Appendix

A. Maximized Entropy via the chosen Out-Distribution Loss Function

In section 3.1 we state that by our choice of out-distribution loss function we maximize the softmax entropy. As a reminder, the softmax entropy is defined as

$$E(f(x)) := -\sum_{j\in\mathcal{C}} f_j(x)\log(f_j(x)) \tag{9}$$

and the out-distribution loss function as

$$\ell_{out}(f(x)) := -\sum_{j\in\mathcal{C}} \frac{1}{|\mathcal{C}|}\log(f_j(x))\,, \tag{10}$$

where $\mathcal{C}$ denotes the the set of trained classes and $f(x) \in (0,1)^{|\mathcal{C}|}$ the softmax probability vector. Then, *minimizing* $\ell_{out}(f(x))$ is equivalent to *maximizing* the softmax entropy $E(f(x))$. This statement can be proven straightforwardly using Jensen's inequality. Since the softmax definition implies $f_j(x) \in (0,1)\ \forall j \in \mathcal{C}$ and $\sum_{j\in\mathcal{C}} f_j(x) = 1$, Jensen's inequality applied to the convex function $-\log(\cdot)$ yields

$$\ell_{out}(f(x)) \geq -\log\left(\sum_{j\in\mathcal{C}} \frac{1}{|\mathcal{C}|} f_j(x)\right) = \log(|\mathcal{C}|) \tag{11}$$

and applied to the concave function $\log(\cdot)$

$$E(f(x)) \leq \log\left(\sum_{j\in\mathcal{C}} f_j(x)\frac{1}{f_j(x)}\right) = \log(|\mathcal{C}|) \tag{12}$$

with equality if $f_j(x) = \frac{1}{|\mathcal{C}|}\ \forall j \in \mathcal{C}$.

B. Separability by means of Data Distribution

The violin plots in figure 4 visualize the separability of in-distribution and out-distribution pixels (binary classification) in LostAndFound and Fishyscapes, respectively. These plots summarize different statistics such as median and interquartile ranges and also show the full distribution of the data. The density corresponds to the relative pixel frequency at a given entropy value of the considered class. In the following, we refer to the shape of the violin plots as distribution.

First, we focus on evaluating LostAndFound OoD objects, see figure 4 (a). For the baseline model we observe that a large mass of data corresponding to the negative class is located at very low entropy values (median 0.02), *i.e.*, most road pixels are classified with high confidence. Moreover, the 75th percentile is located at an entropy value of 0.04 and the sample of highest value at 0.57. Regarding the pixels of the positive class, we see that the distribution is rather dispersed. The median is at 0.29 and the interquartiles range from 0.13 to 0.44. We conclude that, on average, positive samples have higher entropy values than negative ones, *i.e.*, pixels of an OoD object are classified with higher uncertainty than for road pixels. However, for perfect performance one seeks a threshold such that both distributions (of the positive and negative class) are separated. This is not the case for the baseline model since a substantial amount of samples still has very low entropy, *e.g.* the 10th percentile of the positive samples is at 0.04, which is also the median of negative samples.

After OoD training, the distribution of negative samples remains in large parts similar compared to the baseline only with little changes. Noteworthy, the median and upper quartile decrease down to entropy values of 0.1 and 0.2, respectively. The distribution's maximum is at 0.66. On the contrary, the changes of the distribution for the positive samples are significant as a large mass is concentrated at very high entropy values. The median is located at 0.59 which is roughly at the same magnitude as the maximum for negative samples. Moreover, the minimum value for positive pixels is at 0.01 which equals the median for negative samples. In particular the latter underlines the significant improvement of separability due to our OoD training. We observe the same behavior for Fishyscapes OoD objects but even more pronounced, see figure 4 (b). After the OoD training, the medians of the two classes, 0.01 for negative samples and 0.87 for positive samples, differ by 86 percent points. Besides, the lower quartile of positive samples at an entropy value of 0.71 as well as the 1st percentile at 0.03 are still above the median of negative samples. Consequently, we conclude that our OoD training is beneficial for identifying OoD pixels.

