Placing Objects in Context via Inpainting for Out-of-distribution Segmentation

Pau de Jorge[†,‡] **Riccardo Volpi**[†] **Puneet K. Dokania**[‡] **Philip H.S. Torr**[‡] **Grégory Rogez**[†]
[†]NAVER LABS Europe, [‡]Oxford University
https://github.com/naver/poc

arXiv:2402.16392v1 [cs.CV] 26 Feb 2024

When deploying a semantic segmentation model into the real world, it will inevitably be confronted with semantic classes unseen during training. Thus, to safely deploy such systems, it is crucial to accurately evaluate and improve their *anomaly segmentation* capabilities. However, acquiring and labelling semantic segmentation data is expensive and unanticipated conditions are long-tail and potentially hazardous. Indeed, existing anomaly segmentation datasets capture a limited number of anomalies, lack realism or have strong domain shifts. In this paper, we propose the Placing Objects in Context (POC) pipeline to realistically add *any* object into *any* image via diffusion models. POC can be used to easily extend any dataset with an arbitrary number of objects. In our experiments, we present different anomaly segmentation datasets based on POC-generated data and show that POC can improve the performance of recent state-of-the-art anomaly fine-tuning methods in several standardized benchmarks. POC is also effective to learn new classes. For example, we use it to edit Cityscapes samples by adding a subset of Pascal classes and show that models trained on such data achieve comparable performance to the Pascal-trained baseline. This corroborates the low sim-to-real gap of models trained on POC-generated images.

1. Introduction

When autonomous agents such as robots or self-driving cars are deployed, they are exposed to the open nature of the real world. Inevitably, they will need to process visual conditions that were not anticipated at training. Among other factors, the presence of unseen objects in the scene poses a particularly dangerous safety hazard. For example, consider an unknown wild animal crossing the street and processed by the model as "road". To tackle this issue, it is important to distinguish out-of-distribution (OOD) categories—*i.e.*, novel objects unseen during training—from the in-distribution (ID) ones. In the context of semantic image segmentation, this task is often referred to as *anomaly segmentation*.

Although several methods have been proposed that allow segmenting anomalies [2,13,17,26,31,35], accurately evaluating the performance of such methods is a challenge in itself. Given a model trained on a particular dataset, the aim is to test its ability to process OOD categories in conditions that resemble the training domain. For example, to test a model trained for semantic segmentation of urban scenes, the ideal test set would be constituted by images showing OOD categories within an urban environment similar to the training one. Yet, the distribution of OOD objects is long-tail and it is inefficient, or even hazardous, to acquire and label images with arbitrary categories.

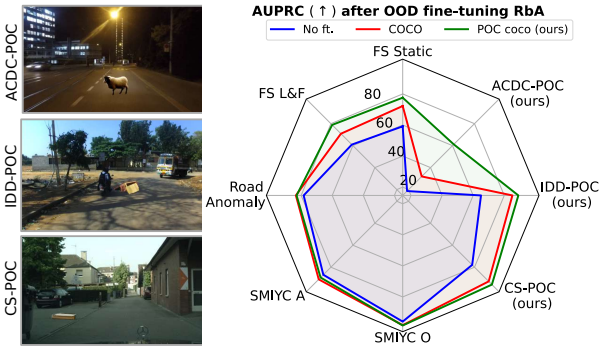

Figure 1: **Left:** Images from our POC-generated datasets. From top to bottom, inserted anomalies are "sheep", "dumped furniture" and "carton box". **Right:** AUPRC on different anomaly segmentation datasets. We evaluate RbA [42] prior to fine-tuning, and after fine-tuning with COCO objects or POC-generated images. Fine-tuning with POC improves results on several benchmarks.

Previous approaches to generate anomaly segmentation datasets can be grouped into three families:

Corresponding authors: pdejorar@gmail.com, riccardo.volpi@naverlabs.com
©2024 NAVER LABS Europe. All rights reserved

Figure 2: **Samples from previous OOD datasets** to visually support Tab. 1: FS Static has unrealistic OOD objects while RoadAnomaly and SMIYC datasets have strong domain shifts from Cityscapes. FS L&F (which manually inserts OOD objects) and StreetHazards (full simulation) have large set-up costs.

Dataset	No domain shift	OOD realism	Dynamic	Low cost
FS Static [3]	✓	✘	(✓)	(✓)
FS L&F [45]	(✓)	✓	✘	✘
SMIYC O. [5]	✘	✓	✘	✘
SMIYC A. [5]	✘	✓	✘	✘
RoadAnom. [34]	✘	✓	✘	✘
StreetHaz. [23]	✘	(✓)	(✓)	✘
POC (ours)	✓	(✓)	✓	✓

Table 1: **Comparison of anomaly test sets.** We qualitatively compare datasets on four main axes. We score them into good (✓), medium ((✓)) and bad (✘). Further discussion in Sec. 2.

Stitching and blending OOD objects from other sources into images from the original dataset [3]; *Collecting images* from driving scenes and annotating OOD objects [5, 34, 45]; *Full simulation* of urban scenes with anomalies [23]. *Stitching and blending* is relatively inexpensive if OOD objects are segmented elsewhere, but it often leads to unrealistic insertions (*e.g.*, object's size and contrast). *Collected images* contain real anomalies, but are expensive to acquire and have a significant distribution shift with respect to the original dataset (it is not easy to find or replicate images following the original setup). *Full simulation* allows for perfectly blended objects but bears a high set up cost and results in a severe drift from the train distribution. See Fig. 2 for illustrative examples.

Ideally, methods to generate anomaly segmentation datasets should satisfy four main desiderata: *i)* Minimal domain shift with respect to the training set—since large domain shifts may lead to underestimating anomaly segmentation capabilities; *ii)* Generating realistic images; *iii)* Allowing for a dynamic generation of new images with arbitrary OOD objects; *iv)* Bearing low set-up costs.

This motivates us to introduce the **Placing Objects in Context** pipeline (POC), which enables practitioners to generate anomaly segmentation test sets by realistically inserting *any* (OOD) object into *any* image on the fly. See a comparison along our desiderata between approaches followed to generate previous benchmarks and the proposed POC in Tab. 1.

In order to insert new objects realistically, we control the location of the added object and only apply local changes to preserve the overall scene semantics. We resort on open-vocabulary segmentation [18] to select valid regions where to place the object, *e.g.*, "the road". We then feed the selected region to an inpainting model [48] with a conditioning prompt, *e.g.*, "a cat". After inpainting, we apply again the segmentation model to the modified area to automatically annotate the added object and detect generation failures, *i.e.*, when the object was not properly generated.

In our experiments, we show that fine-tuning on POC-generated data can significantly improve the performance of state-of-the-art anomaly segmentation methods—outperforming models fine-tuned via the standard practice of *stitching* COCO objects. We also present three POC-generated *evaluation* sets based on Cityscapes and two other self-driving datasets, and benchmark different anomaly segmentation methods on them (see Fig. 1 for a first glimpse of results).

Finally, since POC can add arbitrary objects, we show it can be used to learn new classes. For instance, augmenting Cityscapes images with animal classes leads to 93.14 mIoU on Pascal's test set (considering the same classes) *without seeing any real animal*, while directly training on Pascal yields 94.75—namely, models trained on POC-edited images yield a rather small sim-to-real gap.

We open-sourced the code to reproduce our experiments at https://github.com/naver/poc.

2. Related work

Anomaly segmentation methods. Initial works relied on approximating uncertainty via softmax probabilities [22, 32], model ensembles [29] or dropout [16, 41]. Yet, models tend to be overconfident, resulting in high confidence also for OOD samples [19, 25, 43]. Alternative confidence measures have been

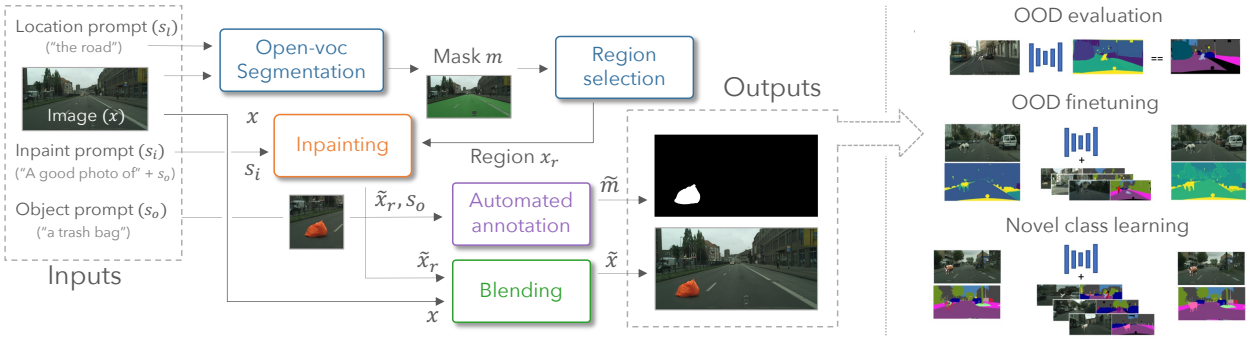

Figure 3: **Illustratation of our POC pipeline and applications.** Our pipeline is built on top of inpainting and open-vocabulary segmentation models to insert arbitrary objects into images realistically. The modified image and mask can be used for different tasks.

proposed that either rely on logits [9,23,26,37] or density estimators [30]. Another body of works reconstruct images with generative models and detect anomalies as discrepancies between original images and their reconstructions [20,34,35,58]. Currently, the most promising methods use OOD data to fine-tune the models [6,17,56]. In particular, they crop OOD objects from COCO [33] and *stitch* them in Cityscapes images. In this work, we build on three state-of-the-art OOD fine-tuning methods [38,42,46] and combine them with POC.

Anomaly segmentation datasets. In Tab. 1 we compare existing anomaly segmentation datasets across four axes (*Domain shift*, *OOD realism*, *Dynamism* and *Set up cost*) while in Fig. 2 we show corresponding sample images.

Fishyscapes Static [3] randomly stitches OOD objects from Pascal [15] to Cityscapes images, thus, there is no *domain shift*, but stitched OOD objects lack *realism*. If OOD object images and masks are available, datasets can be generated *dynamically*. However, if new objects need to be used, they need to be acquired from additional datasets or from the web with mild *set-up costs*.

At the other end of the spectrum, **RoadAnomaly** [34] and **Segment-me-if-you-can (SMIYC)** datasets [5] download real images with anomalies from the web. This ensures *OOD realism*, but often leads to a large *domain shift*. Moreover, manual labelling of OOD objects has a significant *set-up cost* and is *not dynamic*, *i.e.*, new images would need to be acquired and labelled to generate new samples. **Lost & Found** [45] mimicked the Cityscapes set-up to reduce domain shift, but planted OOD objects artificially, which leads to low variability and is even more difficult to scale.

Street Hazards [23] presents a fully simulated dataset. This allows for *dynamic* generation of images while the simulation engine inserts OOD objects *realistically* in terms of lighting. However, the pose and size of the object are pseudo-random, which is not always realistic and there is a strong *domain shift* from simulation to real images. Moreover, accurate 3D models for all objects are required which has a significant *set-up cost* and is hard to scale.

Our proposed pipeline is *plug & play* and can add objects into images with *no set-up costs*. Since it is built on top of open-vocabulary models, it can *dynamically* insert any object by changing the text prompts. We observe qualitatively and quantitatively that our pipeline leads to more *OOD realism* and applying inpainting also avoids *domain shift*.

Generative models. Diffusion models [53] have led to unprecedented quality in image generation [12, 24, 50, 54]. In particular, text-to-image models can receive a text prompt to easily condition the image generation process [44,47,48,51]. Besides image generation, there has been a growing interest in image editing. Particularly relevant to our goal are works that perform image inpainting, *i.e.*, modifying only masked parts of an image [1,47,48]. General-purpose editing models that can edit images based on text prompts [4,40] are also relevant for our goals. In particular, InstructPix2Pix [4] has recently shown very realistic results following text instructions, *e.g.*, "make it look like it was a sunset". However, we found that it often failed to add new objects to the scene *e.g.*, "add a dog on the street". A more detailed discussion can be

found in Appendix C. Rather than training a diffusion model to edit images end-to-end like InstructPix2Pix, we build a pipeline on top of Stable Diffusion [48] to automatically add objects into images.

Similar to us, Du *et al.* [14] use text-to-image models to generate OOD images for classification and Karazija *et al.* [27] to perform zero-shot semantic segmentation. While the latter [27] relies on a frozen feature extractor and focuses on zero-shot segmentation, we *extend* an existing dataset with new classes. Different from the former [14], we target OOD in segmentation and rather than generating fully OOD images, we insert OOD objects realistically.

3. Placing Objects in Context (POC)

We now present our proposed pipeline, Placing Objects in Context (POC). To recap, our desiderata to generate OOD datasets are *i)* minimal shift w.r.t. training set, *ii)* realism, *iii)* dynamic generation and *iv)* low set-up costs.

3.1. The POC pipeline

Our pipeline is built on top of two open-source models with permissive licenses: an inpainting model from Stable Diffusion (SD [48]) and an open-vocabulary segmentation model (GSAM) [18] based on SAM [28] and GroundingDINO [36]. In the following, we detail the four main stages that constitute our pipeline, we provide a full overview of our pipeline in Fig. 3.

Selecting a region to inpaint. In order to insert objects realistically, it is important to find suitable locations for them (*i.e.*, a cat should not be levitating). We leverage GSAM [18] in order to segment a *valid* area to insert the object based on a location prompt s_l, *e.g.*, "the road". Within this valid area, we select a region r with random size and location based on user-specified limits to increase diversity. In Appendix E we conduct a human study to assess the gain in realism obtained by guiding the object location *vs.* a random location.

Object inpainting. After selecting r we crop a square around it x_r and apply SD to obtain $\tilde{x}_r$. The strong vision grounding from SD allows us to add objects more realistically. For instance, we observe that the inpainting model tends to adjust the size of the object (*e.g.*, a bird will be much smaller than a garbage bin for the same region) although there is a tendency towards filling all the inpainting area. We also observe that the illumination of the added object changes based on the image. This is particularly noticeable in night images from ACDC (see Fig. 1).

Automated annotation. In most downstream applications, we need the corresponding mask of the added object. We leverage again the open-vocabulary GSAM, to get $\tilde{m}$ based on an object prompt s_o. We also use this step to reject images with generation failures (*e.g.*, the object was not generated or it was very unrealistic). However, we observed that applying GSAM to the full image often led to false positives and better results were obtained by applying it only to the inpainted region. Also note that using a model like GSAM we can provide more accurate labels than simpler methods like foreground/background segmentation [27] that can not distinguish between a bicycle and its rider.

Object blending. While SD tends to preserve the details of the original image, it introduces slight modifications to the textures (especially fine-grained) and some noticeable changes, *e.g.*, in lane markings. To reduce undesired edits, we blend the original image x and inpainted one $\tilde{x}$ as:

$$\tilde{x} := (1 - m) \odot x + m \odot \tilde{x}$$

where $:=$ indicates an update operation, $\odot$ the element-wise product and $m = \mathcal{G}(\tilde{m})$ is the object mask convolved with a Gaussian kernel. This allows for a smooth transition between the inpainted object and the rest of the image, while preserving some realistic local modifications, like shadows or reflections. We also tried applying an image2image generative model with an empty prompt after inpainting, but after conducting a human study we conclude this does not prevent undesired modifications in general and often modifies the image beyond the inpainting region, which negatively impact realism (see Appendix F). Thus, we keep only the gaussian blur.

3.2. Generating datasets with POC

Although our pipeline can be used for multiple applications, we have two main motivations: *(i)* extending datasets for *anomaly segmentation* and *(ii)* learning new classes. In both cases we use Cityscapes as the known classes and, regardless of the number

of added classes, all POC-datasets have 3× the size of Cityscapes— *i.e.*, we augment each image three times with randomly sampled prompts. Note that this does not lead to any imbalance in terms of fine-tuning steps, since all training schedules have the same total amount of iterations.

