What's Outside the Intersection? Fine-grained Error Analysis for Semantic Segmentation Beyond IoU *Supplementary Material*

Maximilian Bernhard[1,2], Roberto Amoroso[3], Yannic Kindermann[1]
Lorenzo Baraldi[3], Rita Cucchiara[3], Volker Tresp[1,2], Matthias Schubert[1,2]

[1]LMU Munich, [2]MCML, [3]University of Modena and Reggio Emilia

bernhard@dbs.ifi.lmu.de

Overview

This supplementary document is organized as follows:

A. Visualization of Error Categories

In Figure 1, we visualize ground truth and prediction along with the error types that occur for selected classes on a sample from ADE20K. Most importantly, we observe that the visualized error categories comply with our intuition for their definition, *i.e.*, boundary errors occur close to correct foreground-background transitions, extent errors occur when predicted segment boundaries are severely misplaced, and segment errors occur when entire segments are mispredicted.

B. Qualitative Comparison of Single and Combined Models

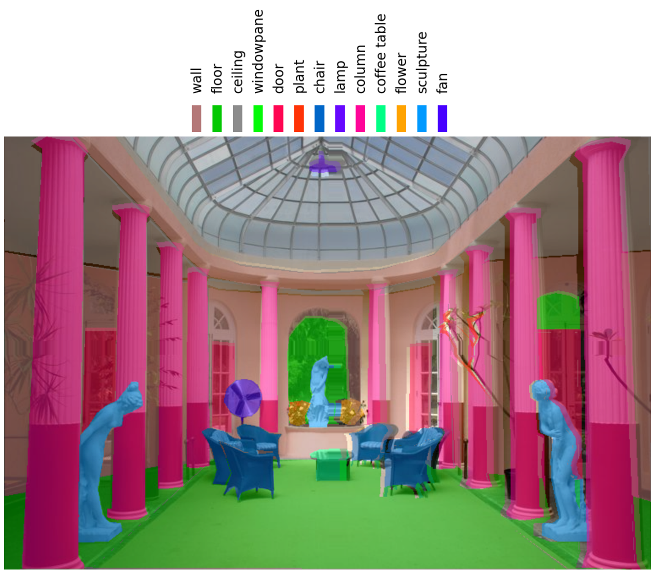
Figure 2 depicts qualitative results for SETR + ViT-L, Mask2Former + R101-D8, and their combination. As shown in the paper, Mask2Former is superior to SETR in precisely delineating segments, whereas SETR has an advantage in the classification of objects and regions. This finding is confirmed by the visualized predictions in the figure. By combining these two models, we aim to leverage their individual strengths while mitigating their weaknesses. Subfigure (d) demonstrates the effectiveness of this approach: both SETR's superior classification performance (*e.g.*, the counter in the image) and Mask2Former's superior ability to accurately segment details (*e.g.*, the lamps, the vase, and the flowers) are preserved in the combined predictions. This gives a better understanding of the $mIoU$ improvements achieved with the combinations in the paper.

C. Comparing Error Distributions across Classes

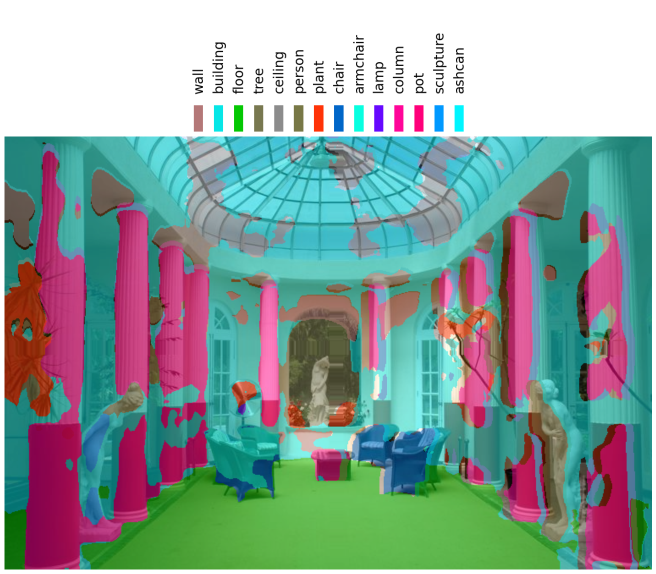
Since our proposed error categorization considers each semantic class separately, one can analyze the error distribution of each class individually. We visualize a subset of the error distributions per class for PSPNet + R50-D8 on COCO-Stuff 164k in Figure 3. Looking at the plot, we can see that there are substantial differences between classes. In particular, the class ”fog” exhibits an error distribution that is fundamentally different from those of other classes, having an extremely high segment error rate, while boundary, and extent errors are exceptionally low. That is, the model often fails to recognize the class ”fog”, but when it does, it is able to segment it remarkably well. In order to investigate this observation further, we visually inspected the COCO-Stuff dataset and found that there is a label ambiguity between ”fog” and ”clouds”, for which we provide examples in Figure 4. Thus, the observed anomaly in the distribution of error types indicates a problem in the annotations, but not on the model side. This example demonstrates once again that our error analysis can provide meaningful insights.