C. Segment-wise Metrics for Meta Classifiers

As outlined in section 4, we train meta classifiers based on hand-crafted metrics. These metrics are derived from the softmax probabilities $f(x) \in (0,1)^{|\mathcal{Z}|\times|\mathcal{C}|}, x \in \mathcal{X}$ of deep convolutional neural networks, information we get in every forward pass. As a reminder, let $\hat{\mathcal{Z}}_{out}(x)$ be the set of pixel locations in image $x \in \mathcal{X}$ that are predicted to be OoD, see section 5. A connected component $k \in \hat{\mathcal{K}}(x) \subseteq \mathcal{P}(\hat{\mathcal{Z}}_{out}(x))$ represents an *OoD segment / object prediction* due to the entropy being above the given threshold. This is different to other works dealing with segment-wise meta classification [9, 34, 44, 45] as they consider connected components sharing the same class label as segments.

We estimate uncertainty per OoD segment k by averaging pixel-wise scores at the segment's pixel locations $z \in k$. In addition to the plain softmax probabilities $f^z(x)$, we also incorporate three pixel-wise dispersion measures, namely

$\forall z \in k$ the (normalized) entropy

$$\bar{E}(f^z(x)) = -\frac{1}{|\mathcal{C}|} \sum_{j \in \mathcal{C}} f_j^z(x) \log(f_j^z(x)) \,, \tag{13}$$

the variation ratio

$$V(f^z(x)) = 1 - f^z_{\hat{c}(z)}(x) \,, \tag{14}$$

and the probability margin

$$M(f^z(x)) = V(f^z(x)) + \max_{j \in \mathcal{C} \setminus \{\hat{c}(z)\}} f_j^z(x) \tag{15}$$

with $\hat{c}(z) := \arg\max_{j \in \mathcal{C}} f_j^z(x)$ being the class label according to the maximum a posteriori principle.

The segment's size $S(k) = |k|$ is not only needed for averaging but also serves as meta classification input on its own. Moreover, let $k_{in} \subset k$ be the set of pixel locations in the interior of the segment k, *i.e.*, $k_{in} = \{(h, w) \in k : [h \pm 1] \times [w \pm 1] \in k\}$. This also gives us the pixel locations of the boundary $k_{bd} = k \setminus k_{in}$. In order to capture geometry features of a segment, we consider the relative sizes

$$\tilde{S} = S/S_{bd} \;\text{ and }\; \tilde{S}_{in} = S_{in}/S_{bd} \tag{16}$$

by treating the segment's boundary and interior separately.

For all metrics outlined up to now, we additionally consider the variances of the pixel-wise scores. They measure the deviation from the segment score mean and in our experiments, and it turns out that they have a great impact on meta classifiers for OoD detection.

Let $k_{nb} = \{z' \in [h \pm 1] \times [w \pm 1] \subset |\mathcal{H}| \times |\mathcal{W}| : (h, w) \in k, z' \notin k\}, \mathcal{Z} = \mathcal{H} \times \mathcal{W}$ be the neighborhood of k. As metric if one segment is misplaced we include

$$N(j|k) = \frac{1}{|k_{nb}|} \sum_{z \in k_{nb}} \mathbb{1}_{\{j = \hat{c}(z)\}} \; \forall \, j \in \mathcal{C} \tag{17}$$

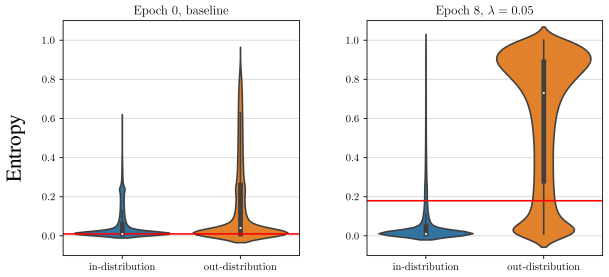
which is the proportion of neighborhood pixels, with class $j \in \mathcal{C}$ having the highest softmax score, to neighborhood size. Another metric for localization purposes is the segment's geometric center

$$C_h(k) = \frac{1}{S} \sum_{i=1}^{S} h_i \;\text{ and }\; C_w(k) = \frac{1}{S} \sum_{i=1}^{S} w_i \tag{18}$$

with $z_i = (h_i, w_i) \in k \; \forall \, i = 1, \ldots, |k|$, *i.e.*, averaging over the segment's pixel coordinates in vertical and horizontal direction.