Generation prompts. In all datasets we follow a similar approach. Each object class has an object prompt s_o, or several for diversity (*e.g.*, "car", "suv", "van" all belong to the Cityscapes class "car"). Then, the inpainting prompt is built as s_i = "A good photo of" + s_o. Given our datasets are all for autonomous driving, we use as location prompt s_l = "the road" for all objects except the class "bird", which has unconstrained location. We found this simple approach to yield good results without the need for prompt engineering.

Generating OOD *test* sets. To evaluate anomaly segmentation methods, we generate three new test sets, namely *CS-POC*, *IDD-POC* and *ACDC-POC*. We start from the Cityscapes [10], IDD [57], and ACDC[52] validation sets (which have increasing domain shift w.r.t. Cityscapes [11]) and use the same list of OOD objects to augment the images within the validation set. The rationale behind the creation of three sets is to disentangle the difficulty in detecting certain OOD objects (that might be harder to detect as OOD) *vs.* the difficulty caused by a large distribution shift on the ID classes (while the OOD classes remain the same). A second goal is to showcase that POC can be used with different datasets seamlessly (see additional examples for different environment prompts in Appendix D. These datasets contain 25 different OOD classes arbitrarily chosen to be plausible anomalies in an urban environment (*e.g.*, wild animals, garbage bags and bins, *etc.*). Given our OOD objects are all *synthetic*, following [3] we also add ID objects (*e.g.*, cars or persons) to ensure the model is not "simply" performing synth *vs.* real discrimination. A full list of all the objects can be found in Appendix A.

Generating OOD *fine-tuning* sets. Recent work on OOD fine-tuning have used COCO classes not present in Cityscapes to extract OOD objects. For consistency, we generate *POC-coco* using the names of COCO classes used in previous works as prompts to inpaint them with POC (as opposed to cropping and stitching from COCO images). Additionally, we generate another fine-tuning dataset, *POC-alt*, with the same 25 OOD classes used in our evaluation sets to see the dependence of the fine-tuning methods on the number and type of OOD objects.

Extending datasets to *learn new classes*. Given a dataset $\mathcal{D}$ with set of classes $\mathcal{K}$, we consider the task of generating an *extended* dataset $\tilde{\mathcal{D}}$ with set of classes $\tilde{\mathcal{K}} = \mathcal{K} \cup \mathcal{U}$ such that it can be used to train models that perform well on $\mathcal{D}$ as well as learning the additional set of classes $\mathcal{U}$. Following the autonomous driving use-case, we extend the Cityscapes dataset with the 6 animal classes present in the PASCAL dataset (which we use to evaluate). Our motivation is that wild animals cause accidents [49] and being able to segment them individually (not just as anomalies) might help prevent collisions. We call this dataset *POC-A*, however, we also add the same classes on the CS validation set which we call *CS extended*. Additionally, we generate another dataset where we inpaint the animal classes but also CS classes present in PASCAL to study the generalization capabilities.

4. POC for anomaly segmentation

We have already discussed that, in order to reliably evaluate the risk of deploying a model in a certain scenario, we need test images with minimal distribution shift w.r.t. the original distribution and realistic OOD objects. In a similar spirit, we hypothesize that when fine-tuning models for anomaly segmentation, more realistic anomalies will lead to better results. To test this hypothesis, we take three recent methods for OOD fine-tuning that rely on stitching COCO objects into CS images, but instead of COCO, we use our POC pipeline to generate the anomalies for fine-tuning.

4.1. Experimental setting

Anomaly segmentation methods. *RPL* [38] learns a module to detect anomalies via contrastive learning, on top of a segmentation network that is kept frozen. This allows improving OOD detection with minimal degradation to the closed-set performance. *Mask2Anomaly* (M2A) [46] and *RbA* [42] are both based on the novel Mask2Former architecture [8] which performs segmentation at the mask level, *i.e.*, by grouping pixels into "masks" and classifying the whole masks into the closed-set categories. This significantly reduces the pixel-level noise on the anomaly scores. RbA [42] performs OOD fine-tuning with a squared hinge loss while M2A [46] also uses contrastive learning. We use the original code with

Method	OOD data	FS Static			FS Lost&Found			SMIYC Anomaly			SMIYC Obstacle		
		F1* ↑	AuPRC ↑	FPR ↓	F1* ↑	AuPRC ↑	FPR ↓	F1* ↑	AuPRC ↑	FPR ↓	F1* ↑	AuPRC ↑	FPR ↓
M2A [46]	No ft.	65.39	60.07	7.41	45.53	36.77	13.87	88.63	92.99	3.94	69.28	73.62	6.91
	COCO	**82.73**	**88.35**	2.22	70.92	64.46	13.11	90.52	**94.69**	3.28	89.97	94.76	0.39
	POC alt.	82.61	87.40	3.15	74.49	68.84	11.35	**92.21**	93.78	**2.05**	90.84	95.34	0.26
	POC coco	81.99	87.01	**2.12**	**76.52**	**72.97**	**9.21**	88.8	92.13	8.35	**91.35**	**96.04**	**0.13**
RPL [38]	No ft.	20.99	13.95	38.54	5.40	1.57	66.28	50.19	52.99	39.55	46.02	45.15	3.19
	COCO	83.83	89.62	1.23	58.02	58.41	3.22	**73.4**	**78.26**	**18.14**	89.74	**94.29**	**0.34**
	POC alt.	88.37	94.44	0.71	**70.32**	**74.22**	2.67	69.42	78.21	26.89	**89.83**	93.61	0.88
	POC coco	**88.54**	**95.07**	**0.49**	69.45	69.17	**1.64**	62.65	68.00	46.78	87.76	92.31	0.92
RbA [42]	No ft.	58.21	59.15	17.68	63.7	60.95	10.63	86.47	86.89	86.44	92.75	95.88	0.16
	COCO	67.04	72.23	3.98	69.72	70.87	8.69	84.71	**91.07**	5.35	94.85	98.19	**0.04**
	POC alt.	68.31	77.2	3.4	74.05	76.74	5.36	**86.48**	90.47	**4.36**	**94.92**	98.35	**0.04**
	POC coco	**68.56**	**77.58**	**3.32**	**76.48**	**78.87**	**3.32**	85.21	89.4	4.99	94.59	**98.39**	**0.04**

Method	OOD data	RoadAnomaly			Cityscapes – POC			IDD – POC			ACDC – POC		
		F1* ↑	AuPRC ↑	FPR ↓	F1* ↑	AuPRC ↑	FPR ↓	F1* ↑	AuPRC ↑	FPR ↓	F1* ↑	AuPRC ↑	FPR ↓
M2A [46]	No ft.	54.75	55.74	52.15	40.95	35.36	13.05	45.57	42.94	11.96	18.58	10.34	26.74
	COCO	77.19	78.89	**18.50**	88.26	93.88	0.54	80.98	85.54	**1.03**	70.78	72.83	**6.17**
	POC alt.	**81.45**	**82.29**	36.65	**91.39**	**95.81**	0.43	82.82	87.74	1.09	**74.02**	**74.45**	7.56
	POC coco	78.92	77.96	24.61	89.09	94.67	**0.37**	**83.05**	**89.07**	1.26	72.77	72.04	8.38
RPL [38]	No ft.	24.4	14.99	70.42	22.74	13.79	44.87	7.95	3.45	61.43	3.31	1.26	79.01
	COCO	60.08	59.20	27.14	82.06	88.41	0.71	72.35	78.68	1.76	63.64	65.42	2.56
	POC alt.	**66.39**	**69.42**	21.68	86.1	93.18	0.48	79.66	85.78	0.97	**79.42**	**85.68**	**0.78**
	POC coco	61.38	64.16	**21.36**	**87.58**	**94.30**	**0.36**	**82.8**	**89.97**	**0.69**	76.0	81.16	1.31
RbA [42]	No ft.	72.78	78.4	11.83	73.49	77.89	3.67	65.54	65.05	78.92	24.25	18.65	90.04
	COCO	78.54	83.4	8.31	87.17	92.92	0.49	79.22	85.25	1.19	33.45	31.96	11.19
	POC alt.	**78.32**	**84.08**	**8.29**	89.34	95.02	0.37	83.69	**89.1**	**0.73**	54.99	58.37	**8.39**
	POC coco	77.34	82.95	8.82	**90.48**	**95.83**	**0.3**	**83.77**	89.09	0.89	**57.59**	**61.14**	9.06

Table 2: **Anomaly segmentation metrics after OOD finetuning.** We use three recent OOD fine-tuning methods and report results prior to fine-tuning (*No ft.*), after fine-tuning with *COCO* objects and using our POC pipeline to inpaint coco objects (*POC coco*) or an alternative set of 25 objects (*POC alt.*) likely to be found on the street. Best and second best numbers for each method are highlighted in **bold** and underlined respectively. The best number over all methods is shaded in **gray**. Using our pipeline improves performance in most cases. Moreover, fine-tuning with COCO also leads to significant improvements in our POC eval sets, consistent with previous benchmarks.

default settings for each method and only modify the fine-tuning dataset.

OOD fine-tuning data. For each method, we consider different ways of generating the dataset used to fine-tune the anomaly segmentation modules. First, we consider a baseline where no fine-tuning occurs (*No ft.* in our tables). Then, we consider fine-tuning on *COCO* stitching. When we fine-tune using our POC-generate images, we consider two cases: *POC coco* (inpainting the same classes as COCO stitching) and *POC alt.* (inpainting alternative classes, more likely to be found on the street).

Anomaly datasets and metrics. We evaluate on five commonly used datasets, discussed in Sec. 2, and on our POC-generated datasets. Following previous work [6, 38], we compute three different metrics: maximum F1 score over all thresholds ($F1^*$), Area under the Precision-Recall Curve (AuPRC) and False Positive Rate at 95% recall (FPR).

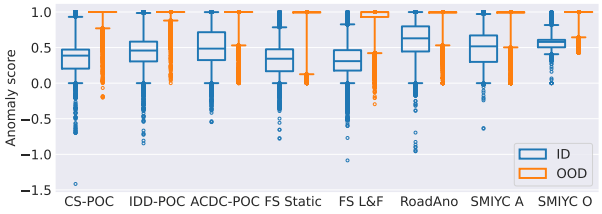

Figure 4: **Boxplots of anomaly scores.** All datasets have consistently very high scores for OOD pixels while ID pixels of datasets with strong distribution shifts also have shifted scores. Thus, distribution shifts may lead to underestimated performance.

4.2. Results

POC improves OOD detection. In Tab. 2, we show fine-tuning with *POC coco* is remarkably better than the *No ft.* baseline despite only using data with synthetic anomalies. Moreover, *POC coco* also brings important improvements over COCO fine-tuning in some settings. For instance, in FS Static (for RPL or RbA), in SMIYC Obstacle (for M2A or RbA) and in FS Lost & Found (for all methods). In other settings, it is competitive with COCO fine-tuning except in SMIYC Anomaly (for RPL), where we observe a significant drop. One reason may be that the strong domain shift in SMIYC Anomaly limits the benefits of using more realistic data. On the other hand, in FS Lost & Found, which has the closest setting to Cityscapes and real OOD objects, we carry the largest improvements.

Robustness to the choice of OOD classes. We observe that inpainting different OOD classes than COCO, *POC alternative*, also leads to strong anomaly segmentation results, improving over the COCO baseline in several settings and sometimes surpassing *POC coco*. This indicates that fine-tuning methods are somewhat robust to the choice of OOD classes. Importantly, the flexibility of the POC pipeline opens the possibility of studying which classes might be best depending on the use-case in future work.

Finally, note that the best score of all methods (highlighted with gray background) corresponds to one of the POC fine-tuning datasets in almost all settings.

Performance on POC evalutation sets. As expected, fine-tuning with our pipeline brings the best results on our POC eval sets. Yet, we also observe that COCO fine-tuning leads to notable improvements, similar to the ones observed in previous datasets. This indicates that POC datasets accurately reflect anomaly segmentation capabilities and can be used to efficiently build OOD detection benchmarks. Interestingly, we also observe that POC-alt does not always outperform POC-coco despite sharing the same anomaly classes as the POC test sets. One explanation could be that POC-coco has more anomaly classes (80 compared to 25 in POC-alt), thus, potentially more diversity.

Domain shifts hinder anomaly segmentation. Considering our synthetic evaluation datasets, if we move from POC-CS to POC-IDD and POC-ACDC we observe a drop in performance in all methods. Since the sets contain the same anomaly classes, the drop is due to the domain shift between train and evaluation rather than hard-to-detect anomalies. Additionally, in Fig. 4 we show boxplots of the predicted anomaly scores (higher meaning higher chance of anomaly) for ID *vs.* OOD pixels. Interestingly, we observe that OOD pixels have very high scores independently of the dataset, while ID scores vary significantly between datasets. Datasets with strong domain shifts (*i.e.*, SMIYC Anomaly, RoadAnomaly and POC-ACDC) carry larger ID anomaly scores. While anomaly segmentation under domain shift might also be an interesting task, we argue that, in practice, agents will be carefully deployed in restricted areas (*e.g.*, one city). To accurately evaluate the risk represented by anomalies, ideally, there should be no domain shift.

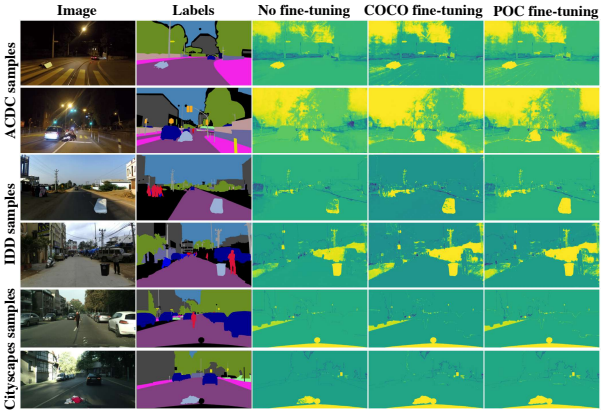

Figure 5: **Anomaly score maps.** Per-pixel anomaly scores on POC-generated images obtained with M2A [46] before and after fine-tuning with COCO and POC data. COCO and POC fine-tuning have notable improvements over the *No ft.* baseline, *e.g.*, note the garbage bag or matress in second and third images.

Anomaly segmentation maps. In Fig. 5 we show per-pixel anomaly scores on POC-generated images computed with Mask2Anomaly prior to fine-tuning and after fine-tuning either with COCO or POC data. Aside from the inpainted anomalies (labelled in gray), we observe that in ACDC (Top) the night sky has a particularly high score and in IDD (Middle) a peculiar instance of a known class (the all-road car) is also highlighted, potentially misleading anomaly segmentation. More anomaly score maps can be found in Appendix I.

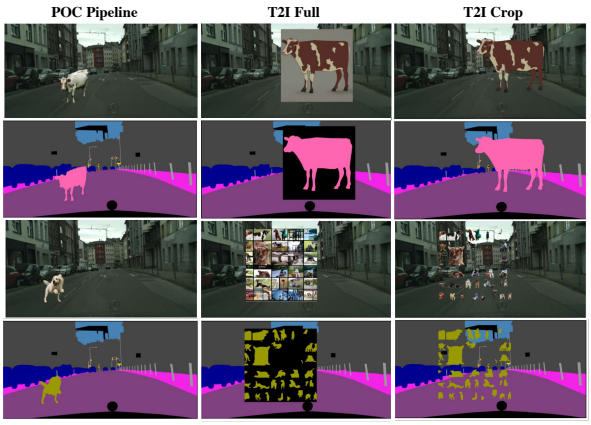

Figure 6: **Training image samples.** Text2Image (T2I) methods often lead to unrealistic objects. Moreover, sometimes T2I outputs are misaligned with caption, *e.g.*, "an image of a dog" leading to a *collage* of dog images. More samples in Appendix H.