D. Analyzing the Effect of Training Schedules

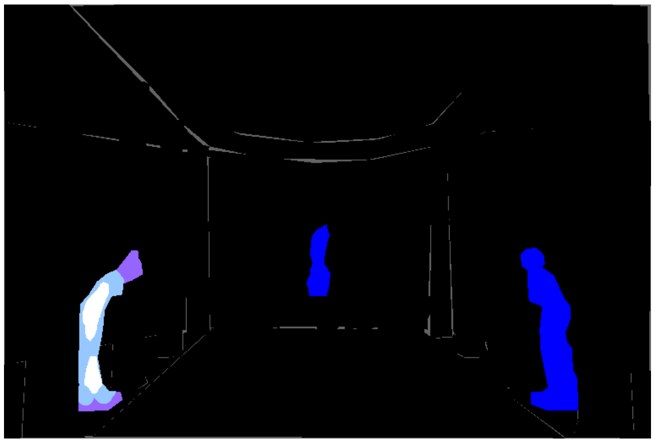
We examine the effect of training schedules on the example of PSPNet on COCO-Stuff 164k in Table 1. The schedules considered are those defined in MMSegmenta-



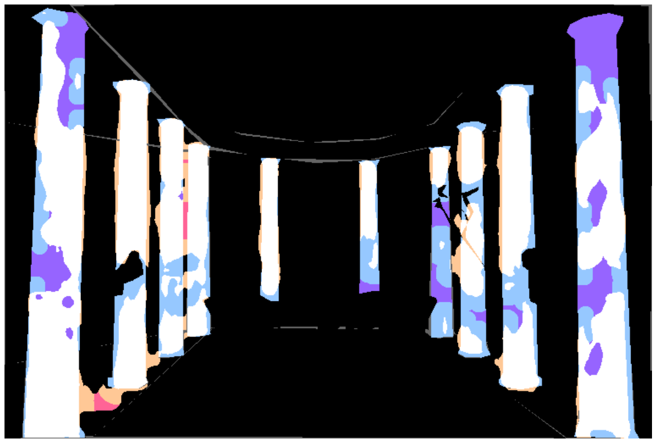
(a) Ground truth



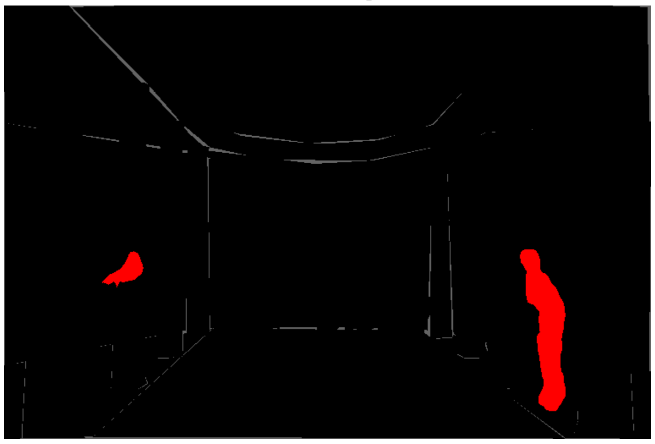
(b) Prediction



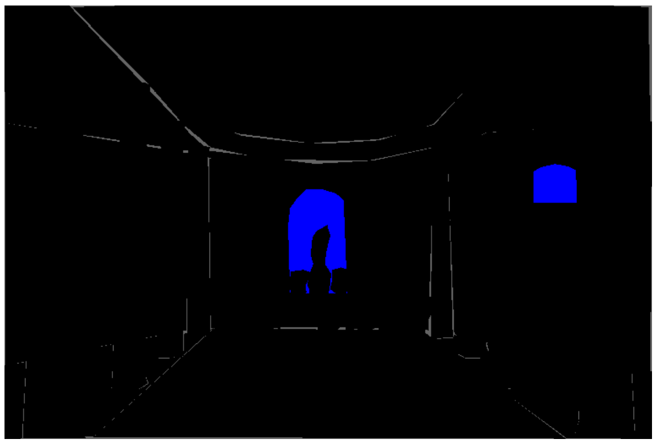
(c) Errors "sculpture"



(d) Errors "column"



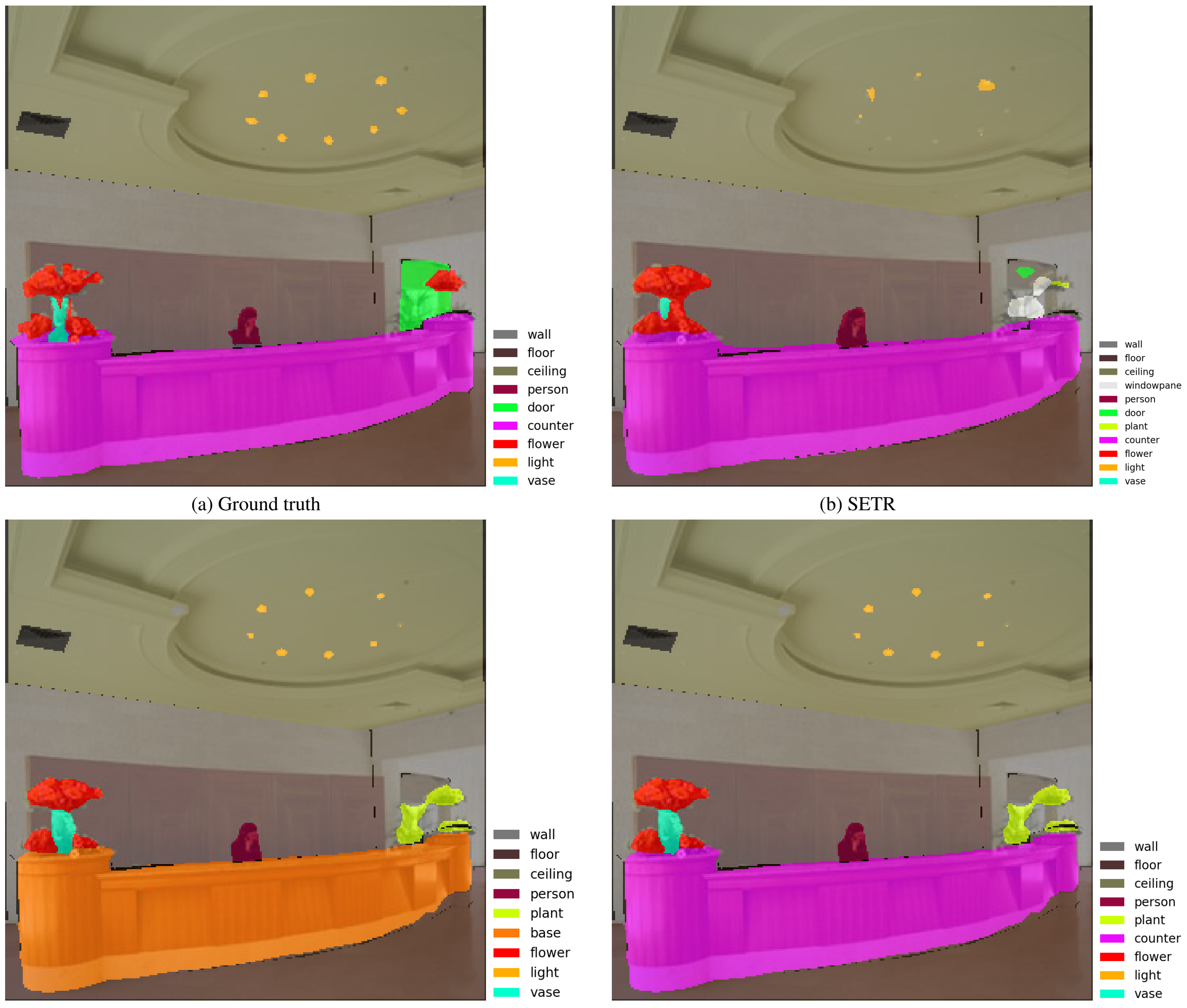
(e) Errors "person"



