A. Details on Dataset Creation and the Injection of Label Errors

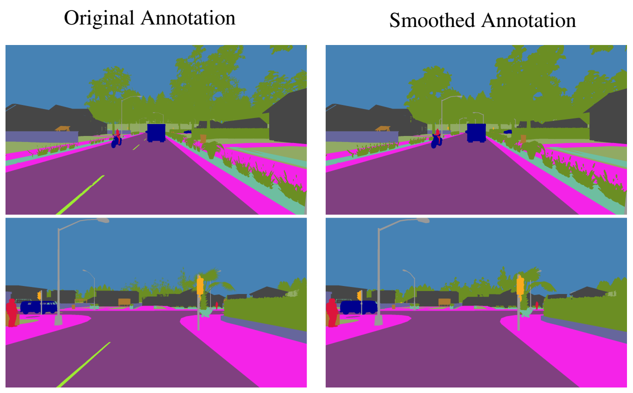

Figure 4. Two annotations, before and after we applied smoothing. Note that, we do not smooth every class. Smoothing is applied onto the classes of pedestrians, poles, vegetation and vehicles. Additionally, we remove road lines from the annotation.

CARLA. We create the dataset by randomly selecting a spawn point for the ego vehicle and recording in the first step a segmentation mask of the empty scene. We then randomly spawn objects around the ego vehicle and record an input image as well as another segmentation mask of the scene, while all objects including the ego vehicle are static. The latter pair of image and ground truth serve as input image and unperturbed / clean reference ground truth for evaluation. Thereafter, we copy the connected component of the clean annotation mask into the first one and randomly select components to omit in order to induce label errors. When recording the same scene multiple times in CARLA, the ground truth typically varies at boundaries of connected components due to rendering effects. Our procedure of ground truth generation avoids this effect.

The CARLA simulator's semantic segmentation sensor provides very precised annotated segmentation masks. To emulate human annotated masks and to make our synthetic dataset more realistic, we smooth these masks in order to reduce the degree of detailedness. We proceed class by class and consider the classes of Pedestrians, Poles, Vegetation, and Vehicles (in arbitrary order). More precisely, our smoothing process proceeds as follows: For a given street scene, we first create binary maps for each class where every pixel of the class under consideration for smoothing obtains some value $i \in \mathbb{N}$ and we associate 0 to every other pixel. The value i enables us to control how much an object is smoothed. For our smoothing we set $i = 10$. We then apply a Gaussian smoothing kernel onto the binary masks and use these smoothed binary masks as index map to overwrite the values of the corresponding pixels in the original segmentation mask with the currently smoothed class value. To avoid that we override values of pixels that belong to a component that is in fact in front (in terms of visual depth) of a component of the smoothed class, we use depth information provided by CARLA to determine during smoothing for each pixel which class is in front. After we finished this smoothing process we fill the windows of vehicles in the scene as CARLA ignores them in the labeling process. Lastly, we remove the road lines and overwrite them with the class road. The result of this process can be seen in fig. 4.

Cityscapes. Due to the availability of polygonal annotations and the fact that the labeling style is hierarchical (objects are mostly labeled on top of the background components), we can drop polygons according to the procedure mentioned in eq. (3). In this way, we obtain a perturbed ground truth by dropping polygons and then generating the pixel-wise segmentation masks out of the JSON files. Although the labeling style is hierarchical, it is possible that the removal of a polygon in the annotation leads to an unlabeled region in the resulting perturbed mask. As it turned out in section 5, this is a realistic error and therefore we do not ignore unlabeled components in our evaluation but consider them as a class in the dataset.

