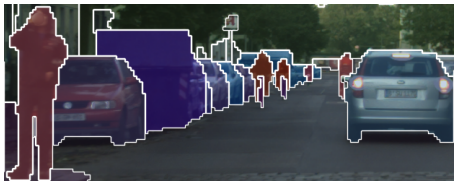


Figure 4: Comparison of panoptic quality surrogate loss on Cityscapes for different values of loss interpolation parameter N in (4). With $N = 5$ convergence is reached faster, eventhough we do not parallelize over N.

(a) Our baseline. All three bicycles are not detected, false detection above the right car and near the person on the left.



(b) After end-to-end training. Better localization and bicycles are correctly detected

Figure 5: Comparing panoptic labels on Cityscapes test set, boundary of each instance is drawn in white. (Best viewed digitally and in color).

SSAP [23], SMW [63] are closest to ours in-terms of the post-processing, and Panoptic-DeepLab in-terms of architecture resp. Our end-to-end model outperforms SSAP even in a setting where SSAP uses test-time augmentation and a larger backbone. SMW reports results only on Cityscapes using two independent DeepLabV3+ models and a Mask-RCNN. We outperform it with our end-to-end approach while still using a simpler model. While Panoptic-Deeplab outperforms our baseline model, our end-to-end training outperforms it in turn.

In Figure 4 we plot the PQ surrogate loss (6) decrease during end-to-end training using different numbers of interpolation parameter N in (4). Our proposed improvement in the backpropagation scheme of [52] trains faster and achieves lower training loss.

Inference times. Although parallelization can be simply done during training, our approach has no real-time performance during inference as the AMWC solver requires around 2 seconds per image from Cityscapes.

5 Conclusion

We have proposed an end-to-end trainable approach for panoptic segmentation incorporating a combinatorial optimization layer for post-processing and directly learning a panoptic surrogate loss. Our choice has lead to a simple and elegant formulation with a minimal number of hyperparameters. We argue that learning through combinatorial optimization layers is possible and leads to improved performance even with simple and suboptimal solvers. However, backpropagation schemes should be suitably augmented for robustness in this case.

While our work suggests that combinatorial optimization is helpful in neural networks, most solvers (including the ones we used) are sequential and executed on CPU, which limits their applicability. For combinatorial optimization to become a more commonly used layer in neural networks, solvers must be designed that are inherently parallel and executable on GPUs.

Broader Impact

This work introduces a new end-to-end trainable architecture for panoptic segmentation, a fundamental task in computer vision used in down-stream tasks. The broader impact of our work depends on the concrete downstream task.

References

[1] Akshay Agrawal, Shane Barratt, and Stephen Boyd. Learning convex optimization models. *arXiv preprint arXiv:2006.04248*, 2020.

[2] Brandon Amos and J Zico Kolter. Optnet: Differentiable optimization as a layer in neural networks. In *International Conference on Machine Learning*, pages 136–145. PMLR, 2017.

[3] Anurag Arnab, Shuai Zheng, Sadeep Jayasumana, Bernardino Romera-Paredes, Måns Larsson, Alexander Kirillov, Bogdan Savchynskyy, Carsten Rother, Fredrik Kahl, and Philip HS Torr. Conditional random fields meet deep neural networks for semantic segmentation: Combining probabilistic graphical models with deep learning for structured prediction. *IEEE Signal Processing Magazine*, 35(1):37–52, 2018.

[4] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. Deep equilibrium models. *arXiv preprint arXiv:1909.01377*, 2019.

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

[6] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

[7] Gedas Bertasius, Qiang Liu, Lorenzo Torresani, and Jianbo Shi. Local perturb-and-map for structured prediction. In *Artificial Intelligence and Statistics*, pages 585–594. PMLR, 2017.

[8] Quentin Berthet, Mathieu Blondel, Olivier Teboul, Marco Cuturi, Jean-Philippe Vert, and Francis Bach. Learning with differentiable perturbed optimizers. *arXiv preprint arXiv:2002.08676*, 2020.

[9] Gruia Calinescu. *Multiway Cut*, pages 567–569. Springer US, Boston, MA, 2008.

[10] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European Conference on Computer Vision*, pages 213–229. Springer, 2020.

[11] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 801–818, 2018.

[12] Yunjin Chen and Thomas Pock. Trainable nonlinear reaction diffusion: A flexible framework for fast and effective image restoration. *IEEE transactions on pattern analysis and machine intelligence*, 39(6):1256–1272, 2016.

[13] Bowen Cheng, Maxwell D Collins, Yukun Zhu, Ting Liu, Thomas S Huang, Hartwig Adam, and Liang-Chieh Chen. Panoptic-deeplab: A simple, strong, and fast baseline for bottom-up panoptic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12475–12485, 2020.