(f) Errors "windowpane"

▨	ignore	🟧	FP_bnd	🟦	FN_bnd
⬛	TN	🟥	FP_ext	🟪	FN_ext
⬜	TP	🟥	FP_seg	🟦	FN_seg

Figure 1. **Visualization of error types** for a sample from ADE20K with DeepLabV3+.



(a) Ground truth (b) SETR (c) Mask2Former (d) Mask2Former ∘ SETR

Figure 2. **Qualitative Example** from ADE20K for SETR + ViT-L, Mask2Former + R101-D8, and their combination. The combined prediction (d) preserves the fine detail of Mask2Former (lamps, vase, flowers) as well as the superior segment classification of SETR (counter).

		mE$_\star$oU			$\widetilde{\textbf{mE}_\star\textbf{oU}}$		
Schedule	**IoU**	bnd	ext	seg	bnd	ext	seg
80k	38.8	5.9	16.3	39.0	16.8	32.8	39.0
160k	39.6	5.9	16.2	38.2	17.0	32.2	38.2
320k	40.5	6.0	15.6	37.8	16.3	30.9	37.8

Table 1. **Comparing different training schedules** for PSPNet + R50-D8 on COCO-Stuff 164k.

tion[1]. Overall, a longer schedule improves the $mIoU$. Looking at the error rates, we can see that this improvement is mainly caused by fewer segment and extent errors, whereas the number of boundary errors remains relatively stable. This suggests that the low-level cues needed to segment boundaries are learned rather early, while high-level semantic features needed for classification and capturing extents are better learned with more training iterations.

[1]https://github.com/open-mmlab/mmsegmentation/tree/e64548fda0221ad708f5da29dc907e51a644c345/configs/_base_/schedules

E. Analyzing the Effect of Test-time Augmentation

We also investigate the effect of test-time augmentation (TTA) on the distribution of error types. We used the TTA