B. A Class-wise Breakdown of Results for Induced Label Errors.

In this section, we present the class-wise results for Cityscapes in experiment 1 with pertubation rate $\hat{p} = 0.5$ using the sota weights for Nvidia's multiscale attention net (appendix B) and for the Deeplab net (appendix B). Ignoring the class Bus, for the attention net we obtained the best F1 score for the class Person with 74.88% which also contains the most label errors. The lowest F1 scores occur for the classes Traffic Light and Traffic Sign. From a visual inspection of predicted masks, it seems that the DNN relies more on the geometry of these objects, while paying less attention to their textures. Similarly, for the Deeplab architecture the class Person achieves the second highest F1 score with 76.64%. Here, the best class is Rider with 80.39% which has a F1 score of only 58.99% with Nvidia's net. The lowest scores are again given for class Traffic Light and class Traffic Sign.

C. Class-wise Breakdown of Found Label Errors in Frequently used Datasets

This section is an extension to the discussion of section 5.2. Here we provide additional details for each dataset we studied and present results for selected classes. In appendix D we present for each dataset a collection of label errors found in the respective datasets.

Cityscapes. In table 8 we have given the class-wise results for Cityscapes and fig. 7 presents example errors we found.

	(m)IoU	TP	FN	FP	Prec	Rec	F1
Bicycle	84.26	81	44	26	75.70	64.80	69.83
Bus	95.48	1	0	1	50.00	100.00	66.67
Car	96.86	100	61	56	64.10	62.11	63.09
Motorcycle	78.24	8	6	0	100.00	57.14	72.73
Person	87.91	161	76	32	83.42	67.93	74.88
Rider	75.47	41	46	11	78.85	47.13	58.99
Traffic Light	79.88	38	1	63	37.62	97.44	54.29
Traffic Sign	87.64	55	17	179	23.50	76.39	35.05
Truck	92.64	2	8	3	35.59	20.00	26.67
Overall	86.82	487	259	371	56.76	65.28	60.72

Table 6. Class-wise results for Cityscapes using the Nvidia's Multi-scale Attention net.

	(m)IoU	TP	FN	FP	Prec	Rec	F1
Bicycle	79.00	76	49	24	76.00	60.80	67.56
Bus	94.00	1	0	0	100.00	100.00	100.00
Car	96.50	102	59	29	77.86	63.35	69.86
Motorcycle	73.80	7	7	1	87.50	50.00	63.64
Person	88.20	159	78	19	89.33	67.09	76.63
Rider	75.40	43	44	2	95.56	49.43	80.39
Traffic Light	79.00	33	6	44	62.86	84.62	56.90
Traffic Sign	82.80	56	16	206	21.37	77.78	33.53
Truck	78.80	4	6	0	100.00	40.00	57.14
Overall	81.40	481	265	325	59.68	64.48	61.98

Table 7. Class-wise results for Cityscapes using the Deeplab net.

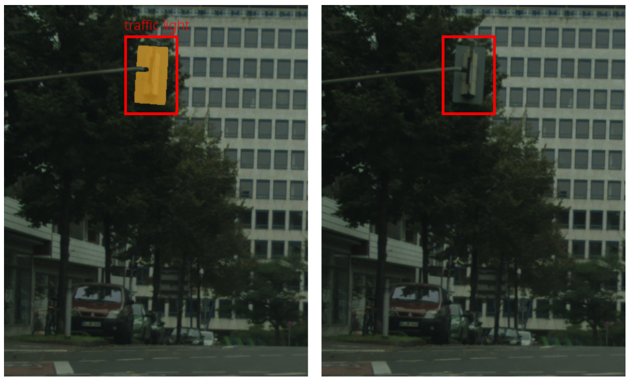

Figure 5. The connected component shown here is originally labeled as void. Our method predicts a label error here which is likely true. However, since we cannot confirm this without any doubt we validated it as false positive.

In the training set we have found 106 label errors by reviewing 200 predictions. For the classes of person, rider/bicycle and vehicles altogether we found 90 true positives, 60 false positive, obtaining a precision of 60%. For traffic lights and road signs we achieve a precision of 50% by finding 25 label errors in 50 predictions. Apart from the class bus which only consists of one prediction, we observe the highest precision for the class rider with 88.89% while, at the same time the IoU is the lowest of all predicted classes with 75.47%. This indicates that our approach does not necessarily depend on a high IoU. It is also possible that the computed IoU for this class is dragged down by label errors in this class. The most errors we found for this class were caused by labeling a rider as a person.