5. POC to learn new classes

Besides anomaly segmentation, another natural application for our POC pipeline is dataset *extension*. In this section we consider extending the dataset in two directions: adding novel objects to learn new classes and adding instances of existing classes to improve generalization.

5.1. Experimental settings

***Extended* datasets.** We use Cityscapes as our base dataset. Given our autonomous driving motivation, we are interested in classes that may cause road accidents if undetected. Thus, we add *animal* classes to obtain *POC A* and *POC CS+A* (both animal and cityscapes classes). Similar to OOD detection, finding a test set (*i.e.*, Cityscapes images with animals) is challenging. Inspired by Karazija *et al.* [27], we use Pascal [15] animal classes to assess the synth2real performance of POC-trained models. For completeness, we also evaluate on Cityscapes *extended* with POC.

T2I baseline. Karazija *et al.* [27] use text2image diffusion models to cluster the features of a frozen model for zero-shot segmentation. Although it is not the same task (in our case we would not like to use a frozen model but to train one from scratch), we include two augmented datasets in our experiments: *T2I Full* and *T2I Crop* where we take the Full (or cropped) image generated by a T2I model and stitch it into Cityscapes images (we use the text2image model from the same work used in our POC [48]). In Fig. 6 we show a comparison of the synthetic datasets. Using T2I baselines often leads to unrealistic object size and position. Moreover, perhaps due to the lack of context, the T2I model often generates images misaligned with our goal (*e.g.*, a composition of many small dog images). We obtain the segmentation labels with the same open-vocabulary model as our POC pipeline, which is more flexible than foreground/background segmentation suggested in the original work [27].

Architectures. We perform experiments using three different models: *DLV3+* [7] with ResNet101 [21]; *ConvNext*, a recent convolutional architecture[39]; *Segmenter* [55], a recent transformer architecture. We use the default training settings for each model. Note our goal is not to compare the architectures, but the extended datasets.

5.2. Results

POC performs better than T2I baselines. In Tab. 3 we show the mIoU for each model after training on our different datasets, as well as two baselines trained on Cityscapes and Pascal. In all architectures, we find that training with POC datasets leads to better results than their T2I counterparts on CS extended and Pascal (animal), *i.e.*, only animal classes of Pascal. We argue this is due to a higher realism in generated images as observed in Fig. 6. Interestingly, POC also improves performance on the original Cityscapes. Perhaps extending classes acts as a form of regularization.

Generalization is key to learn from synthetic data. We observe that the performance of DLV3+ trained on *POC A* when evaluated on Pascal (30.43) is much lower than that of Segmenter (92.4), which achieves an *mIoU competitive with the baseline trained directly on Pascal* (94.75). In order to understand that, we evaluate the performance of such models on the Pascal classes also present on Cityscapes (*i.e.*, car, motorcycle, bike, person, train and bus). This shows that DLV3+ has much less generalization capability independently of the synthetic classes (*i.e.*, DLV3+ trained on Cityscapes has a much lower mIoU than Segmenter). One could argue that perhaps the low performance of DLV3+ on Pascal is due to the large domain shift, however, in Fig. 7 we show qualitative results on web images of driving scenes, closer to the

Model	Train set	CS (19 cl.)	CS ext. (19 + 6 cl.)	Pascal (animal)	Pascal (citysc.)
DLV3+	Pascal (baseline)	–	–	80.57	85.81
	CS (baseline)	79.09	–	–	42.22
	T2I Full	77.13	73.2	28.32	23.51
	T2I Crop	78.23	81.49	26.15	25.09
	POC A	**79.98**	**84.09**	**30.43**	35.83
	POC CS+A	79.9	83.8	28.11	**53.57**
CNXT	Pascal (baseline)	–	–	94.43	93.92
	CS (baseline)	81.57	–	–	70.49
	T2I Full	82.26	82.99	60.41	65.6
	T2I Crop	82.43	86.04	61.09	67.55
	POC A	**82.94**	**86.68**	65.46	70.87
	POC CS+A	82.54	86.09	**69.75**	**82.43**
Segm.	Pascal (baseline)	–	–	94.75	91.43
	CS (baseline)	76.19	–	–	79.87
	T2I Full	77.19	79.46	81.96	74.32
	T2I Crop	77.62	81.27	75.95	75.44
	POC A	**78.48**	**82.28**	92.4	79.1
	POC CS+A	78.39	81.92	**93.14**	**89.55**
GSAM	Open Vocab.	42.0	41.06	75.13	76.08

Table 3: **mIoU evaluation.** We train three architectures on Pascal and Cityscapes (as baselines) and compare two Text2Image (T2I) generation methods with our POC. We also add the open vocabulary model used in POC for completeness.

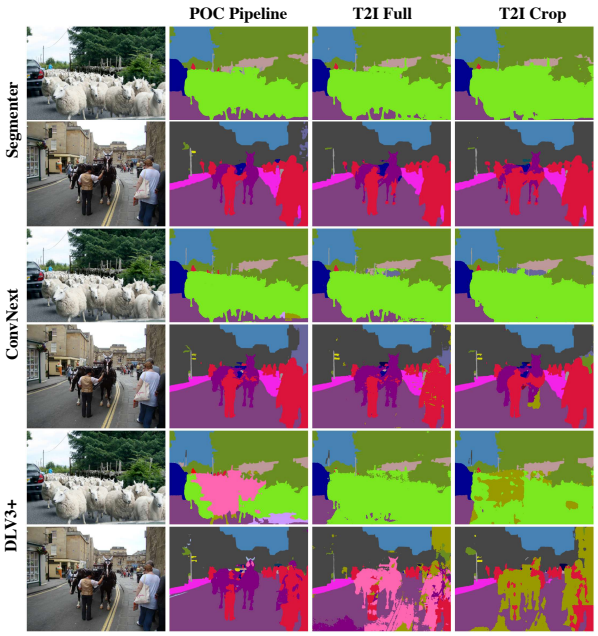

Figure 7: **Qualitative results.** We present results on web images that have a milder domain shift compared to Pascal dataset. We observe less notable differences between CNXT and Segmenter but DLV3+ is still significantly worse. More results in Appendix J.

Cityscapes domain, and still observe that DLV3+ is notably worse than the other methods.

We hypothesize that, in order to learn transferable features from synthetic data, the generalization capability of the model plays a key role. Indeed, although generative models have improved the realism of generated content remarkably, there is still a synth2real gap. Thus, more robust models (*i.e.*, with strong transferability of learned features) may be able to extract more useful features rather than overfitting to brittle patterns in generated data. Pre-training might also play a role: DLV3+ backbone was pretrained on Imagenet-1k, while ConvNeXt and Segmenter on Imagenet-21k. Nevertheless, we also observe a gap between ConvNeXt and Segmenter, both in the generalization from CS to Pascal (CS classes) and in Pascal (animals) when trained on *POC A*. Perhaps self-attention or image tokenization play a role, but this would require a more in-depth analysis. Yet, understanding the generalization gap between such models is out of the scope of this work.

POC to augment existing classes. Given the competitive performance of Segmenter with *POC A* on Pascal (animals) compared to Pascal (CS classes), we are compelled to also inpaint Cityscapes classes with POC, obtaining *POC CS+A*. We observe remarkable improvements in Pascal (CS classes) in all three networks, significantly outperforming the baseline that was only trained on Cityscapes. Coupled with the milder but consistent improvement on the original Cityscapes, this indicates that our pipeline could also be helpful in improving generalization.

Open vocabulary baseline. Considering that we use a single open-vocabulary segmentation method (GSAM [18]) to label all objects inserted with POC, we also evaluate its performance on this task as a baseline. Although its performance on Pascal is reasonable, it significantly underperforms on Cityscapes. We hypothesize this may be due to the one-vs-all nature of open-vocabulary predictors (where each class is predicted individually with a prompt) that does not perform well on complex images with many classes. Interestingly, the Segmenter model trained with POC significantly outperforms GSAM on Pascal. We find this remarkable as we could interpret it as knowledge distillation from a teacher model (GSAM) while significantly improving its performance. On the other hand, note that the setting is very different when we use GSAM in the POC pipeline, as we know which object

is inpainted. We also observe a strong improvement when labelling only the cropped region $\tilde{x}_r$ instead of the full image $\tilde{x}$.

6. Concluding remarks

To accurately assess anomaly segmentation capabilities of models deployed in open-world settings, we argue that datasets should be realistic and carry a small domain shift w.r.t. the training distribution, as we show it can hinder OOD detection. Towards this goal, we introduce the Placing Object in Context (POC) pipeline, that allows adding *any* (OOD) object into *any* image based on simple prompting. POC uses diffusion models and open-vocabulary segmentation to achieve high realism and versatility.

We showcase POC's flexibility by generating three anomaly segmentation test sets: POC-CS, POC-IDD and POC-ACDC. Moreover, we observe that generating data for OOD fine-tuning with POC brings significant improvements in standard anomaly segmentation benchmarks.

Beyond anomaly segmentation, we use POC-generated datasets to learn new classes without any real example. Interestingly, we observe that the combination of more realistic synthetic data with recent segmentation models with strong generalization capabilities can lead to remarkable performances, competitive with training on real data.

In future work, we hope to better understand how to select the optimal list of anomalies for fine-tuning and how can modern architectures make the best use of synthetic data to learn new classes, also in a continual learning setup.

References

[1] Omri Avrahami, Dani Lischinski, and Ohad Fried. Blended diffusion for text-driven editing of natural images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18208–18218, 2022. 3

[2] Victor Besnier, Andrei Bursuc, David Picard, and Alexandre Briot. Triggering failures: Out-of-distribution detection by learning from local adversarial attacks in semantic segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15701–15710, 2021. 1

[3] Hermann Blum, Paul-Edouard Sarlin, Juan Nieto, Roland Siegwart, and Cesar Cadena. Fishyscapes: A benchmark for safe semantic segmentation in autonomous driving. In *proceedings of the IEEE/CVF international conference on computer vision workshops*, pages 0–0, 2019. 2, 3, 5

[4] Tim Brooks, Aleksander Holynski, and Alexei A Efros. Instructpix2pix: Learning to follow image editing instructions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18392–18402, 2023. 3, 1, 2

[5] Robin Chan, Krzysztof Lis, Svenja Uhlemeyer, Hermann Blum, Sina Honari, Roland Siegwart, Pascal Fua, Mathieu Salzmann, and Matthias Rottmann. Segmentmeifyoucan: A benchmark for anomaly segmentation. *arXiv preprint arXiv:2104.14812*, 2021. 2, 3

[6] Robin Chan, Matthias Rottmann, and Hanno Gottschalk. Entropy maximization and meta classification for out-of-distribution detection in semantic segmentation. In *Proceedings of the ieee/cvf international conference on computer vision*, pages 5128–5137, 2021. 3, 6

[7] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2017. 8

[8] Bowen Cheng, Ishan Misra, Alexander G. Schwing, Alexander Kirillov, and Rohit Girdhar. Masked-attention mask transformer for universal image segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, 2022. 5

[9] Charles Corbière, Nicolas Thome, Avner Bar-Hen, Matthieu Cord, and Patrick Pérez. Addressing failure prediction by learning model confidence. *Advances in Neural Information Processing Systems*, 32, 2019. 3

[10] 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. 5

[11] Pau de Jorge, Riccardo Volpi, Philip HS Torr, and Grégory Rogez. Reliability in semantic segmentation: Are we on the right track? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7173–7182, 2023. 5

[12] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021. 3

[13] Giancarlo Di Biase, Hermann Blum, Roland Siegwart, and Cesar Cadena. Pixel-wise anomaly detection in complex driving scenes. In *Proceedings of the*

IEEE/CVF conference on computer vision and pattern recognition, pages 16918–16927, 2021. 1

[14] Xuefeng Du, Yiyou Sun, Xiaojin Zhu, and Yixuan Li. Dream the impossible: Outlier imagination with diffusion models. *arXiv preprint arXiv:2309.13415*, 2023. 4

[15] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html, 2012. 3, 8

[16] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059. PMLR, 2016. 2

[17] Matej Grcić, Petra Bevandić, and Siniša Šegvić. Dense-hybrid: Hybrid anomaly detection for dense open-set recognition. In *European Conference on Computer Vision*, pages 500–517. Springer, 2022. 1, 3

[18] Grounded-SAM Contributors. Grounded-Segment-Anything. LICENSE Apache-2.0. https://github.com/IDEA-Research/Grounded-Segment-Anything, April 2023. 2, 4, 9

[19] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International conference on machine learning*, pages 1321–1330. PMLR, 2017. 2

[20] David Haldimann, Hermann Blum, Roland Siegwart, and Cesar Cadena. This is not what i imagined: Error detection for semantic segmentation through visual dissimilarity. *arXiv preprint arXiv:1909.00676*, 2019. 3

[21] 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. 8

[22] Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *arXiv preprint arXiv:1610.02136*, 2016. 2

[23] Dan Hendrycks, Steven Basart, Mantas Mazeika, Andy Zou, Joe Kwon, Mohammadreza Mostajabi, Jacob Steinhardt, and Dawn Song. Scaling out-of-distribution detection for real-world settings. *arXiv preprint arXiv:1911.11132*, 2019. 2, 3

[24] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020. 3

[25] Heinrich Jiang, Been Kim, Melody Guan, and Maya Gupta. To trust or not to trust a classifier. *Advances in neural information processing systems*, 31, 2018. 2

[26] Sanghun Jung, Jungsoo Lee, Daehoon Gwak, Sungha Choi, and Jaegul Choo. Standardized max logits: A simple yet effective approach for identifying unexpected road obstacles in urban-scene segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15425–15434, 2021. 1, 3

[27] Laurynas Karazija, Iro Laina, Andrea Vedaldi, and Christian Rupprecht. Diffusion models for zero-shot open-vocabulary segmentation. *arXiv preprint arXiv:2306.09316*, 2023. 4, 8, 5

[28] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. *arXiv preprint arXiv:2304.02643*, 2023. 4

[29] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in neural information processing systems*, 30, 2017. 2

[30] Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. *Advances in neural information processing systems*, 31, 2018. 3

[31] Chen Liang, Wenguan Wang, Jiaxu Miao, and Yi Yang. Gmmseg: Gaussian mixture based generative semantic segmentation models. *Advances in Neural Information Processing Systems*, 35:31360–31375, 2022. 1

[32] Shiyu Liang, Yixuan Li, and Rayadurgam Srikant. Enhancing the reliability of out-of-distribution image detection in neural networks. *arXiv preprint arXiv:1706.02690*, 2017. 2

[33] 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 *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pages 740–755. Springer, 2014. 3

[34] Krzysztof Lis, Krishna Nakka, Pascal Fua, and Mathieu Salzmann. Detecting the unexpected via image resynthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2152–2161, 2019. 2, 3

[35] Krzysztof Lis, Sina Honari, Pascal Fua, and Mathieu Salzmann. Detecting road obstacles by erasing them. *arXiv preprint arXiv:2012.13633*, 2020. 1, 3

[36] Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Chunyuan Li, Jianwei Yang, Hang Su, Jun Zhu, et al. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. *arXiv preprint arXiv:2303.05499*, 2023. 4

[37] Weitang Liu, Xiaoyun Wang, John Owens, and Yixuan Li. Energy-based out-of-distribution detection.