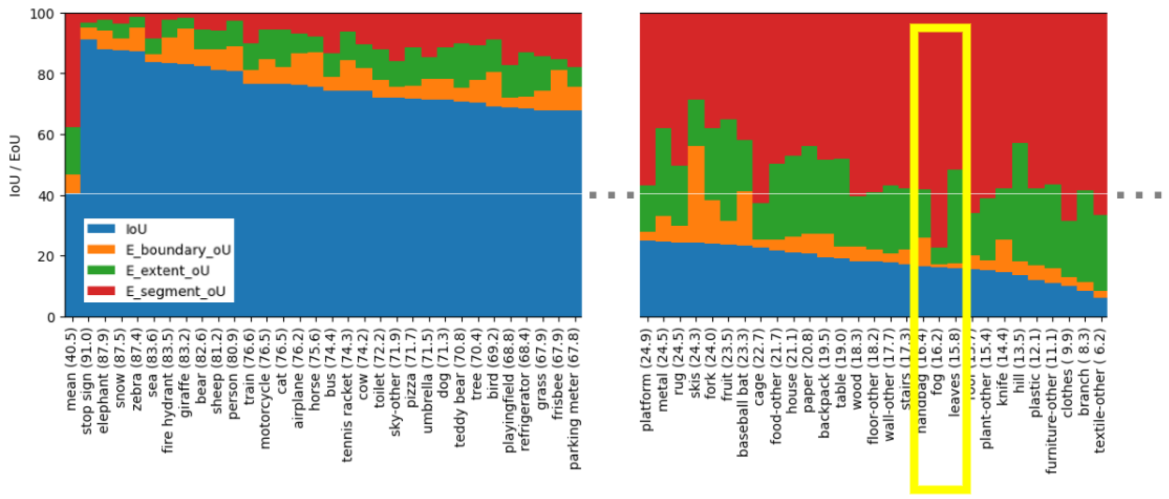
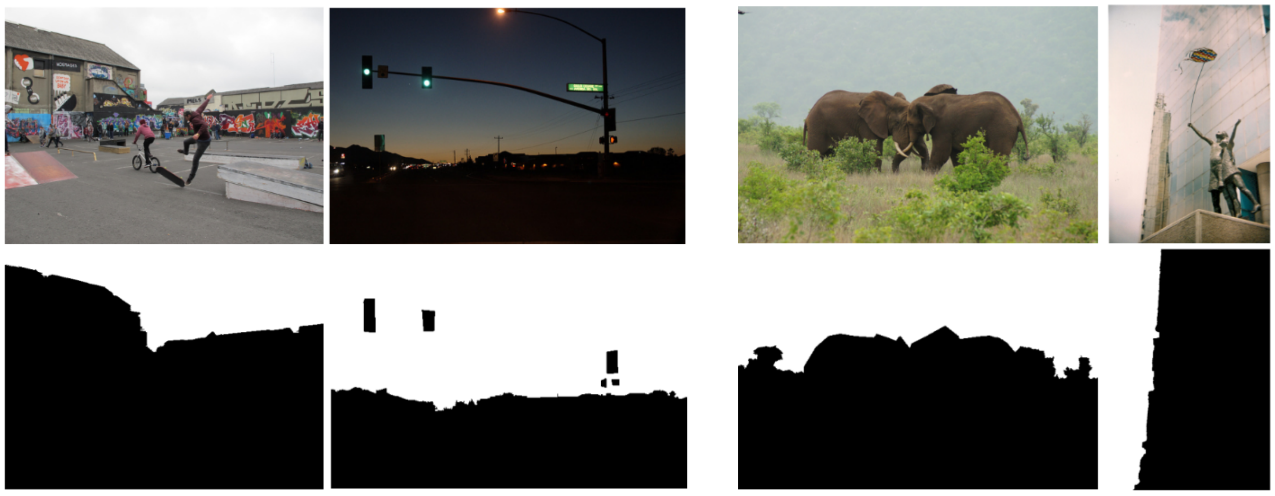


Figure 3. **IoU and EoUs per class on COCO-Stuff 164k** The class ”fog” (highlighted) stands out with a high number of segment errors and low numbers of boundary and extent errors.



(a) Fog (b) Clouds

Figure 4. **Label ambiguity in COCO-Stuff 164k** Two samples for each of the classes ”fog” and ”clouds” together with the binary ground-truth masks for these classes exemplify that there is no clear distinction between the classes in the annotations.

			$\mathbf{mE_{\star}oU}$			$\widetilde{\mathbf{mE_{\star}oU}}$		
Model	**TTA**	**IoU**	*bnd*	*ext*	*seg*	*bnd*	*ext*	*seg*
PSPNet		44.4	9.0	15.9	30.7	19.2	25.2	30.7
PSPNet	✓	45.1	8.0	15.6	31.3	17.2	24.6	31.3
DeepLabV3+		45.5	9.0	15.8	29.8	18.8	24.1	29.8
DeepLabV3+	✓	46.0	7.9	14.9	31.2	17.3	23.9	31.2

Table 2. **Analyzing the effects of test-time augmentation (TTA)** on ADE20K. All models use R101-D8 as backbone.

pipeline as specified in MMSegmentation for ADE20K[2]. This includes resizing the input images to multiple scales as well as flipping. The results are provided in Table 2. Our error analysis shows that the improvements achieved with TTA come from a reduction in boundary and extent errors. We argue that this is the case because class boundaries are usually the regions with the most uncertain predictions and averaging the predicted scores over multiple views of the input reduces this uncertainty.