We observed the same for the validation set. Combining the results of the classes of person, rider/bicycle and vehicles we have a precision of 55.33% and of 46.00% for classes of traffic lights and road signs. The result are slightly worse, which is to be expected as the validation set only consists of 500 images while the training set includes almost 3000 images.

Due to our conservative validation approach, for the class traffic light we validated several predictions as false positive even though they could be viewed as true positives; see fig. 5 for examples.

Validating them as true positive would result in additional 15 true positives and a precision of 93% for the class of traffic lights and of 65% in total. Also, in fig. 6 we present a case that occurred several times and is debatable whether it is a false or true positives. However, following the official Cityscapes labeling policy and to avoid any confusion, we consider them to be false positives.

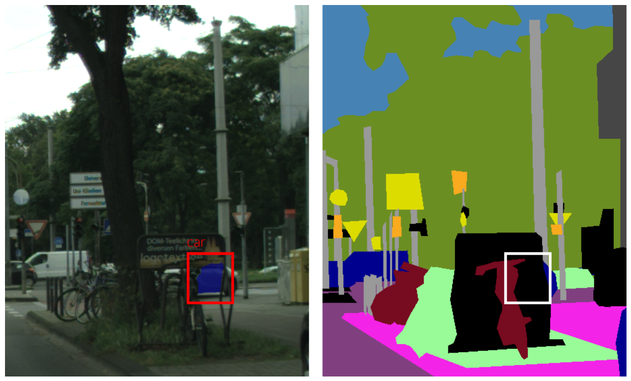

Figure 6. The DNN has correctly found a car component here. However, it is not a foreground object and therefore labeled as void in the ground truth. According to the labeling policy this is a false positive.

	Training set				Validation set		
Class	(m)IoU	TP	FP	Prec	TP	FP	Prec
Bicycle	84.26	16	8	66.67	16	8	66.67
Bus	95.48	1	0	100.00	2	3	40.00
Car	96.86	13	36	26.53	30	32	48.39
Motorcycle	78.24	2	1	66.67	0	0	NaN
Person	87.91	34	17	66.67	35	13	72.91
Rider	75.47	16	2	88.89	7	2	77.78
Traffic sign	79.88	10	18	35.71	12	17	41.38
Traffic light	87.64	13	9	59.09	13	8	61.90
Truck	92.64	1	3	25.00	0	2	00.00
Overall	86.82	106	94	53.00	115	85	57.50

Table 8. Classwise results for the train (left) and validation (right) sets of Cityscapes.

PascalVOC. For PascalVOC, COCO-Stuff, and ADE20K we presented a selection of the most informative classes

PascalVOC					COCO-Stuff					ADE20K				
Class	(m)IoU	TP	FP	Prec	Class	(m)IoU	TP	FP	Prec	Class	(m)IoU	TP	FP	Prec
Bicycle	78.78	0	5	00.00	Building-other	30.81	16	7	69.57	Building	83.89	0	17	00.00
Boat	66.76	4	3	57.14	Ceiling-other	93.76	5	2	71.42	Cabinet	65.40	4	2	66.67
Bottle	81.18	5	6	45.45	Desk-stuff	66.83	4	0	100.00	Chair	57.72	3	3	50.00
Car	83.64	8	8	50.00	Grass	90.81	11	6	64.71	Coffee	64.15	6	0	100.00
Cat	89.21	1	10	09.09	Pavement	26.77	3	3	50.00	Floor	80.32	28	5	84.85
Chair	51.97	17	11	60.71	Playingfield	44.87	9	0	100.00	Grass	59.71	1	5	16.67
Dining table	51.47	10	6	62.50	River	85.00	1	1	50.00	House	25.05	0	3	00.00
Dog	85.01	0	14	00.00	Road	90.70	2	2	50.00	Painting	73.64	1	4	20.00
Person	85.79	38	7	84.44	Sea	96.65	3	2	60.00	Sea	47.57	5	1	83.33
Potted plant	68.02	0	4	00.00	Sky-other	63.13	17	2	89.47	Ship	4.58	2	3	40.00
Sheep	79.61	2	3	40.00	Snow	97.21	6	2	75.00	Sky	92.79	6	3	66.67
Sofa	52.02	6	9	40.00	Table	00.00	0	6	00.00	Table	59.24	4	1	80.00
Train	82.94	1	5	16.67	Tree	73.33	31	9	77.50	Tree	73.58	3	1	75.00
Tv monitor	68.67	0	5	00.00	Wall-concrete	58.59	9	11	45.00	Wall	75.24	7	12	36.84
Overall	78.03	95	105	47.50	Overall	28.20	134	66	67.00	Overall	43.12	110	90	55.00