Advances in neural information processing systems, 33: 21464–21475, 2020. 3

[38] Yuyuan Liu, Choubo Ding, Yu Tian, Guansong Pang, Vasileios Belagiannis, Ian Reid, and Gustavo Carneiro. Residual pattern learning for pixel-wise out-of-distribution detection in semantic segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1151–1161, 2023. 3, 5, 6, 4

[39] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11976–11986, 2022. 8

[40] Chenlin Meng, Yutong He, Yang Song, Jiaming Song, Jiajun Wu, Jun-Yan Zhu, and Stefano Ermon. Sdedit: Guided image synthesis and editing with stochastic differential equations. *arXiv preprint arXiv:2108.01073*, 2021. 3

[41] Jishnu Mukhoti and Yarin Gal. Evaluating bayesian deep learning methods for semantic segmentation. *arXiv preprint arXiv:1811.12709*, 2018. 2

[42] Nazir Nayal, Misra Yavuz, Joao F Henriques, and Fatma Güney. Rba: Segmenting unknown regions rejected by all. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 711–722, 2023. 1, 3, 5, 6, 4

[43] Anh Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 427–436, 2015. 2

[44] Alex Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob McGrew, Ilya Sutskever, and Mark Chen. Glide: Towards photorealistic image generation and editing with text-guided diffusion models. *arXiv preprint arXiv:2112.10741*, 2021. 3

[45] Peter Pinggera, Sebastian Ramos, Stefan Gehrig, Uwe Franke, Carsten Rother, and Rudolf Mester. Lost and found: detecting small road hazards for self-driving vehicles. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1099–1106. IEEE, 2016. 2, 3, 1

[46] Shyam Nandan Rai, Fabio Cermelli, Dario Fontanel, Carlo Masone, and Barbara Caputo. Unmasking anomalies in road-scene segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4037–4046, 2023. 3, 5, 6, 7, 4

[47] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 1(2):3, 2022. 3

[48] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022. 2, 3, 4, 8

[49] Waleed Saad and Abdulaziz Alsayyari. Loose animal-vehicle accidents mitigation: Vision and challenges. In *2019 International Conference on Innovative Trends in Computer Engineering (ITCE)*, pages 359–364. IEEE, 2019. 5

[50] Chitwan Saharia, William Chan, Huiwen Chang, Chris Lee, Jonathan Ho, Tim Salimans, David Fleet, and Mohammad Norouzi. Palette: Image-to-image diffusion models. In *ACM SIGGRAPH 2022 Conference Proceedings*, pages 1–10, 2022. 3

[51] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, et al. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in Neural Information Processing Systems*, 35:36479–36494, 2022. 3

[52] Christos Sakaridis, Dengxin Dai, and Luc Van Gool. Acdc: The adverse conditions dataset with correspondences for semantic driving scene understanding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10765–10775, 2021. 5

[53] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pages 2256–2265. PMLR, 2015. 3

[54] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32, 2019. 3

[55] Robin Strudel, Ricardo Garcia, Ivan Laptev, and Cordelia Schmid. Segmenter: Transformer for semantic segmentation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 7262–7272, 2021. 8

[56] Yu Tian, Yuyuan Liu, Guansong Pang, Fengbei Liu, Yuanhong Chen, and Gustavo Carneiro. Pixel-wise energy-biased abstention learning for anomaly segmentation on complex urban driving scenes. In *European Conference on Computer Vision*, pages 246–263. Springer, 2022. 3

[57] Girish Varma, Anbumani Subramanian, Anoop Namboodiri, Manmohan Chandraker, and CV Jawahar. Idd: A dataset for exploring problems of autonomous navigation in unconstrained environments. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1743–1751. IEEE, 2019. 5

[58] Yingda Xia, Yi Zhang, Fengze Liu, Wei Shen, and Alan L Yuille. Synthesize then compare: Detecting failures and anomalies for semantic segmentation. In

Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I 16, pages 145–161. Springer, 2020. 3

A. POC datasets object list

Our POC evaluation sets (*i.e.*, POC-CS, POC-IDD and POC-ACDC) are obtained by adding objects to different self-driving datasets. Following works like Lost and Found [45], we compiled a list of 25 objects that can be found on the road.

The **anomaly list** is as follows: “stroller”, “trolley”, “garbage bag”, “wheelie bin”, “suitcase”, “skateboard”, “chair dumped on the street”, “sofa dumped on the street”, “furniture dumped on the street”, “matress dumped on the street”, “garbage dumped on the street”, “clothes dumped on the street”, “cement mixer on the street”, “cat”, “dog”, “bird flying”, “horse”, “skunk”, “sheep”, “crocodile”, “alligator”, “bear”, “llama”, “tiger” and “monkey”.

Additionally, we also add a few classes from Cityscapes to make sure that anomaly segmentation models are indeed detecting *anomalies* and not merely identifying *synthetic objects*.

Cityscapes classes inluded are: “rider”, “bicycle”, “motorcycle”, “bus”, “person” and “car”.

B. Anomaly segmentation methods: mIoU on Cityscapes

Although fine-tuning methods with OOD samples can improve anomaly segmentation significantly, they may affect the closed-set performance. In Tab. 4 we report the mIoU on Cityscapes of all methods reported in the main paper, showing that fine-tuning with POC data does not negatively impact closed-set performance.

Method	**Mask2Anomaly**				**RPL**				**RbA**			
OOD data	No ft.	COCO	POC coco	POC alt.	No ft.	COCO	POC coco	POC alt.	No ft.	COCO	POC coco	POC alt.
mIoU ↑	78.29	78.34	78.33	78.49	90.94	90.94	90.94	90.94	82.25	82.15	82.17	82.16

Table 4: **mIoU on Cityscapes validation set.** We compute the mIoU after fine-tuning with different datasets (complementing results in Tab. 2). We observe that fine-tuning with our POC datasets does not degrade the closed-set performance.

C. Adding objects with Instruct Pix2Pix

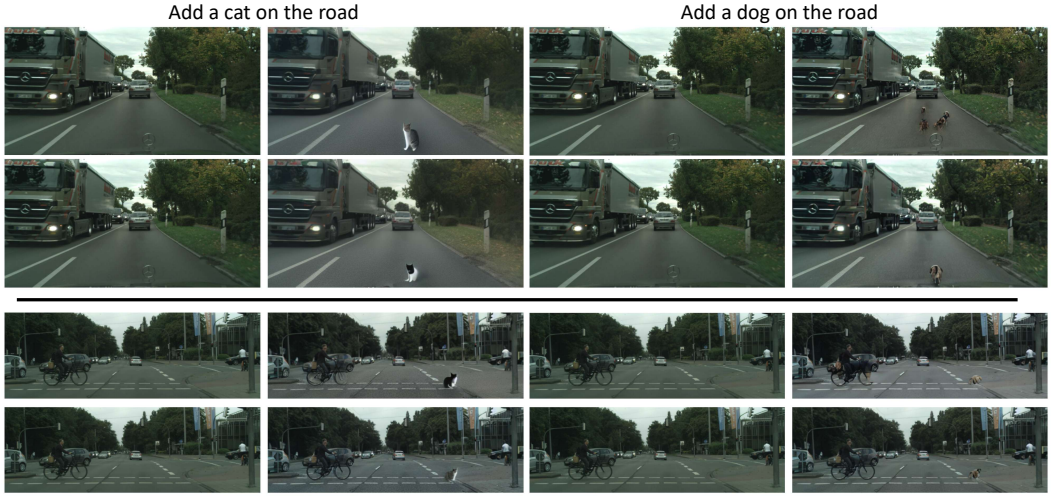

Figure 8: **Sample images from InstructPix2Pix [4].** We observe that InstructPix2Pix has a bias to replace certain objects or features in the scene. In top images it replaces the mercedes logo of the ego vehicle while in bottom images it replaces the edge of the sidewalk.

When building our pipeline, we explored different generative methods. In particular, InstructPix2Pix (IP2P)

[4] has showed remarkable performance following natural language instructions (*e.g.*, "turn the sofa red"). Therefore, it would be natural to have such "general-purpose editing" methods as baselines to add objects to the images. We observed that IP2P seems to be biased towards *modifying* objects in the scene rather than *adding new ones*. In Fig. 8 we show several examples of images generated with IP2P. In our initial experiments, we observe how new objects tend to replace the logo of the ego vehicle (top images). If we remove the bottom of the image (middle section), we then observe that some particular image features (*e.g.*, the edge of the sidewalk) tend to be replaced. Moreover, the added objects tend to lack realism and, given that the changes are not constrained to a particular region, editing via IP2P usually results in undesired modifications in other image regions.

D. Additional inpainting examples

In the main text we show examples where POC is applied to driving scenes, with the rationale that we want to compare against prior anomaly segmentation datasets. Yet, to show the versatility of POC we also include here a few examples of objects inpainted in completely different environments in Fig. 9.

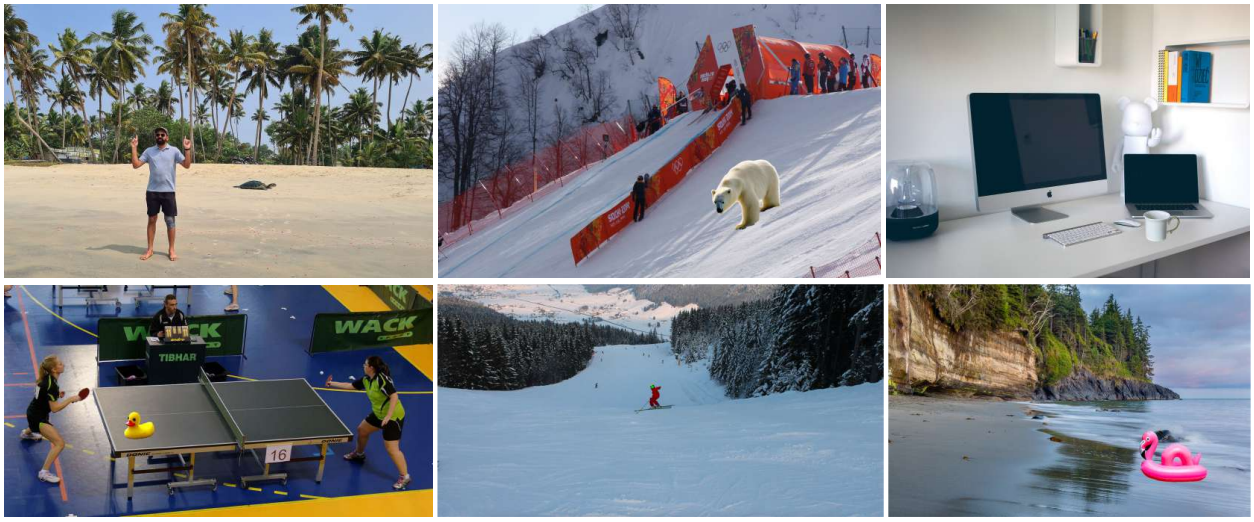

Figure 9: **POC examples:** Images with inpainted objects using our POC pipeline. Left to right and top to bottom, inserted objects are: "sea turtle", "polar bear", "white porcelain mug", "rubber duck", "person skiing" and "inflatable flamingo". In the same order, location prompts are: "the beach", "snow", "the table", "ping pong table", "snow", and "the beach".

E. Ablation of guided region selection

We argued in the main text that in order to add objects into scenes realistically, it is important to properly place them. Thus, we apply GSAM to segment a valid area based on a location prompt (*e.g.*, "the road") and then select a region randomly within the valid area. Without this component in our pipeline, the objects result inpainted in clearly unrealistic positions.

To assess the realism introduced by guiding the object location *vs.* placing objects randomly, we conducted a human study where participants were shown different pairs of images, one with guided location inpainting and the other with random placement, and asked to choose the most realistic image in each pair. We observed that 39% of times the preference was unclear, 43% guided location was preferred and 18% random location was preferred. Note that a large portion of Cityscapes images is road/street, thus, a significant portion of randomly placed objects will be realistic. On the other hand, when the location is different, the generated objects also vary (even if fixing the random seed), which adds some noise to the study. All in all, we do observe a clear preference for guided location compared to random placement. In Fig. 10 we show some examples of image

pairs with guided and random locations. Note how the added “dumped clothes” in the bottom right are both in realistic locations while the other objects are placed unrealistically with the random location.

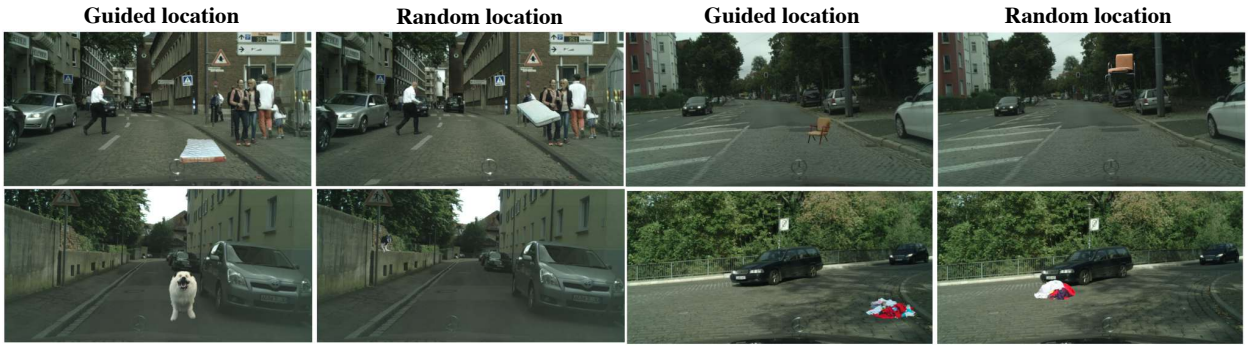

Figure 10: **Location ablation examples:** Different examples of images inpainted with our guided location or random location of objects. We observe how in many cases, random location leads to unrealistic scenes.

F. Ablation of image2image blending

Similar to our location ablation, we also study if applying an image2image (I2I) model after object inpainting leads to better blending. In particular, we performed a human study where participants had to choose the most realistic image between our blending ans I2I. In 71% of the cases I2I blending did not improve results significantly, in 25% it introduced artifacts that significantly reduced realism and in only 4% participants preferred I2I blending.

In particular, we noted that I2I blending adds slight artefacts that can degrade the realism of the image significantly, these become especially noticeable in text or traffic signs where small variations can change the semantics drastically. In Fig. 11 we present two examples of such images where artefacts are highlighted.

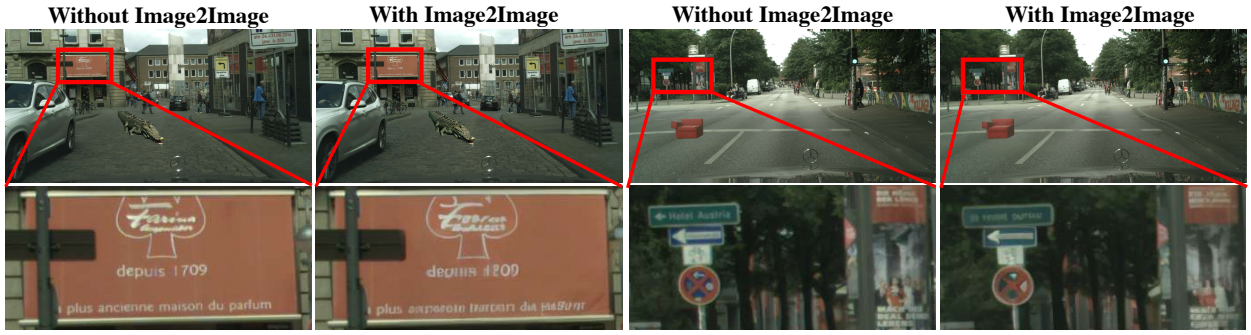

Figure 11: **Examples of object blending:** On the left image pair, we observe the presence of articles in the text of an old perfume shop which is legible on the right image but becomes illegible with I2I. On the right image pair, one can observe differences on the traffic signs. For instance, the text “Hotel Austria" on the green sign on the top (legible when zoomed on the left) becomes again illegible on the right image. Also, the white squared sign with a depiction of a bike without I2I becomes uninterpretable after I2I.