[14] Bruce Christianson. Reverse accumulation and attractive fixed points. *Optimization Methods and Software*, 3(4):311–326, 1994.

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

[16] Caio Corro and Ivan Titov. Differentiable perturb-and-parse: Semi-supervised parsing with a structured variational autoencoder. In *International Conference on Learning Representations*, 2019.

[17] Marco Cuturi, Olivier Teboul, and Jean-Philippe Vert. Differentiable ranking and sorting using optimal transport. 2019.

[18] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[19] Josip Djolonga and Andreas Krause. Differentiable learning of submodular models. *Advances in Neural Information Processing Systems*, 30:1013–1023, 2017.

[20] Justin Domke. Implicit differentiation by perturbation. *Advances in Neural Information Processing Systems*, 23:523–531, 2010.

[21] Justin Domke. Learning graphical model parameters with approximate marginal inference. *IEEE transactions on pattern analysis and machine intelligence*, 35(10):2454–2467, 2013.

[22] Aaron Ferber, Bryan Wilder, Bistra Dilkina, and Milind Tambe. Mipaal: Mixed integer program as a layer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 1504–1511, 2020.

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

[24] Stephen Gould, Richard Hartley, and Dylan Campbell. Deep declarative networks: A new hope. *arXiv preprint arXiv:1909.04866*, 2019.

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

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

[27] Hedda Cohen Indelman and Tamir Hazan. Learning randomly perturbed structured predictors for direct loss minimization. *arXiv preprint arXiv:2007.05724*, 2020.

[28] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

[29] Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7482–7491, 2018.

[30] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

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

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

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

[34] Patrick Knobelreiter, Christian Sormann, Alexander Shekhovtsov, Friedrich Fraundorfer, and Thomas Pock. Belief propagation reloaded: Learning bp-layers for labeling problems. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7900–7909, 2020.

[35] James Kotary, Ferdinando Fioretto, Pascal Van Hentenryck, and Bryan Wilder. End-to-end constrained optimization learning: A survey. *arXiv preprint arXiv:2103.16378*, 2021.

[36] Thorben Kroeger, Jörg H Kappes, Thorsten Beier, Ullrich Koethe, and Fred A Hamprecht. Asymmetric cuts: Joint image labeling and partitioning. In *German Conference on Pattern Recognition*, pages 199–211. Springer, 2014.

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

[38] Ke Li, Kevin Swersky, and Richard Zemel. Efficient feature learning using perturb-and-map. 2013.

[39] Qizhu Li, Anurag Arnab, and Philip HS Torr. Weakly-and semi-supervised panoptic segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 102–118, 2018.

[40] Qizhu Li, Xiaojuan Qi, and Philip HS Torr. Unifying training and inference for panoptic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13320–13328, 2020.

[41] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2G014.

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

[43] David A McAllester, Tamir Hazan, and Joseph Keshet. Direct loss minimization for structured prediction. In *NIPS*, volume 1, page 3. Citeseer, 2010.

[44] Arthur Mensch and Mathieu Blondel. Differentiable dynamic programming for structured prediction and attention. In *International Conference on Machine Learning*, pages 3462–3471. PMLR, 2018.

[45] Shervin Minaee, Yuri Y Boykov, Fatih Porikli, Antonio J Plaza, Nasser Kehtarnavaz, and Demetri Terzopoulos. Image segmentation using deep learning: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[46] Rohit Mohan and Abhinav Valada. Efficientps: Efficient panoptic segmentation. *International Journal of Computer Vision*, pages 1–29, 2021.

[47] George Papandreou and Alan L Yuille. Perturb-and-map random fields: Using discrete optimization to learn and sample from energy models. In *2011 International Conference on Computer Vision*, pages 193–200. IEEE, 2011.

[48] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[49] Max B Paulus, Dami Choi, Daniel Tarlow, Andreas Krause, and Chris J Maddison. Gradient estimation with stochastic softmax tricks. *arXiv preprint arXiv:2006.08063*, 2020.

[50] Hao Peng, Sam Thomson, and Noah A Smith. Backpropagating through structured argmax using a spigot. *arXiv preprint arXiv:1805.04658*, 2018.

[51] Tobias Plötz and Stefan Roth. Neural nearest neighbors networks. *Advances in Neural Information Processing Systems*, 31:1087–1098, 2018.

[52] Marin Vlastelica Pogančić, Anselm Paulus, Vit Musil, Georg Martius, and Michal Rolinek. Differentiation of blackbox combinatorial solvers. In *International Conference on Learning Representations*, 2019.