Table 9. Classwise results for the validation set of PascalVOC, COCO-Stuff, and ADE20K.

in table 9. Examples errors are shown in fig. 8. At first glance, one might expect that classes with low IoU are difficult for our label error detection. PascalVOC provides several counter example for this. While for the class Person we observe high IoU scores together with strong label error detection performance, considering the overall picture there is a clear anti-correlation between label error detection performance and IoU for PascalVOC. Even with low IoU scores, our method is still able to find label errors. For example, the DNN exhibits a comparatively low IoU score of 51.97% for the chair class but still our method found 17 label errors with a precision of 61%. We observe the same for the class Dining table in which we found 10 label errors with a precision of 63% and an IoU of 51.47%. Calculating the Pearson correlation between the IoU and the precision scores we obtain a low negative value of -0.27. Altogether, the findings indicate that the low mIoU might be partially caused by a poor label quality of the dataset.

COCO-Stuff. Selected class results are given in table 9 and some errors are presented in fig. 9. The best results are achieved for the classes of Building-other, Grass, Playingfield, Sky-other and Tree with precision scores significantly above 50% and a representative amount of predictions. Overall, for this dataset we achieved the highest precision of 67% in 200 prediction across 36 predicted classes. For this dataset we also can observe the IoU and precision scores also seem to be barley correlated. This observation is supported by a low correlation score of 0.36.

ADE20K. In this dataset we found 104 label errors in 200 predictions across 58 classes. Table 9 shows results for selected classes and fig. 10 some example errors. As we already mentioned the class definitions of this dataset are in part not sufficiently distinct. In addition, since we were not able to find class descriptions for this dataset, we inferred from the ground truth annotation which objects the classes represent. This led to a significant amount of false positives since we were oftentimes unable to assign TP with appropriate confidence. This applies in particular to the class “building”. In ADE20K, there also exists a class named house which seems to accommodate buildings for human habitation. For the DNN it is difficult to distinguish these two classes, hence it predicts most of the buildings/houses as buildings This leads to a high IoU for the class building, a low IoU of 25.05 for the class house, and a precision of 0% for the building class. However, despite the mentioned issues, the results for ADE20K are comparable to those obtained for the other datasets. Also for this dataset we have a low correlation score of 0.19.

D. Additional Examples of label Errors in Frequently used Datasets

Below we present collages of examplary label errors we found in the datasets discussed in the previous section. Each collage contains 16 pairs of images where each pair represent one label error. The right image represents a section of the ground truth segmentation mask and the left image displays a missed or flipped component. Figures 7 to 10 are collections of label errors in Cityscapes, PascalVOC, COCO-Stuff and ADE20K, respectively.