G. AUPRC plots

In Fig. 12 we visualize the AUPRC results for all methods, complementing the visualization in Fig. 1.

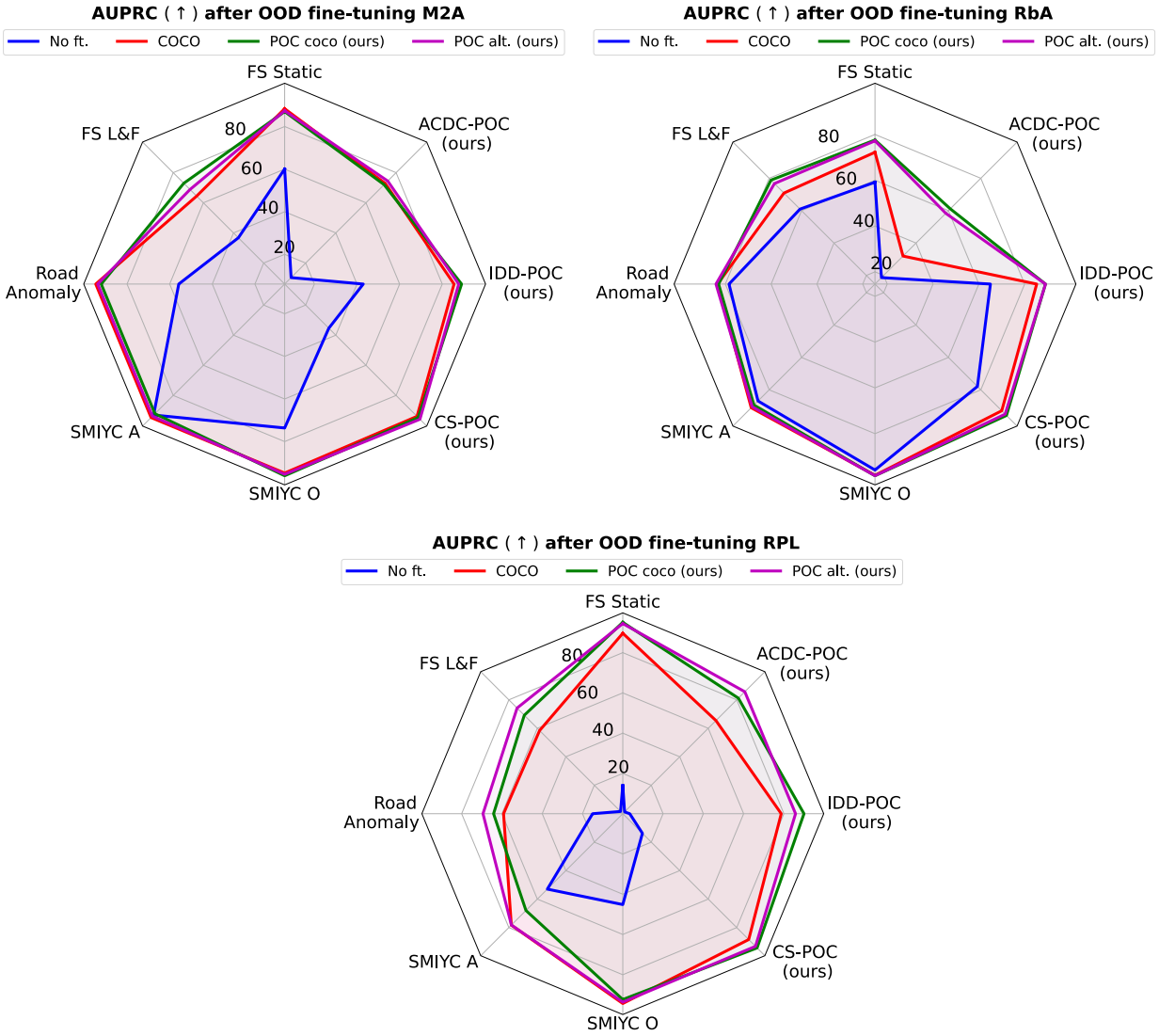

Figure 12: **AUPRC on different anomaly segmentation datasets.** We compare three different anomaly segmentation methods, M2A[46], RPL[38] and RbA[42] with different fine-tuning datasets. Fine-tuning with POC-generated images tends to bring improvements or match COCO fine-tuning in most settings.

H. Additional Pascal training samples

In Fig. 13 we show additional samples generated with our POC pipeline as well as T2I baselines (that use text-to-image models inspired by [27]).

Figure 13: **Training image samples.** Additional training images to learn new classes, complementing Fig. 6.

I. Additional anomaly score maps

In this appendix section we add more visualizations of anomaly score maps with different methods. We show results from our three POC-generated datasets as well as samples from previous anomaly datasets.