For each segment k we then have $m = 83$ metrics in total (as $|\mathcal{C}| = 19$ in our experiments). This forms a structured dataset

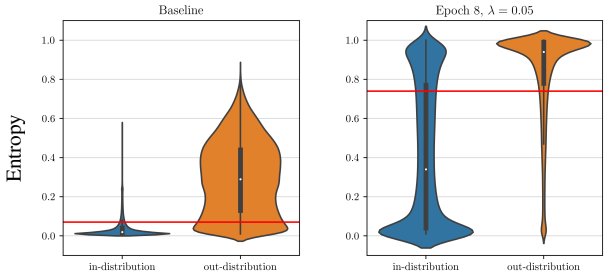
$$\mu \subseteq \mathbb{R}^{|\cup_{x \in \mathcal{X}} \hat{\mathcal{K}}(x)| \times m} = \mathbb{R}^{|\cup_{x \in \mathcal{X}} \hat{\mathcal{K}}(x)| \times 83} \tag{19}$$



Baseline performance | Cityscapes void OoD training

(Detecting Cityscapes unlabeled objects)

Figure 10: Separability between in-distribution and out-distribution pixels in Cityscapes. Pixels labeled as train class according to the ground truth are considered as in-distribution, pixels labeled with the void class as out-of-distribution. For the results with Cityscapes void OoD training the baseline model (left) was retrained with entropy maximization on the Cityscapes void class (right), *i.e.*, using Cityscapes unlabeled objects as OoD proxy for $\mathcal{D}_{out}$.



Baseline performance | Cityscapes void OoD training

(Detecting LostAndFound OoD objects)

Figure 11: Separability between in-distribution and out-of-distribution pixels in the OoD dataset LostAndFound. For the results with Cityscapes void OoD training the baseline model (left) was retrained with entropy maximization on the Cityscapes void class (right).

serving as input for the meta classification model $g : \mu \to [0, 1]$, the latter being a simple logistic regression in our case. By means of this linear model, we learn to discriminate whether a segment k has an intersection with the ground truth (while all inputs are independent of the ground truth segmentation), see also equation (7).

D. OoD Training with Cityscapes void Class

Before using the COCO dataset as OoD proxy, we conducted some experiments with the Cityscapes void class as

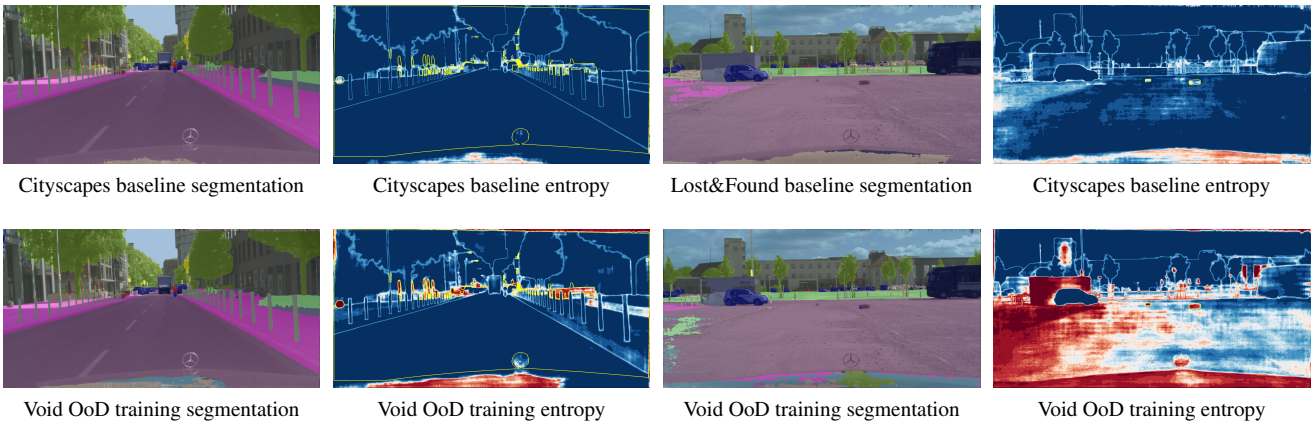


Figure 12: Comparison between baseline model and retrained model, with entropy maximization on Cityscapes unlabeled objects, for one Cityscapes and one LostAndFound scene. The first and third column displays the segmentations obtained by the respective models on either a Cityscapes or LostAndFound input image, the second and forth column displays the corresponding entropy heatmaps. In the entropy heatmaps, the OoD objects are marked with yellow lines.