Surprisingly, we also observe an increase in segment errors when using TTA. In fact, a closer inspection of the error statistics reveals that only the false negative segment errors $FN_{seg}oU$ increase (from 17.4% to 19.4% and from 15.5% to 18.1% for PSPNet and DeepLabV3+, respec-

[2] https://github.com/open-mmlab/mmsegmentation/blob/e64548fda0221ad708f5da29dc907e51a644c345/configs/_base_/datasets/ade20k.py#L27

tively), while the false positive segment errors $FP_{seg}oU$ decrease (from 13.3% to 11.8% and from 14.2% to 13.1% for PSPNet and DeepLabV3+, respectively). This behavior can be explained by the averaging of the TTA predictions, making the final predictions more consistent and less fragmented. However, more fragmented predictions increase the chance of true positive predictions for at least small parts of the ground-truth segments, which is why the predictions without TTA yield substantially lower values for $FN_{seg}oU$.

F. Boundary Errors vs. Boundary IoU

Since the terminology for the Boundary IoU and our boundary errors are quite similar, we provide a clear distinction between the two concepts here. Boundary IoU measures the IoU on pixels that are close to the boundary but inside the ground-truth foreground ($G_d \cap G$ in the Boundary IoU paper, where G_d is the set of pixels in the boundary region of the ground-truth mask) as well as on pixels that are close to the boundary but inside the predicted foreground ($P_d \cap P$). On the other hand, boundary errors in our method are close to both the boundary of the true positives and the boundary of the true negatives.

Hence, the two main differences are that, for boundary errors, the boundaries of true positives and true negatives are considered instead of the boundaries of the predicted and the ground-truth mask, and that only the intersection of these boundary regions is relevant. Considering the boundaries of true positives and true negatives implements the intuitive constraint that boundary errors can only occur in the vicinity of true predictions (*i.e.*, when a transition from class foreground to the background has been recognized). In other words, Boundary IoU aims to measure the overall segmentation quality, whereas the boundary error rate is intended to be an indicator of the segmentation quality along boundaries that have been correctly recognized (at least roughly). Moreover, our definition of boundary errors is not solely based on the proximity to the boundaries, but also includes modifications to counteract unwanted effects (see Equations 2 and 3 in the paper). These increase the maximum relevant distance of pixels to boundaries to $2d$ (compared to d in Boundary IoU).

G. Proof of the Disjointness of the Error Categories

While it is easy to implement an error assignment such that each erroneous pixel receives exactly one error category, seeing the disjointness of the three categories defined in Section 3 of the paper is not as straightforward. Therefore, we provide proof of this claim.

- $E_{bnd} \cap E_{ext} = \emptyset$: Trivial by definition.
- $E_{ext} \cap E_{seg} = \emptyset$: Also trivial by definition.
- $E_{bnd} \cap E_{seg} = \emptyset$: Suppose there is a pixel location $x \in E_{bnd} \cap E_{seg}$. Without loss of generality, also assume that $x \in FP$. From Equation 3 in the paper, we can deduce that there is an $x' \in \mathcal{S}(FP''_{bnd})_x$ such that $\mathcal{N}_1(x') \cap TP \neq \emptyset$. Let x'' be a TP pixel in $\mathcal{N}_1(x')$. Then, we know that $x'' \in \mathcal{S}(P)_x$ as x'' is a direct neighbor of $\mathcal{S}(FP''_{bnd})_x$. With $x'' \in \mathcal{S}(P)_x$, we obtain $\mathcal{S}(P)_x \cap TP \neq \emptyset$, contradicting $x \in E_{seg}$ and, therefore, concluding the proof.

Furthermore, every erroneous pixel belonging to at least one of the proposed error categories follows immediately from their definitions in the paper.

H. Reproducibility

For most of the models used in our analysis, we used both the code and the checkpoints from MMSegmentation v1.0.0. Accordingly, the $mIoU$s determined in our analysis are identical to those reported in MMSegmentation. For the models that are not available in MMSegmentation, we used the inference code and checkpoints from their official repositories to generate predictions. Interestingly, in these cases, we occasionally observed deviations from the $mIoU$ scores reported in the original papers (*e.g.*, 58.0% instead of 58.3% for OneFormer + DiNAT-L).