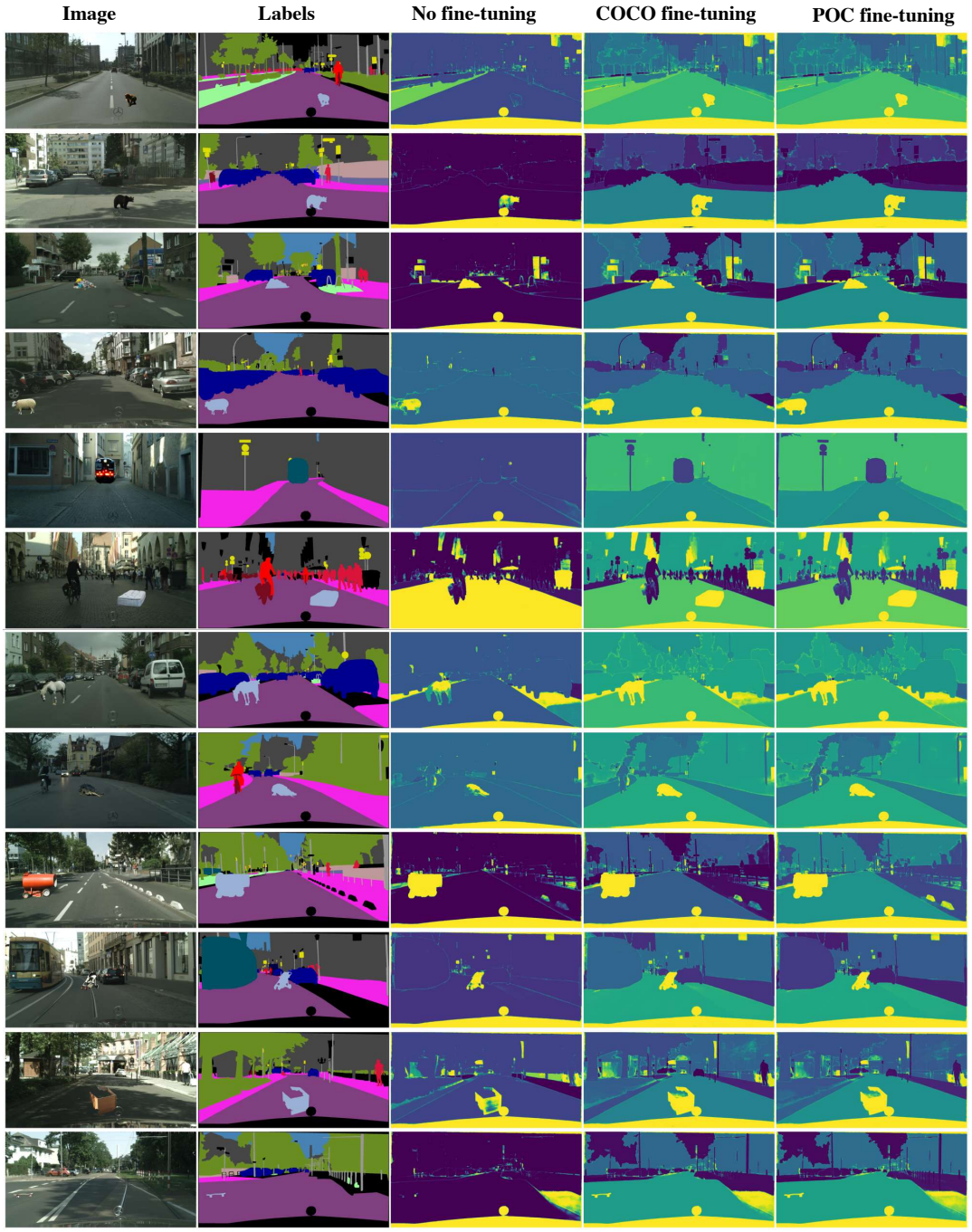

Figure 14: **M2A anomaly scores on CS-POC samples.**

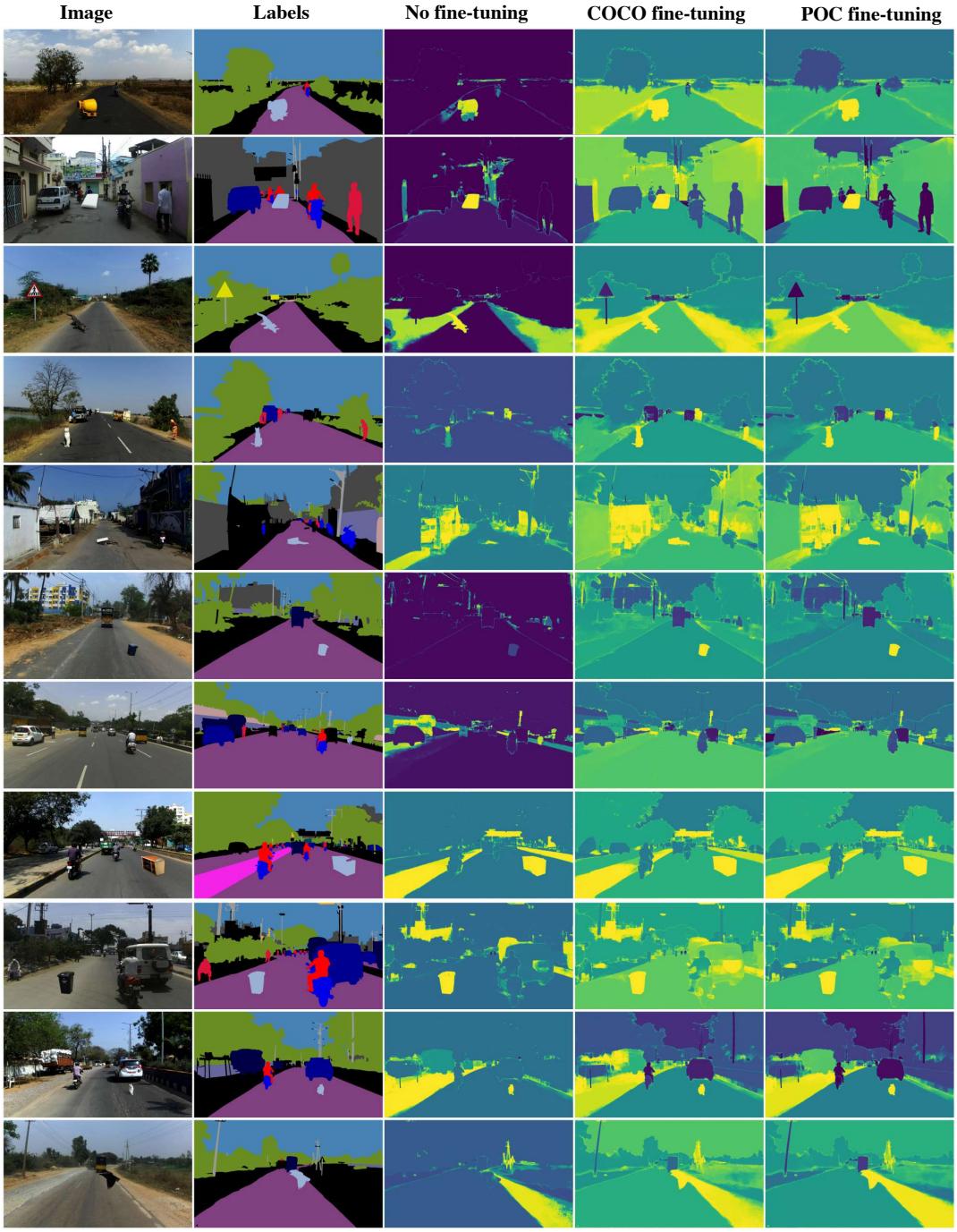

Figure 15: **M2A anomaly scores on IDD-POC samples.**

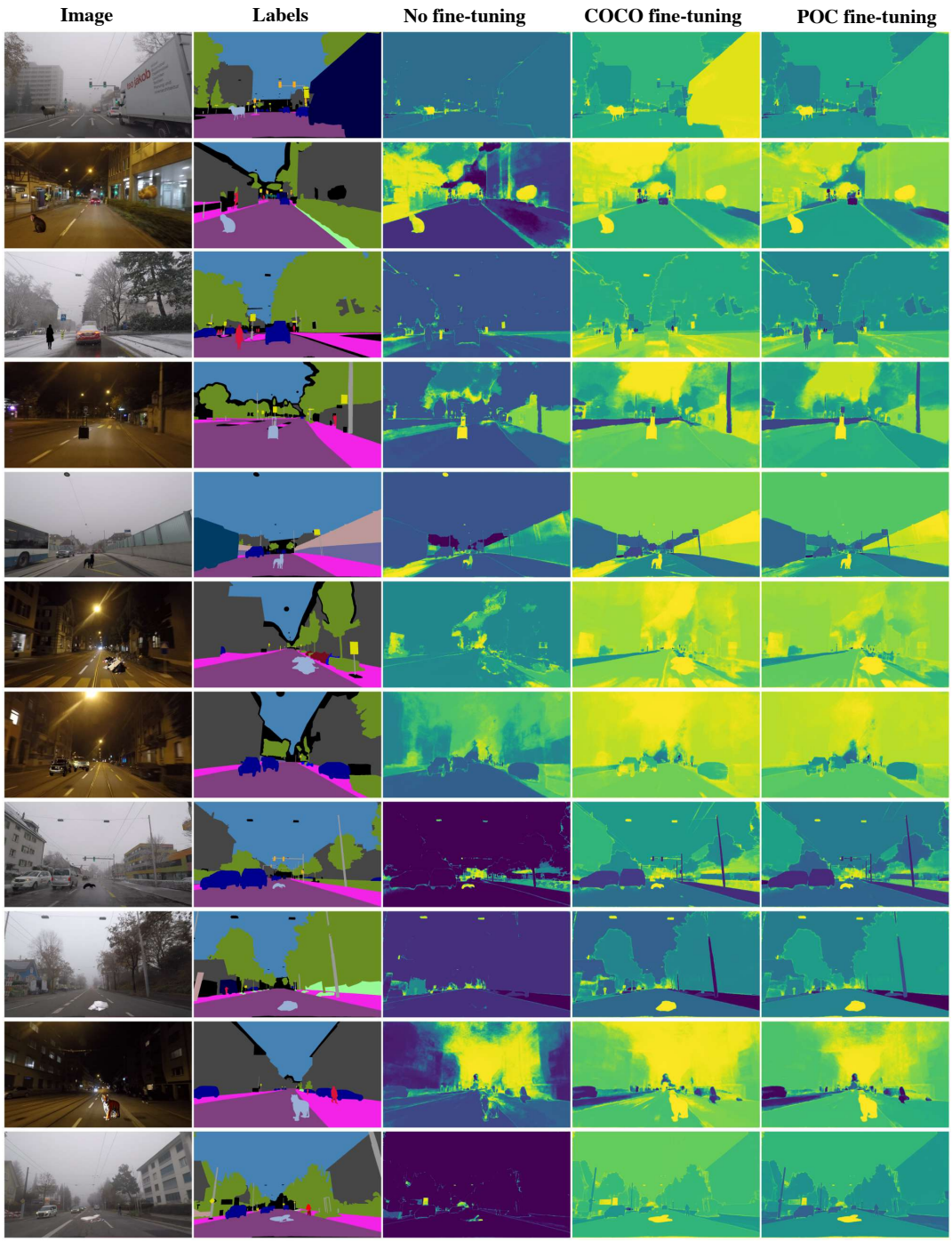

Figure 16: **M2A anomaly scores on ACDC-POC samples.**

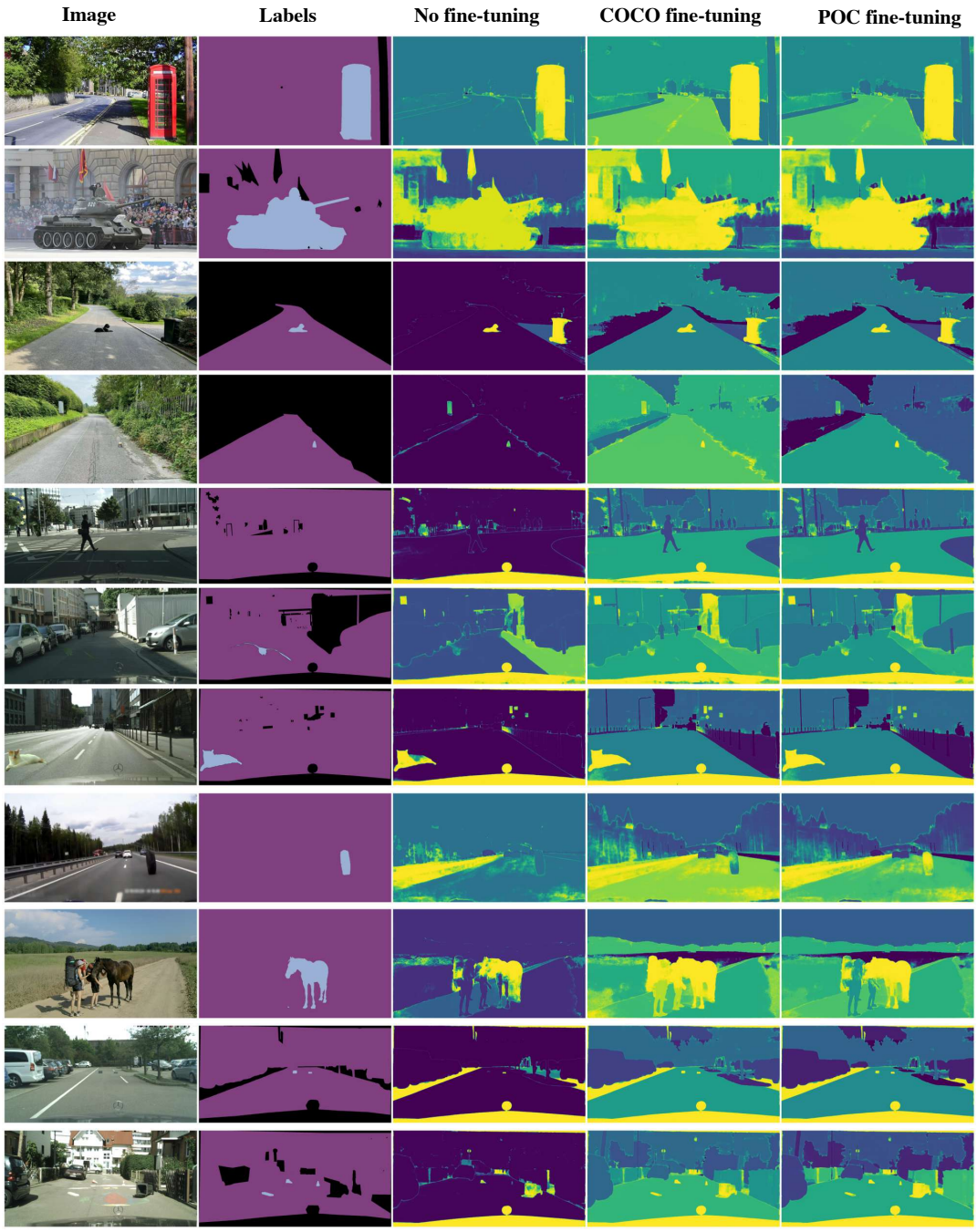

Figure 17: **M2A anomaly scores on samples from related datasets (see Fig. 2).**

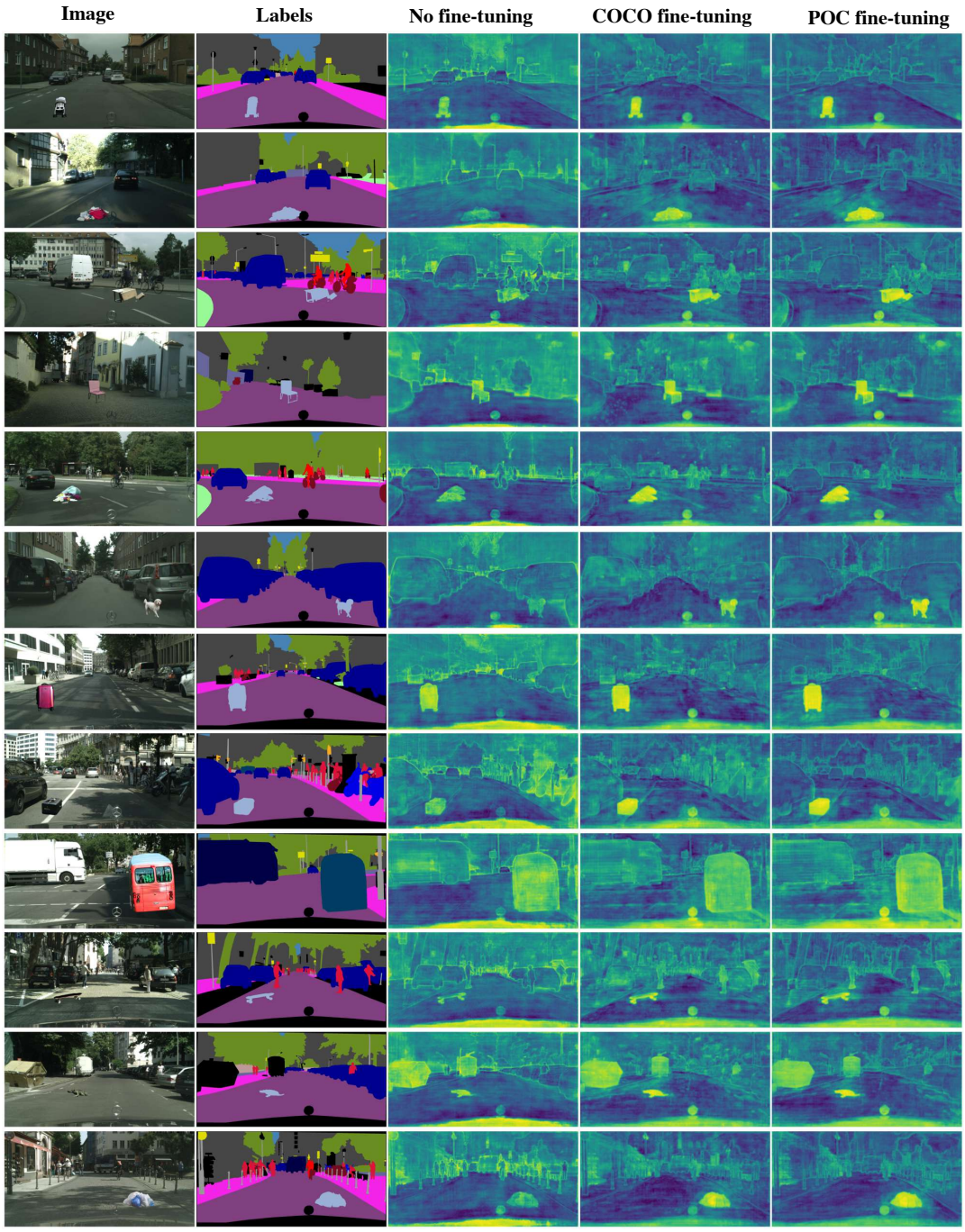

Figure 18: **RPL anomaly scores on CS-POC samples.**

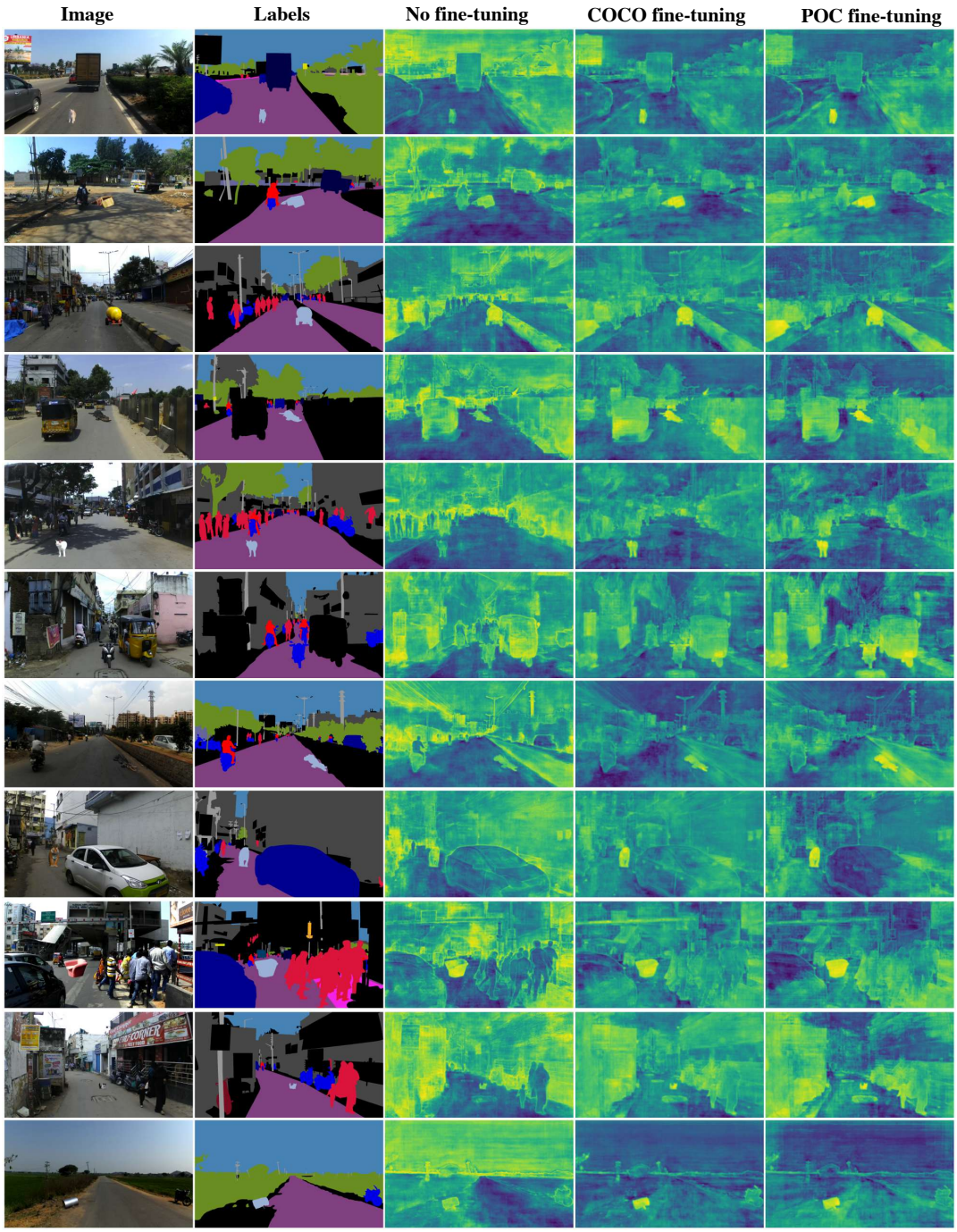

Figure 19: **RPL anomaly scores on IDD-POC samples.**

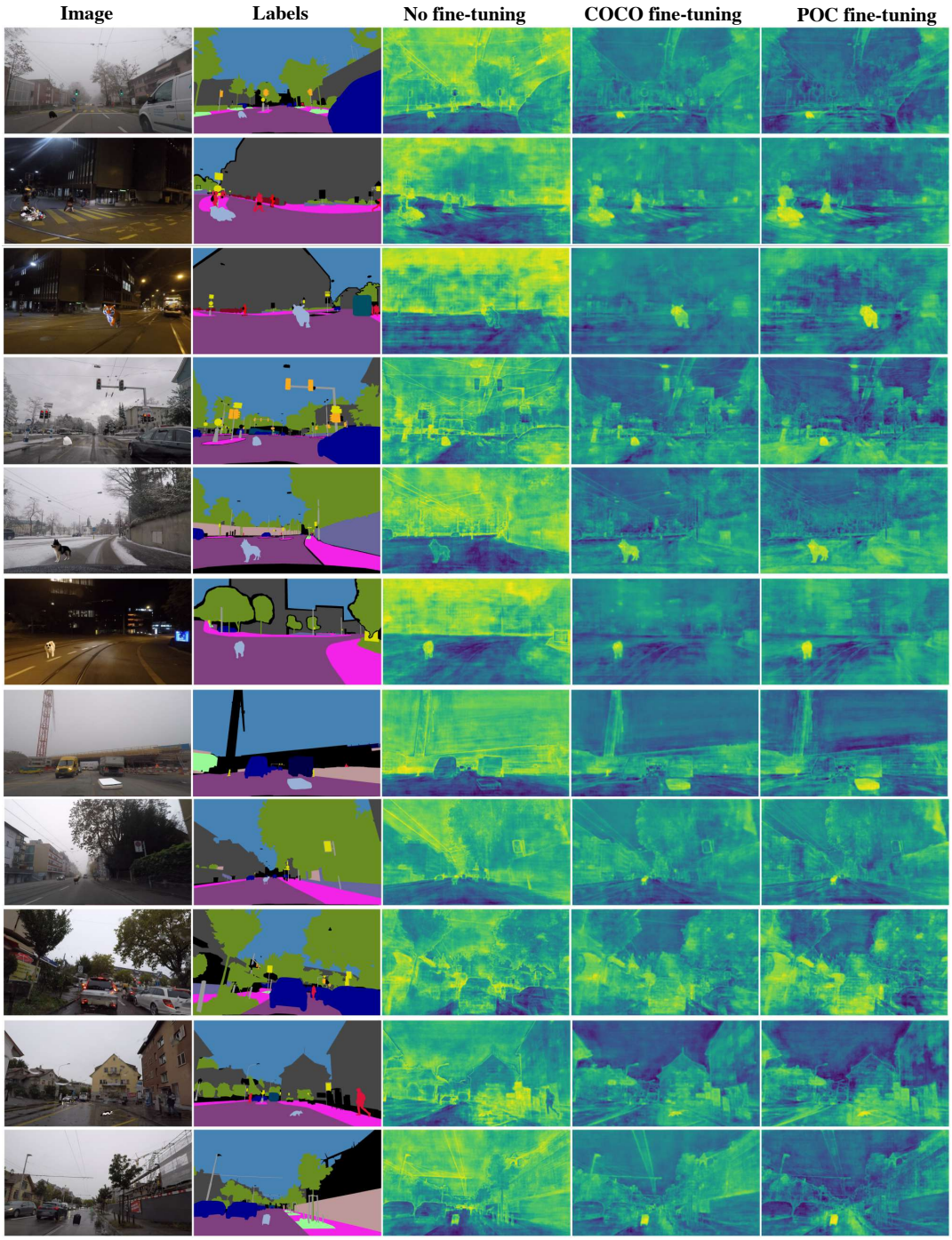

Figure 20: **RPL anomaly scores on ACDC-POC samples.**