OoD proxy for $\mathcal{D}_{out}$ in order to perform entropy maximization. This class includes objects that cannot be assigned to any of the Cityscapes training classes, therefore they remain unlabeled and are ignored during training. We refer to this retraining approach using the Cityscapes unlabeled objects as OoD proxy as *void OoD training*. We find the best results in our experiments for the DeepLabv3+ as baseline model after 8 epochs of void OoD training and out-distribution loss weight of $\lambda = 0.05$. With respect to the Cityscapes validation dataset, the retrained model clearly improves at identifying unseen unlabeled objects, see figure 10.

However, the same retrained model fails to generalize to unseen OoD objects available in the LostAndFound dataset, see figure 11. Not only the softmax entropy of OoD pixels is boosted but also the entropy of a significant amount of in-distribution pixels. This is even more considerable due to the strong class imbalance in LostAndFound. With respect to the AUROC, the void OoD training decreases the OoD detection score by 5 percent points down to 0.88, while decreasing the more relevant metric AUPRC by even 29 percent points down to 0.17 compared to the baseline model.

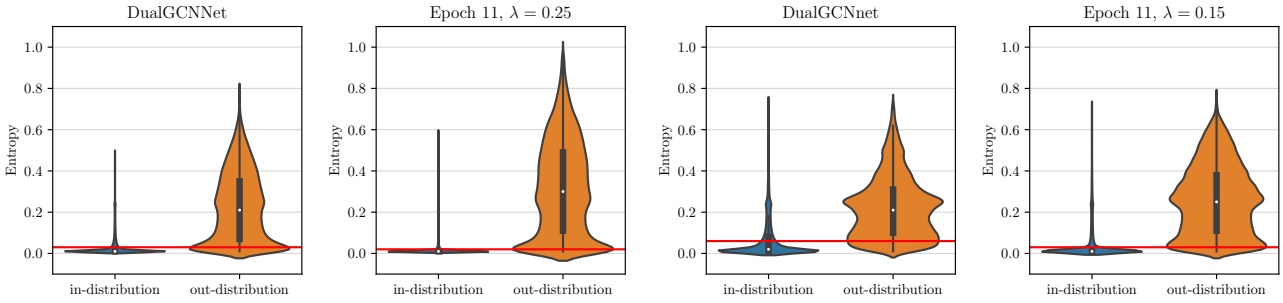
A visual comparison of the effects of void OoD training is shown is figure 12. The retraining does not noticeably impact the segmentation performance, neither for Cityscapes nor LostAndFound. In particular for the segmentation of the Cityscapes scenes, there are only minor differences visible, *i.e.*, the difference in performance for the original task is marginal. This is in line with the observation that retraining with the multi-criteria loss function, see equation equation (2), and the COCO dataset as OoD proxy leads only to a marginal loss of mIoU for the Cityscapes validation dataset. With respect to the Cityscapes images, the softmax entropy inside unlabeled objects is clearly boosted due to void OoD training. This makes identifying such objects easier in comparison to the baseline model.

Regarding the LostAndFound the differences in segmentations are more visible although still not being significant. On the contrary, by comparing the entropy heatmaps for the baseline model and the model after void OoD training, one observes that not only the entropy of pixels inside the OoD objects is boosted but also many in-distribution pixels. This detrimentally impacts the discrimination performance between in-distribution and out-distribution pixels as these two classes cannot be separated well via entropy thresholding. This supports the impression of the pixel-wise evaluation that void OoD training is not suitable for the detection of objects other than the Cityscapes unlabeled objects.