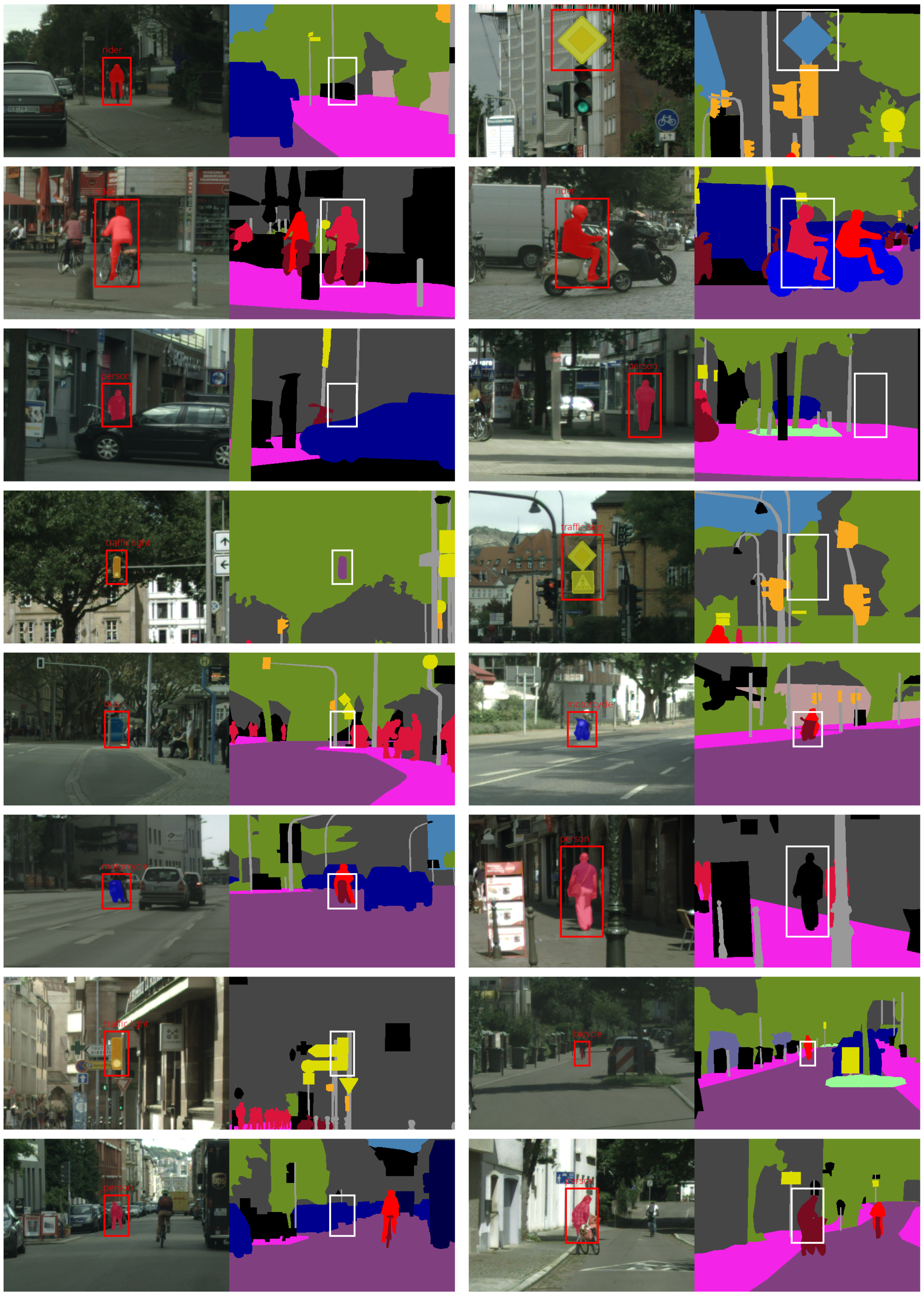

Figure 7. A collection of label errors present in Cityscapes. Left: prediction of our label error detection method; right: “ground truth” annotation. Our method is able to find both, overlooked and flipped labels.