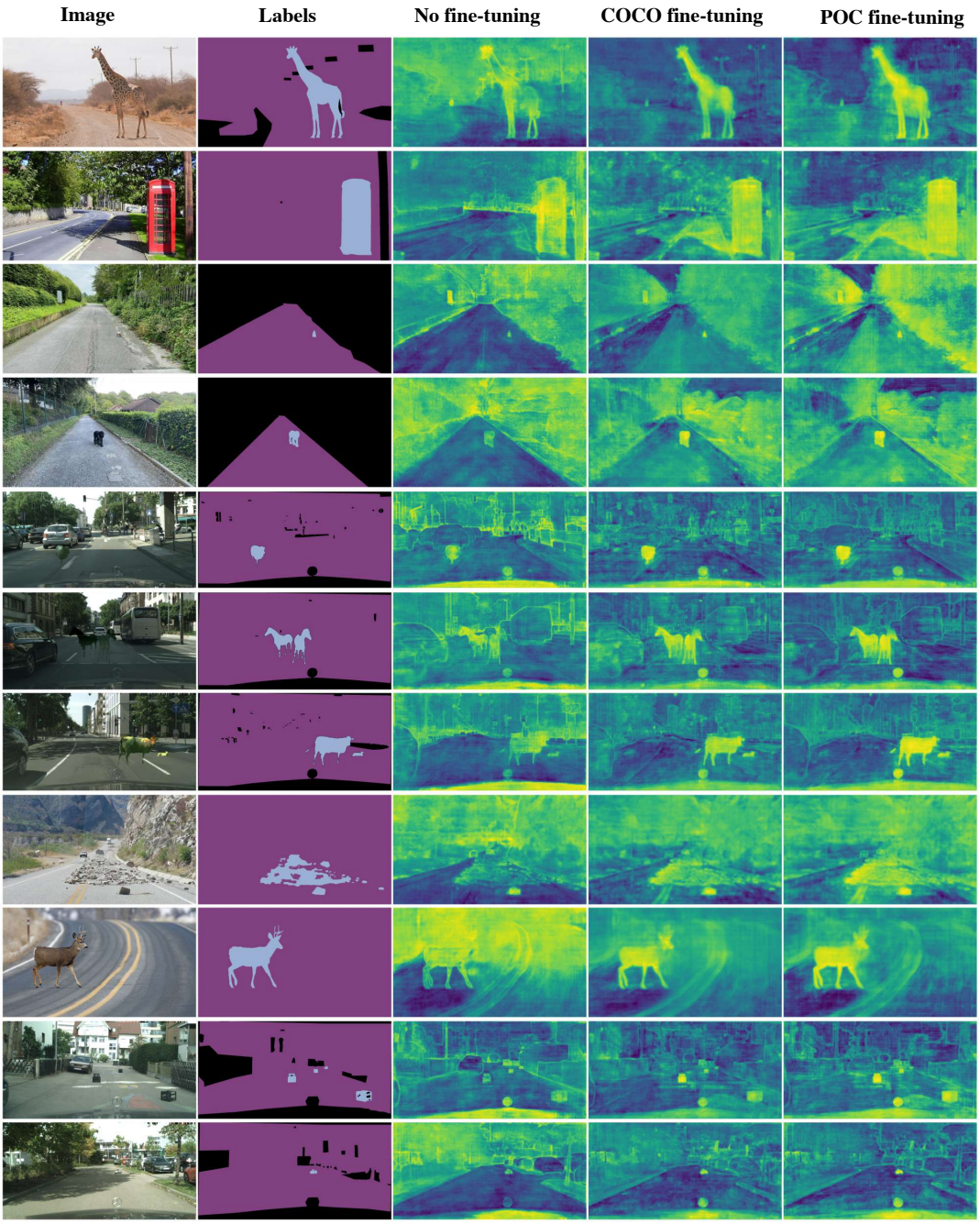

Figure 21: **RPL anomaly scores on samples from related datasets (see Fig. 2).**

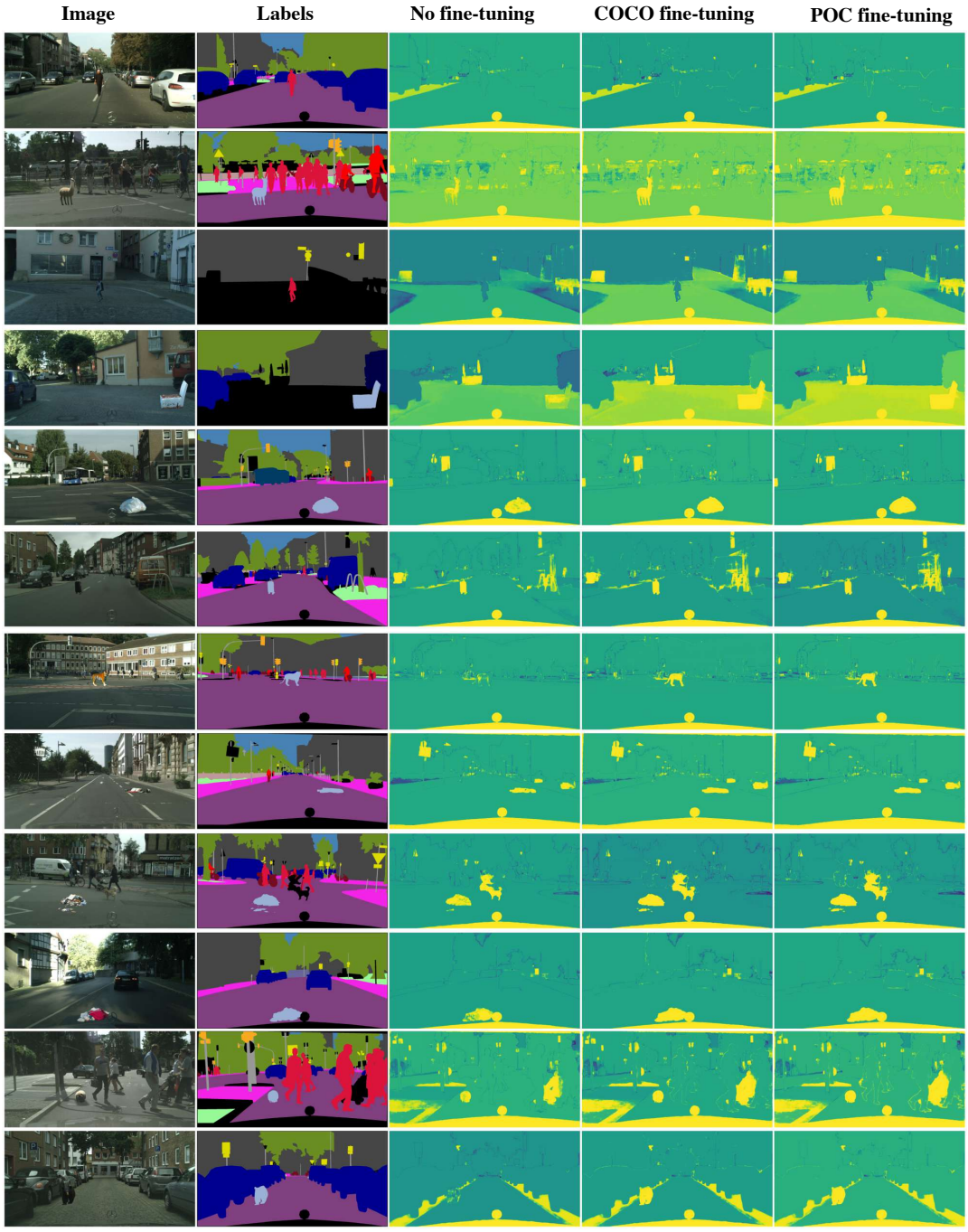

Figure 22: **RbA anomaly scores on CS-POC samples.**

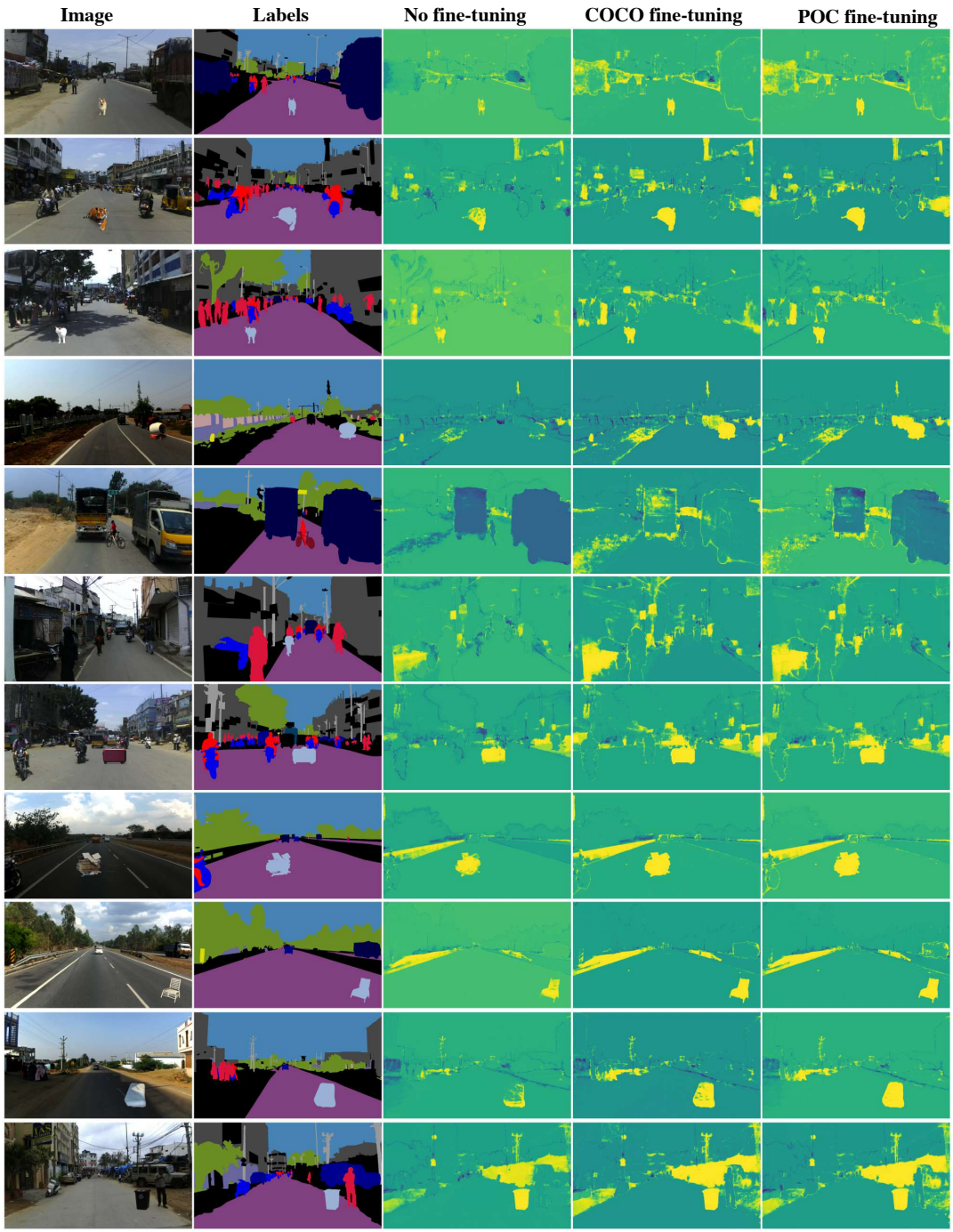

Figure 23: **RbA anomaly scores on IDD-POC samples.**

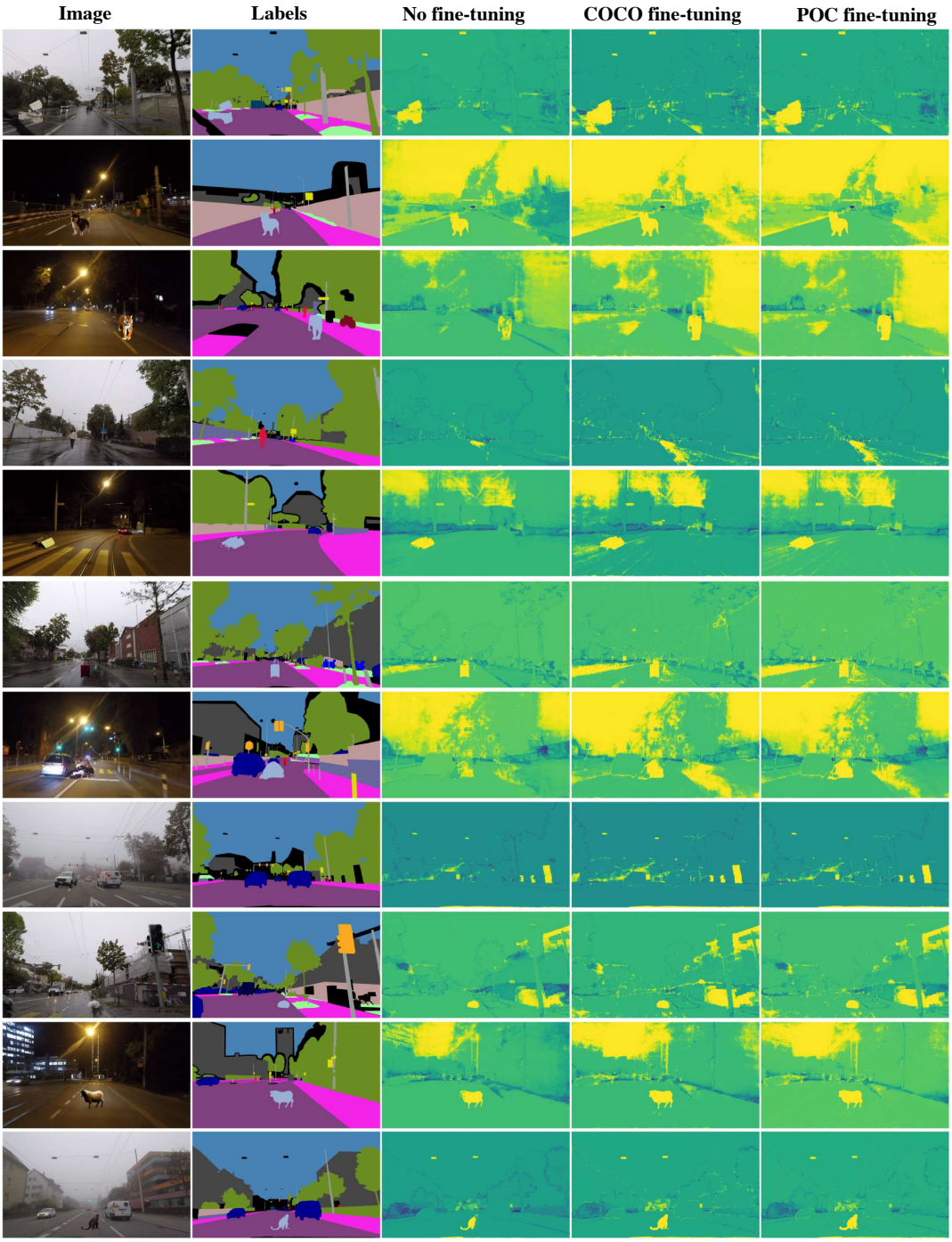

Figure 24: **RbA anomaly scores on ACDC-POC samples.**

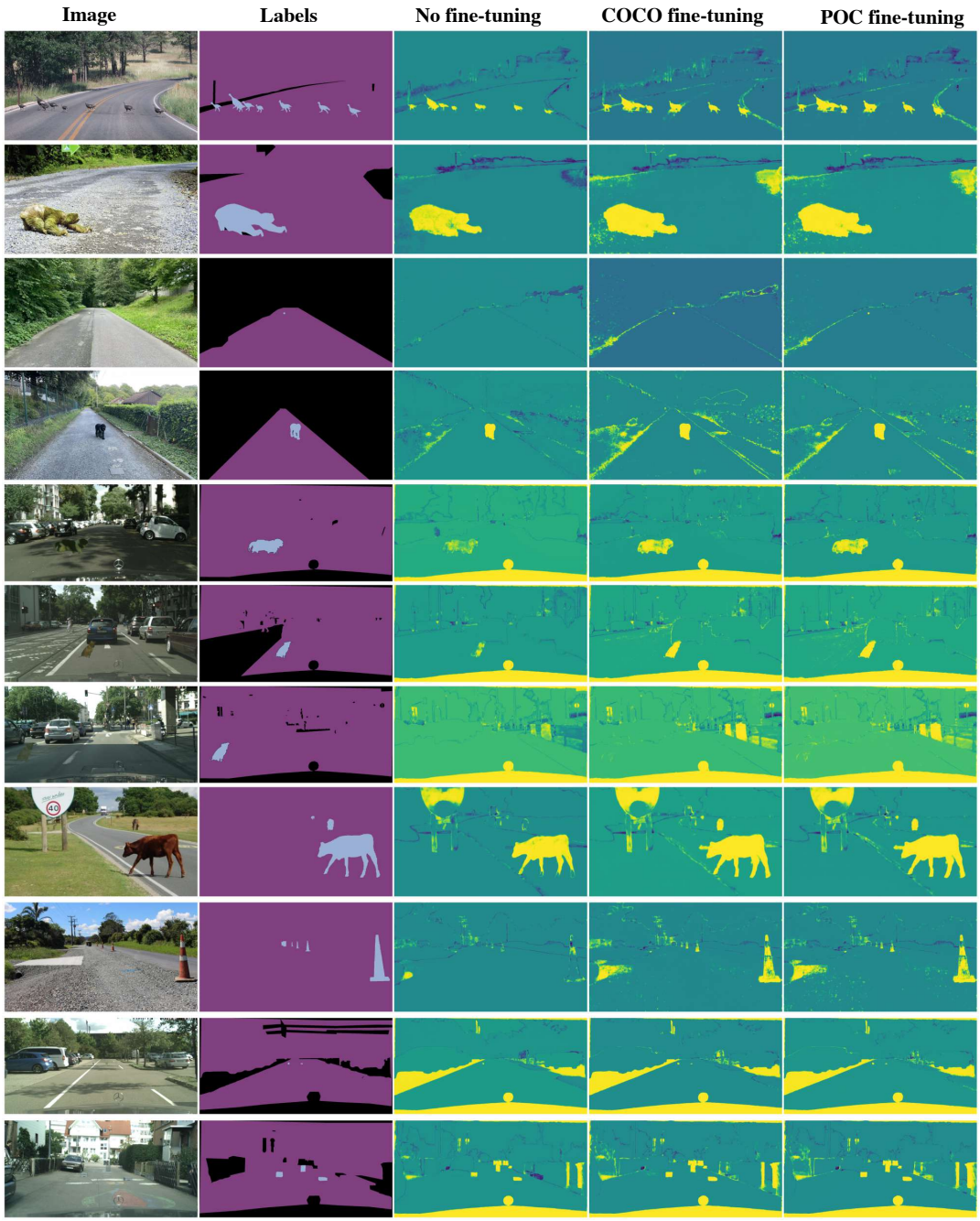

Figure 25: **RbA anomaly scores on samples from related datasets (see Fig. 2).**

J. Additional qualitative results

In this section we show additional qualitative results for the dataset extension experiments (*c.f.* Sec. 5). We show predictions on all evaluated datasets for DLV3+, ConvNeXt and Segmenter models.