E. OoD Training for DualGCNNet

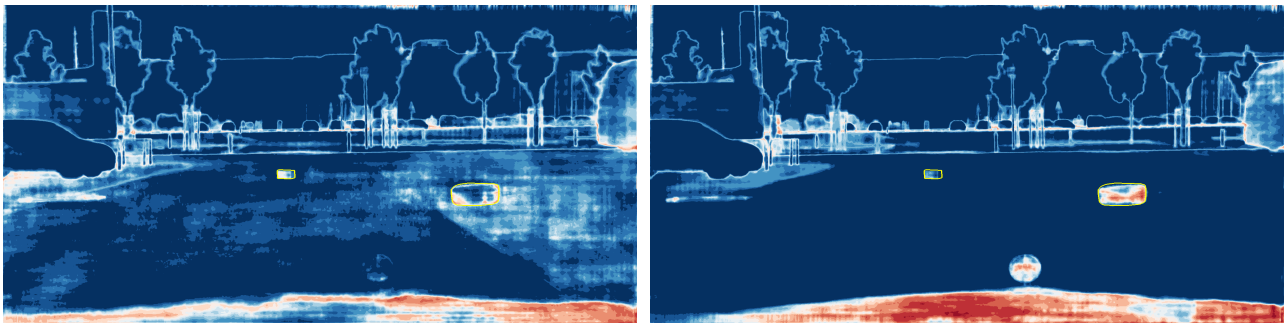
As a second model complementary to the DeepLabv3+ model, we conducted same experiments of OoD training, *i.e.*, retraining with the COCO dataset as OoD proxy, with the DualGCNNet which is a weaker and more lightweight network compared to the state-of-the-art DeepLabv3+ segmentation network. We find the best results after 11 epochs of OoD training with out-distribution loss weight of $\lambda = 0.25$. As optimizer we used Adam with a learning rate of 10^{-6}.

We evaluated the OoD detection for the LostAndFound test and Fishyscapes Validation dataset in a similar manner as in the experiments for DeepLabv3+. For the DualGCNNet model, however, we only compare OoD training against entropy thresholding with the original model. Entropy thresholding with DualGCNNet in its original version is a weak OoD detector with AUPRC-scores of 0.36 and



(a) LostAndFound (b) Fishyscapes

Figure 13: Relative pixel frequencies of LostAndFound (a) and Fishyscapes (b) OoD pixels, respectively, at different entropy values for the baseline model, *i.e.*, before OoD training (*a & b left*), and after OoD training (*a & b right*). The red lines indicate the thresholds of highest accuracy.



Before OoD Training After OoD training

Figure 14: Comparison of softmax entropy heatmaps before (left) and after OoD training (right). The yellow lines mark the OoD objects according to their ground truth annotation.

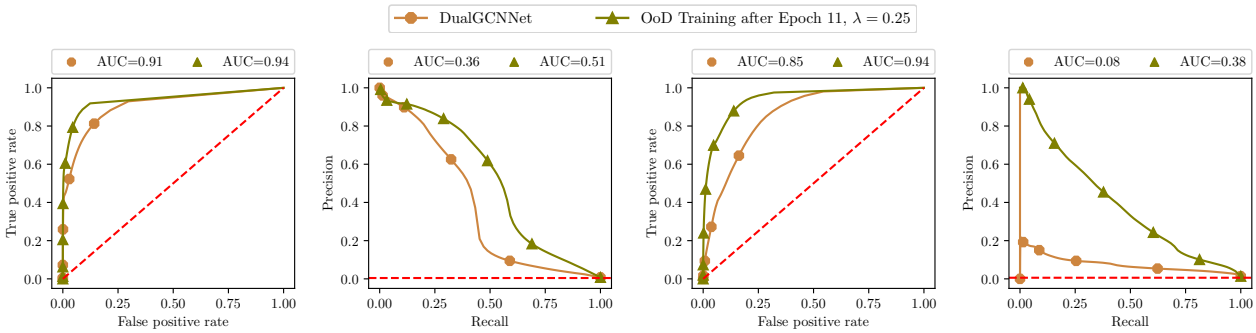
0.07 for LostAndFound and Fishyscapes, respectively, see table 1. We observe that OoD training is not as effective as for the DeepLabv3+ model in terms of absolute performance gain. However, we still observe a decent improvement in separability. By applying OoD training, the AUROC increases by 3 percent points for LostAndFound and even 9 percent points for Fishyscapes up to a score of 0.94 for both datasets. With respect to the PR curves, the AUC improves by 15 percent points up to 0.51 for LostAndFound and by 20 percent points up to 0.38 for Fishyscapes. Noteworthy, these AUC scores after OoD training are higher than for the plain DeepLabv3+ (baseline) model which is already a strong OoD detection model.

These results for the weaker DualGCNNet model further demonstrate the positive effect on the OoD detection ability when performing OoD training with the COCO dataset as OoD proxy. The pixel-wise evaluation results are reported by means of the violin plots in figure 13 and by ROC as well as PR curves in figure 15.