[53] Lorenzo Porzi, Samuel Rota Bulo, Aleksander Colovic, and Peter Kontschieder. Seamless scene segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8277–8286, 2019.

[54] Siyuan Qiao, Liang-Chieh Chen, and Alan Yuille. Detectors: Detecting objects with recursive feature pyramid and switchable atrous convolution. *arXiv preprint arXiv:2006.02334*, 2020.

[55] Michal Rolínek, Vít Musil, Anselm Paulus, Marin Vlastelica, Claudio Michaelis, and Georg Martius. Optimizing rank-based metrics with blackbox differentiation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7620–7630, 2020.

[56] Michal Rolínek, Paul Swoboda, Dominik Zietlow, Anselm Paulus, Vít Musil, and Georg Martius. Deep graph matching via blackbox differentiation of combinatorial solvers. In *European Conference on Computer Vision*, pages 407–424. Springer, 2020.

[57] Jie Song, Bjoern Andres, Michael J Black, Otmar Hilliges, and Siyu Tang. End-to-end learning for graph decomposition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10093–10102, 2019.

[58] Zhi Tian, Chunhua Shen, and Hao Chen. Conditional convolutions for instance segmentation. *arXiv preprint arXiv:2003.05664*, 2020.

[59] Huiyu Wang, Yukun Zhu, Hartwig Adam, Alan Yuille, and Liang-Chieh Chen. Max-deeplab: End-to-end panoptic segmentation with mask transformers. *arXiv preprint arXiv:2012.00759*, 2020.

[60] Huiyu Wang, Yukun Zhu, Bradley Green, Hartwig Adam, Alan Yuille, and Liang-Chieh Chen. Axial-deeplab: Stand-alone axial-attention for panoptic segmentation. In *European Conference on Computer Vision*, pages 108–126. Springer, 2020.

[61] Po-Wei Wang, Priya Donti, Bryan Wilder, and Zico Kolter. Satnet: Bridging deep learning and logical reasoning using a differentiable satisfiability solver. In *International Conference on Machine Learning*, pages 6545–6554. PMLR, 2019.

[62] Bryan Wilder, Eric Ewing, Bistra Dilkina, and Milind Tambe. End to end learning and optimization on graphs. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[63] Steffen Wolf, Yuyan Li, Constantin Pape, Alberto Bailoni, Anna Kreshuk, and Fred A Hamprecht. The semantic mutex watershed for efficient bottom-up semantic instance segmentation. In *European Conference on Computer Vision*, pages 208–224. Springer, 2020.

[64] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. `https://github.com/facebookresearch/detectron2`, 2019.

[65] Zhirong Wu, Yuanjun Xiong, Stella X Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance discrimination. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3733–3742, 2018.

[66] Yujia Xie, Hanjun Dai, Minshuo Chen, Bo Dai, Tuo Zhao, Hongyuan Zha, Wei Wei, and Tomas Pfister. Differentiable top-k operator with optimal transport. *arXiv preprint arXiv:2002.06504*, 2020.

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

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

[69] Yibo Yang, Hongyang Li, Xia Li, Qijie Zhao, Jianlong Wu, and Zhouchen Lin. Sognet: Scene overlap graph network for panoptic segmentation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 12637–12644, 2020.

[70] Xiaohui Zeng, Renjie Liao, Li Gu, Yuwen Xiong, Sanja Fidler, and Raquel Urtasun. Dmm-net: Differentiable mask-matching network for video object segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3929–3938, 2019.

[71] Shuai Zheng, Sadeep Jayasumana, Bernardino Romera-Paredes, Vibhav Vineet, Zhizhong Su, Dalong Du, Chang Huang, and Philip HS Torr. Conditional random fields as recurrent neural networks. In *Proceedings of the IEEE international conference on computer vision*, pages 1529–1537, 2015.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] Compare contributions in Section 1 with results in Section 4
   (b) Did you describe the limitations of your work? [Yes] Yes, for example inference times and decreased localization performance
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] Yes, see broader impact.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results... [No] Our study is mostly empirical
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] Code will be made publicly available after acceptance
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] Yes, major in main content and rest in appendix.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] We show performance increase on testing data for which ground-truth is not public there is a very less chance to find a 'lucky' seed. Nonetheless, supplementary material contains this comparison.

(d) Did you include the total amount of compute and the type of resources used [Yes]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] For example Cityscapes, COCO datasets, PyTorch, etc.
   (b) Did you mention the license of the assets? [No] All assets e.g. Cityscapes dataset are publicly available for academic purposes.
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects... [N/A]