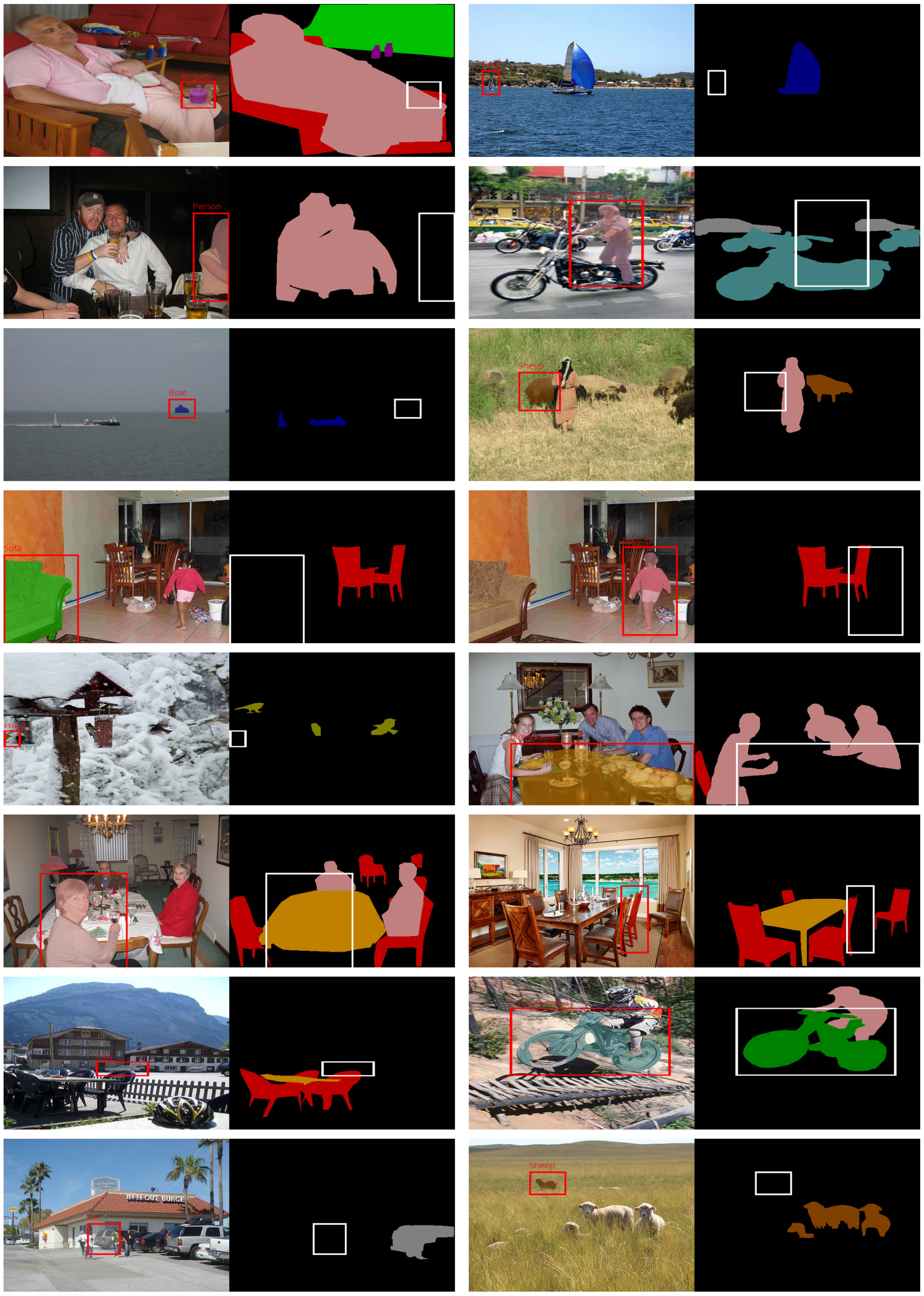

Figure 8. A collection of label errors present in PascalVOC. The visualization scheme follows the one of fig. 7.