F. OoD Training Visualization

The improved separation ability due to OoD training is not only achieved by increasing the softmax entropy of OoD pixels but also by decreasing the softmax entropy for in-distribution pixels. This can be also observed by means of the in-distribution violins, for instance in figure 13. By comparing the shapes of the violins corresponding to the DualGCNNet plain model and the model after OoD training, we notice that the violin shapes remain similar in large parts. The median and the upper quartile, however, decrease down to lower entropy values after OoD training. This indicates that after entropy maximization the model is on the one hand more uncertain at OoD pixel locations and on the other hand more certain about its prediction at in-distribution pixel locations. The same observation also holds for the DeepLabv3+ model, see figure 4. This is in line with the observation made in [49] that training with an OoD proxy may have a regularizing effect.

An illustration is provided in figure 14. For comparison purposes, we refer to the entropy heatmaps provided



(a) LostAndFound (*left:* AUROC, *right:* AUPRC)

(b) Fishyscapes (*left:* AUROC, *right:* AUPRC)

Figure 15: Detection ability of LostAndFound (a) and Fishyscapes (b) OoD pixels, respectively, evaluated by means of receiver operating characteristic curve (*a & b left*) and precision recall curve (*a & b right*). The red lines indicate the performance according to random guessing.

in figure 12 as both figures show the same scene. The visualization of heatmaps clearly shows that due to OoD training pixels with high entropy are more concentrated inside OoD objects. Moreover, the in-distribution objects, especially the pixels corresponding to the road, have lower entropy values than before OoD training. This makes the road seem cleaner with respect to the possible occurrence of OoD objects. After entropy maximization the OoD objects are (visibly) better recognizable within the softmax entropy heatmaps. Therefore, we expect that the meta classification performance is improved as the meta classifiers are able to estimate the shape of OoD objects even better. Moreover, higher entropy values are stronger correlated with the presence of OoD objects.

G. Found Objects due to OoD Training

The objects in the LostAndFound dataset comprises four road harzard classes:

- *humans*: kids (with toys) on the road
- *standard object*: crates in different shapes and colors
- *emotional hazards*: bobby car, ball, dog, *etc*.
- *random hazards*: bumper, euro pallet, pylon, tire *etc*.

Figure 16 illustrates the found objects via softmax entropy thresholding with the baseline model and also the model after OoD Training. One clearly notices that the number of overlooked OoD objects from all classes is significantly reduced due to OoD training.

H. Course of OoD Training

In order to monitor that the baseline model does not unlearn its original task due to OoD training, we evaluate the model's original task performance over the training epochs.

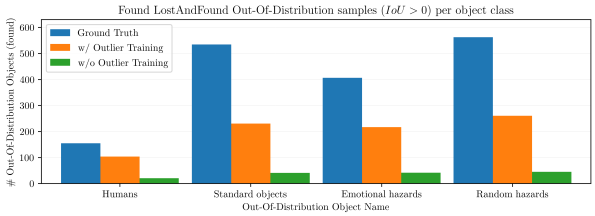


Figure 16: Overview of detected LostAndFound objects (per object class) with $t = 0.7$ before (green) and after OoD Training (orange). The blue bar indicates the number of ground truth instances that can be found in total.

We evaluate the mIoU on the Cityscapes validation dataset against the AUPRC on the LostAndFound test dataset, displayed in figure 17. The state-of-the-art DeepLabv3+ model, which serves as baseline throughout our experiments, achieves an mIoU of 90.30% when equipped only with the standard maximum a posteriori (MAP) decision principle while the same model has an entropy based OoD detection performance of 46.01% in AUPRC. By fine tuning the neural network with entropy maximization on OoD inputs, we on the one hand sacrifice only little in mIoU (of the original task). On the other hand, we observe improved AUPRC scores over the course of training epochs peaking at 76.45%. This considerable gain at detecting OoD samples in LostAndFound comes with a marginal loss in Cityscapes validation mIoU of less than 1 percent point. Moreover, the course of the OoD training illustrates convergence around the best AUPRC score with an mIoU loss that is in the same range as for the best score after OoD training. Concerning the overall performance of perception systems that rely on semantic segmentation, *e.g.*, in applications like automated driving, this is a favorable trade-off in terms of safety that comes with very little computational overhead.