I. Limitations

Although we have extensively demonstrated the practical use of our method, there are certain limitations. Without additional information, the assignment of a pixel to an error category is inevitably based on heuristics. Our definition of segments is based on contiguity and generally represents object instances reasonably well, but not in all cases. For example, a single ground-truth object may be represented by two or more contiguous segments due to occlusion. This may lead to segment errors that would ideally be categorized as extent errors. Nevertheless, we decided not to incorporate instance annotations in order to minimize the requirements for our error analysis and make it applicable to any segmentation dataset and model.

Furthermore, state-of-the-art semantic segmentation models are hardly explainable and hardly interpretable deep neural networks. Thus, retrieving reliable and rigorous explanations for the true root causes of segmentation errors is (currently) not possible, even more in a model-agnostic way. Therefore, our error categorization is intended to convey an intuition about the strengths and weaknesses of models, but it cannot give rigorous explanations as to why certain regions are not correctly segmented.

J. Tool

Since our error analysis is intended to provide insight into semantic segmentation models in an intuitive way, ease of use and a clear presentation of the results are crucial. Therefore, our released tool allows to run the analysis with

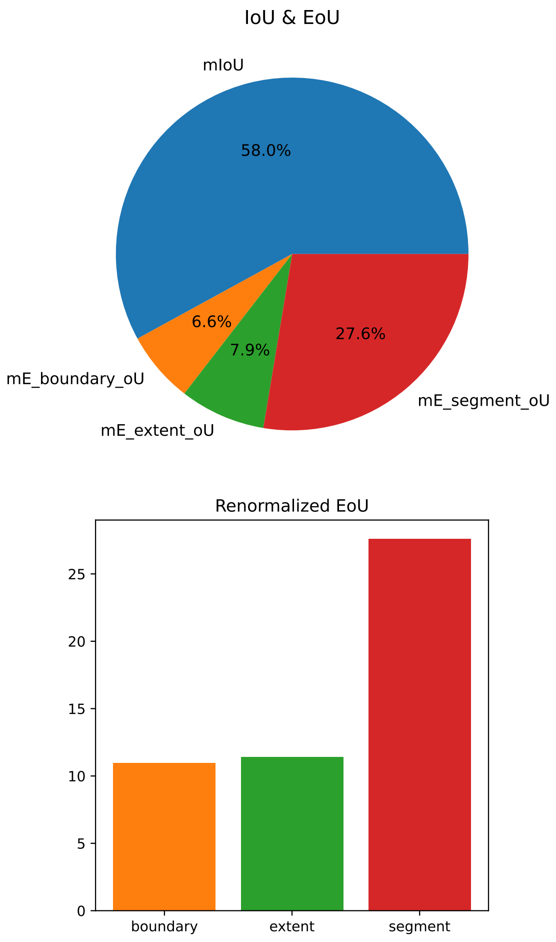


Figure 5. **Example summary diagrams created with our tool** for OneFormer + DiNAT-L on ADE20K val.

just a few lines of code. Provided with paths to directories containing ground-truth and predicted segmentations, the boundary parameter d and some meta-information such as an ignore index and the classes in the dataset, the tool performs the analysis and produces a detailed summary. The result comprises not only statistics for our error categories and $mIoU$, but also other metrics such as *Boundary IoU* and *Trimap IoU*. Thus, the tool fits perfectly into the landscape of existing evaluation strategies for semantic segmentation. Furthermore, it features utilities for plotting to convey insights more easily (see Figures 5, 1, and 3).

Regarding the runtime of our analysis, we observed quite large differences between datasets. For example, the analysis on the 1,449 PascalVOC validation images takes about 5 min, while it takes about 35 min on the 2,000 ADE20K validation images. Among other factors, the runtime depends on the predictions, *i.e.*, evaluating more fragmented predictions takes longer as there are more contiguous segments to consider. Since our method is intended for the final model evaluation and comparison (*i.e.*, the evaluation is run only once), the observed runtimes are clearly practicable.