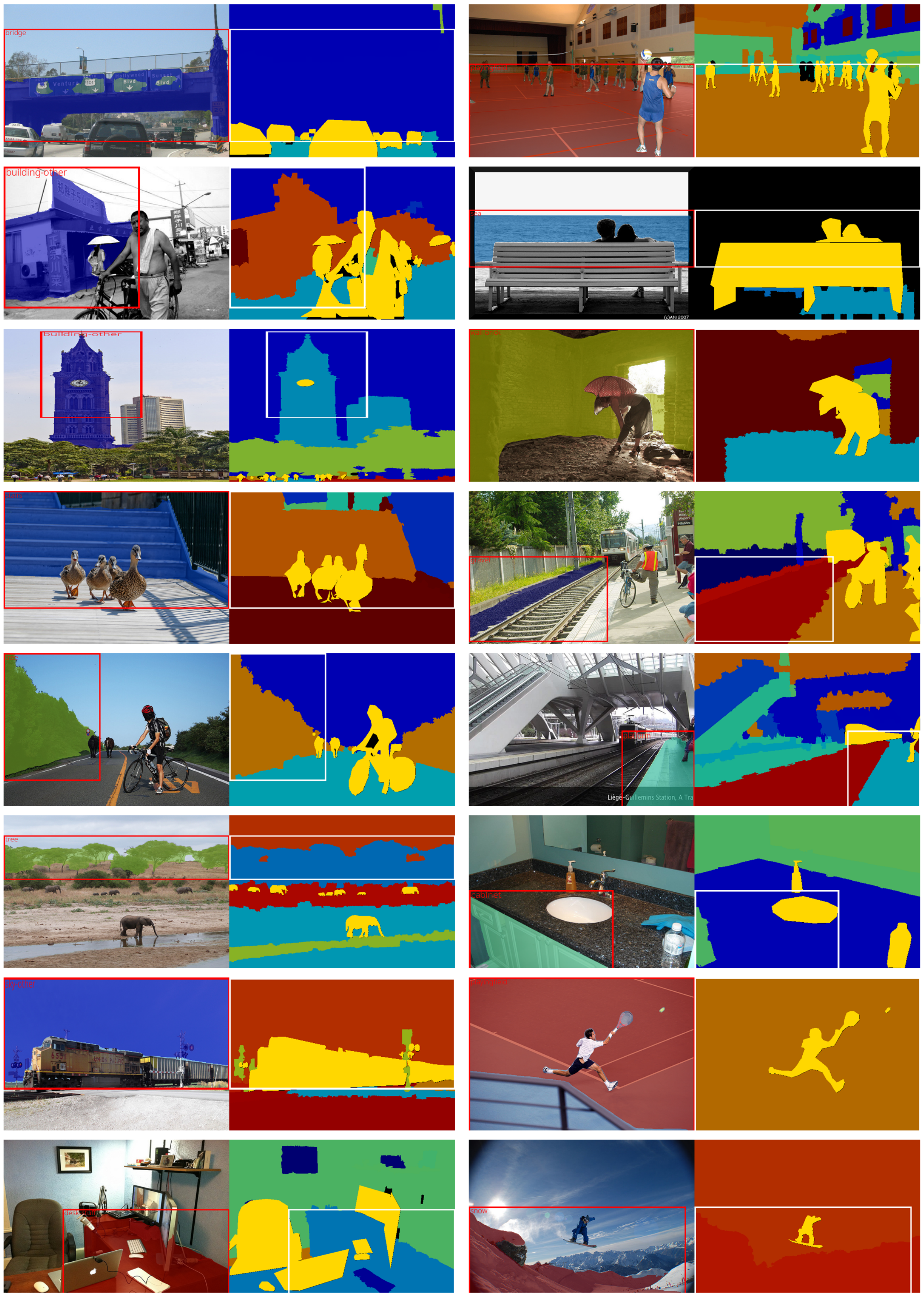

Figure 9. A collection of label errors present in COCO-Stuff. The visualization scheme follows the one of fig. 7.