Figure 26: **DLV3+ predictions on additional web images.**

Figure 27: **ConvNeXt predictions on additional web images.**

Figure 28: **Segmenter predictions on additional web images.**

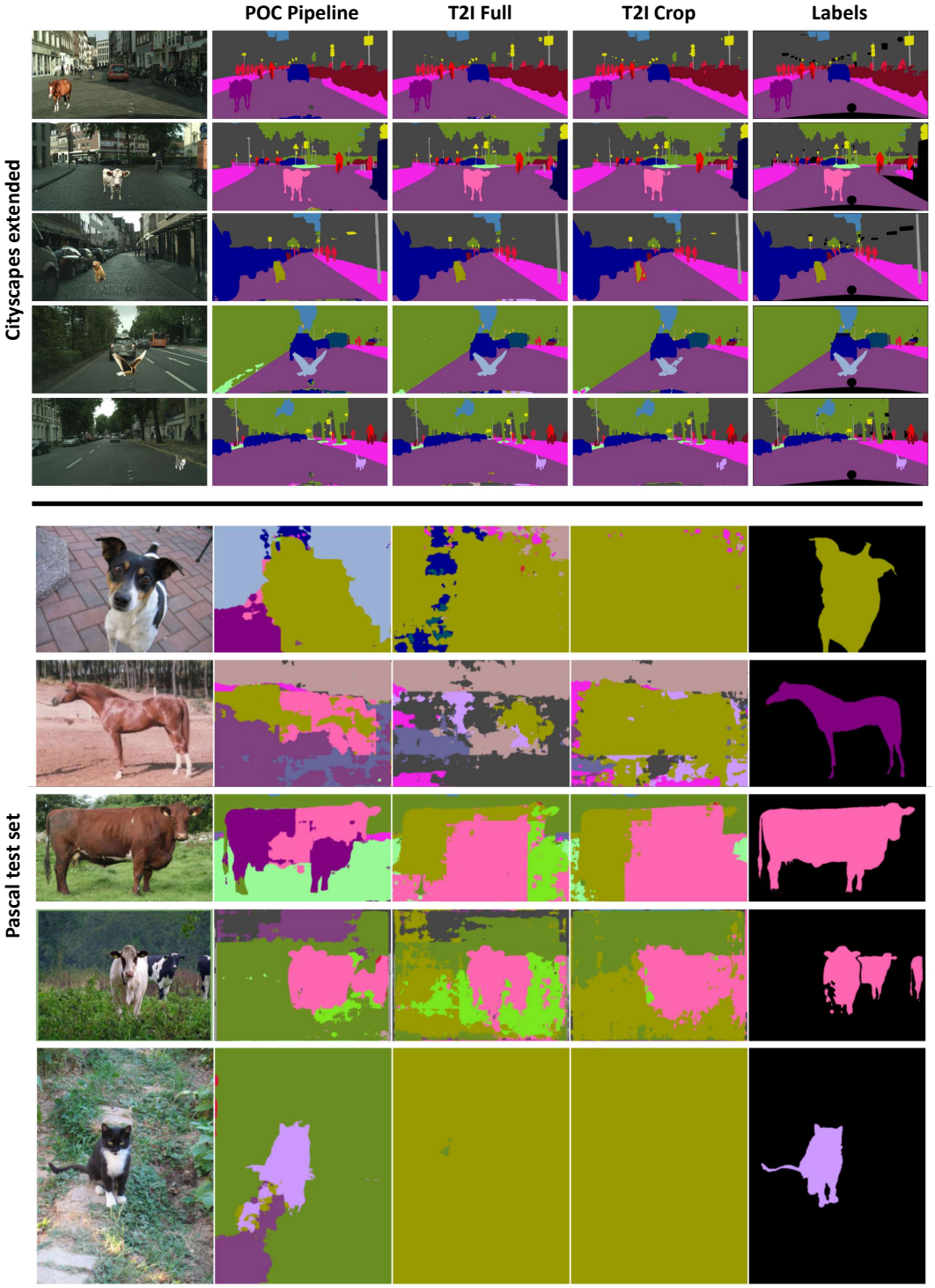

Figure 29: **DLV3+ predictions on extended Cityscapes (*POC A*) and Pascal validation sets.**

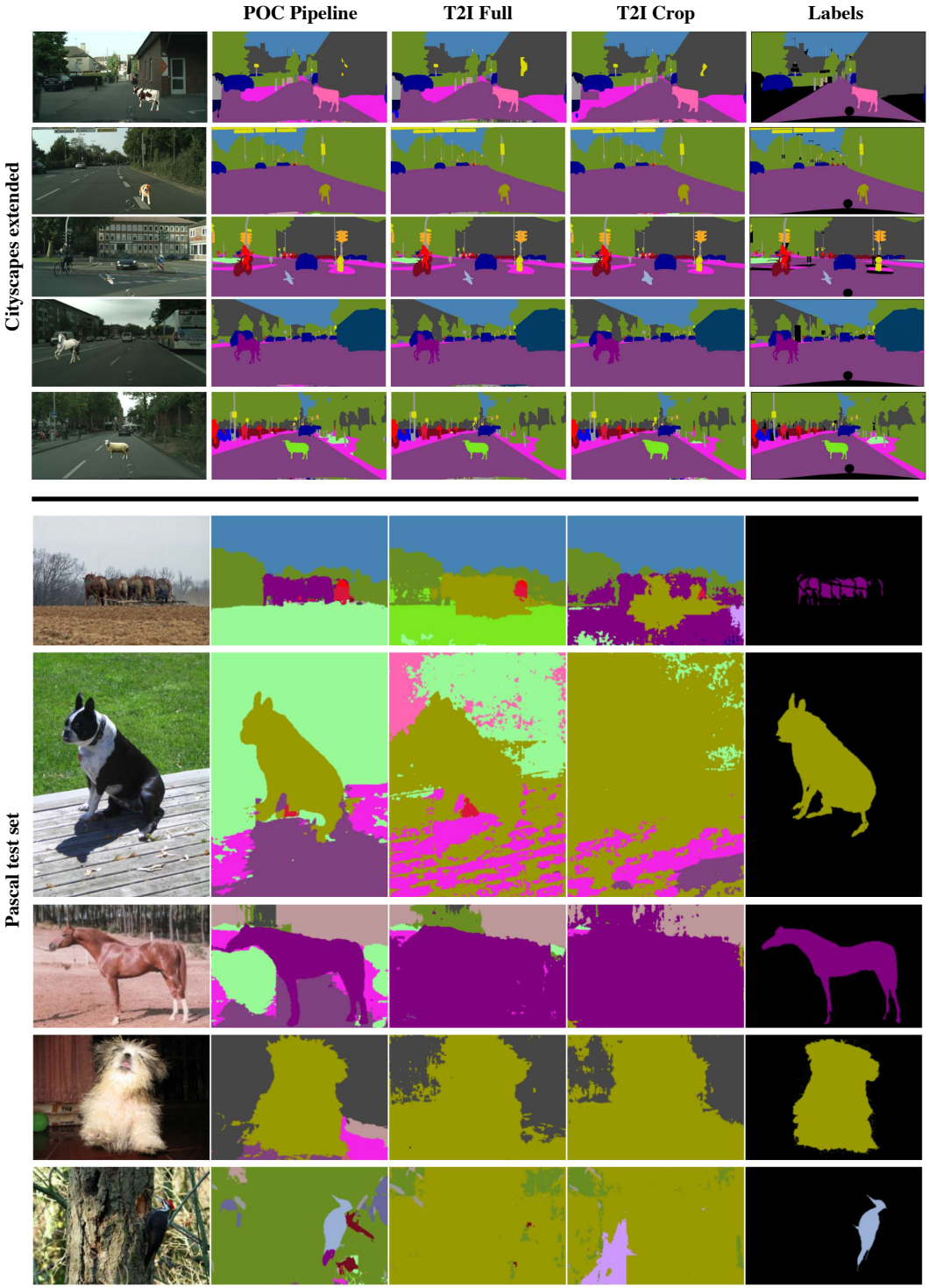

Figure 30: **ConvNeXt predictions on extended Cityscapes (*POC A*) and Pascal validation sets.**

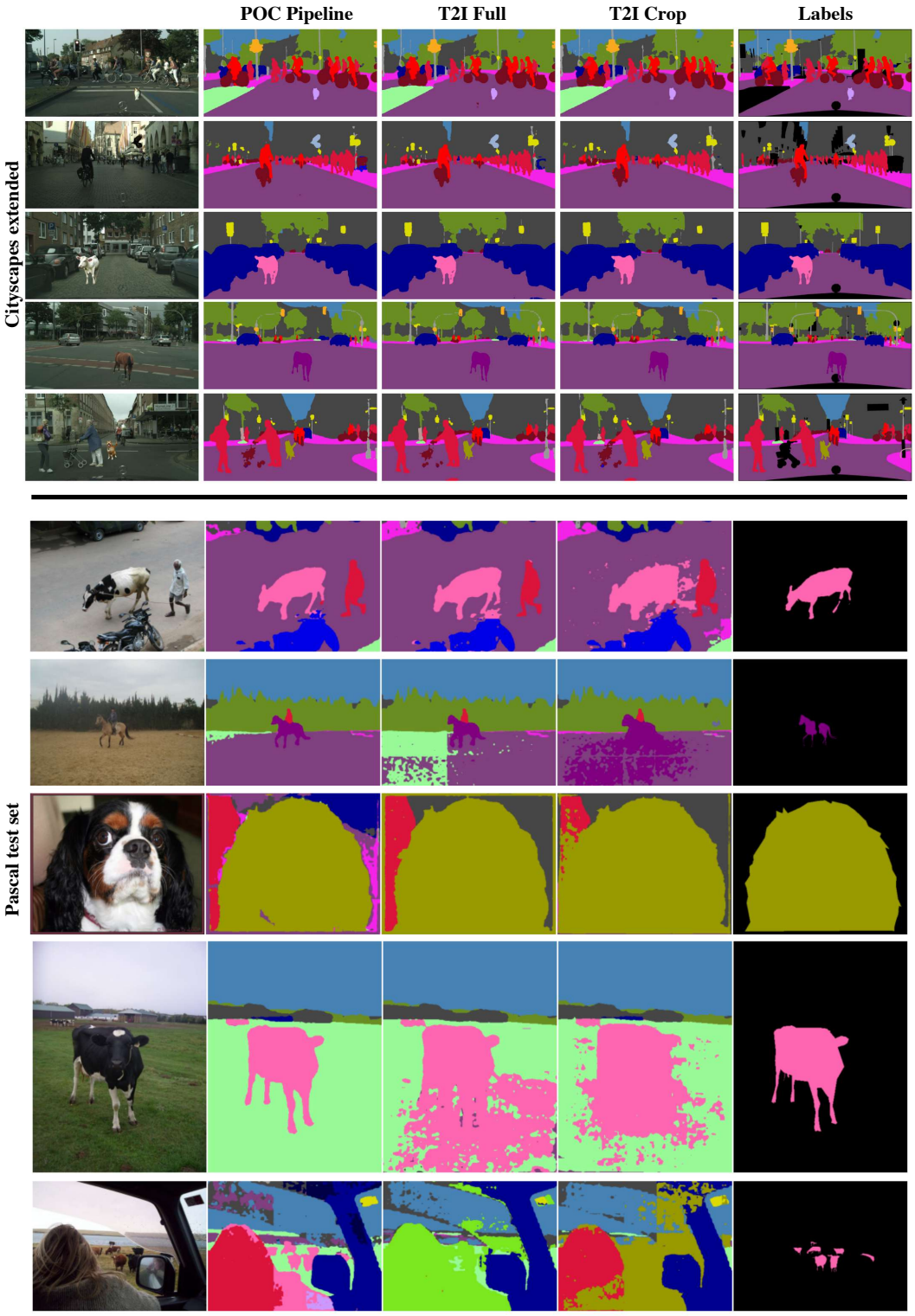

Figure 31: **Segmenter predictions on extended Cityscapes (*POC A*) and Pascal validation sets.**