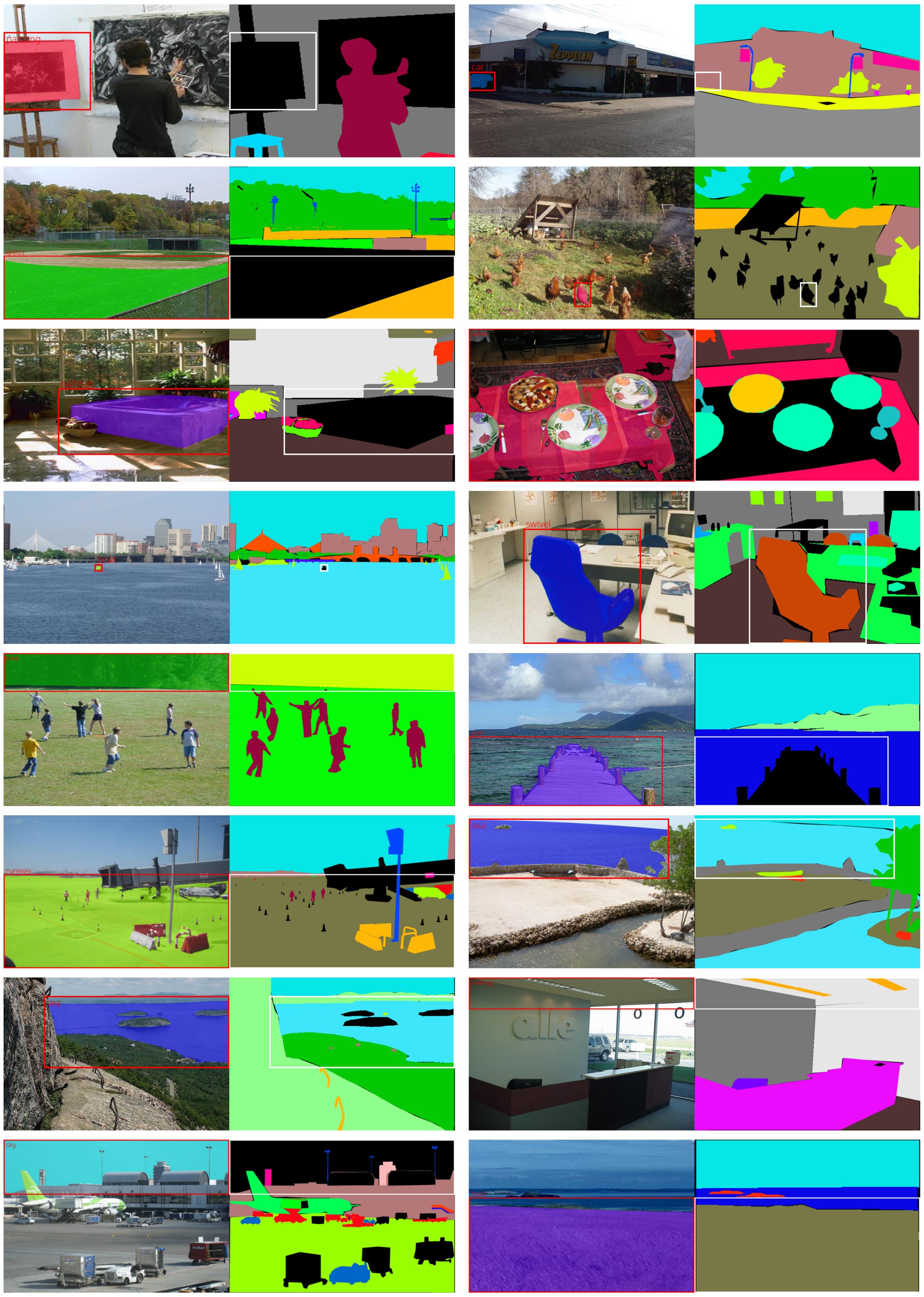

Figure 10. A collection of label errors present in ADE20K. The visualization scheme follows the one of fig. 7.