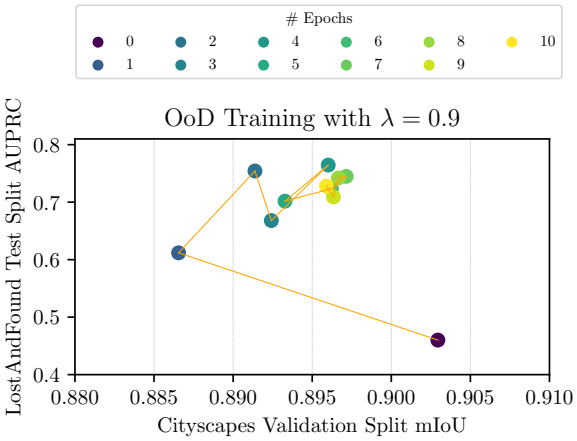


Figure 17: Mean intersection over union (mIoU) for the Cityscapes validation dataset split over the course of OoD training.

I. Meta Classification Visualization

The logistic regressions as meta classifiers have proven their efficiency in identifying and afterwards removing false positive (FP) / incorrect OoD object predictions. In this section we intend to show further examples for the FP OoD removal and thus show the final output of our two-step procedure for OoD detection.

For the plain model the meta classifiers are already able to remove FP OoD predictions reliably, see figure 18 top row. However, some false positive OoD predictions still remain. As pixels with high entropy are more concentrated inside OoD objects after the entropy maximization of the OoD training, the combination of OoD training and meta classification yields the best result in terms of the number of FP OoD predictions, see figure 18 bottom row. The examples in figure 19 further illustrate that the improved OoD detection performance after OoD training can even be enhanced by employing meta classifiers. The removed FP OoD predictions are rather small. However, we already consider one single pixel as FP OoD object prediction if that pixel is incorrectly predicted to be OoD. One could also define an OoD prediction to have a minimum amount of pixels. As our main focus is the reduction of overlooked OoD objects, we stick to the definition of equation (6) and consider an OoD object to be found if at least one pixel of that object is correctly classified as OoD. Therefore, small OoD segments are also fed through the meta classification model. Our two step method, consisting of entropy maximization and meta classification, extends segmentation networks by an improved OoD detection capability and unites both tasks in one model.

J. Meta Classification Feature Analysis

Least angle regression (LARS) is a model selection algorithm. We use this algorithm to select the linear model that fits best the meta classification responses $g(\mu)$ subject to an L_1 penalty term for the model coefficients $\beta \in \mathbb{R}^m = \mathbb{R}^{83}$, *c.f.* equation (19). LARS identifies the variables most correlated with the response. This selection method starts with all coefficients $\beta_1 = \ldots = \beta_p = 0$. In each step, one variable at a time is added to the set of active predictor variables. Thus, LARS adds the best feature $\mu_i, i = 1, \ldots, m$ to include in the active set, *i.e.* features show better correlation with the responses the earlier they are added. The coefficient of the added feature variable is continuously moved from 0 to its least squares coefficient until another variable μ_j has as much correlation with the response. This procedure is then repeated with the coefficients of the features in the active set, now including μ_j, until the active set reaches a predefined size.

Figure 20 visualizes the coefficients paths in the LARS algorithm for meta classification metrics. The correlation and therefore the impact of metrics are compared when the underlying segmentation CNN is in its plain version and also when OoD training is applied. As entropy threshold $t = 0.3$ is chosen as this gives us the best linear models for both meta classification cases with an average precision of 98.84 and 99.53, respectively, see figure 3.

We observe that in general features become active later when OoD training is applied, *i.e.* $||\beta||_1 / \max ||\beta||_1$ is greater when the i-th variable is added to the active set. This implies that the hand-crafted metrics have higher correlations with the meta classification response and therefore have greater impact on the meta classification performance. After OoD training the entropy E has become the most important metric. The entropy and its variance when restricted to the interior of a segment are among the first ten active variables as well (the eighth and ninth active variable). Without OoD training the softmax probability for the road has the highest correlation, with the entropy metric being the seventh feature in the active set. This analysis shows that the entropy boost due to OoD training has a positive effect on the meta classification performance for entropy based OoD object predictions.

Using linear models as meta classifiers, such as logistic regressions in our case, allows us to track each variable of the meta model. LARS is an effective way of analyzing the correlation of variables with the respective response in linear models. Besides being a very lightweight, such a meta classifier contributes as monitoring method to safer as well as more transparent deep learning applications.

Baseline plain | Baseline plain + meta classifier

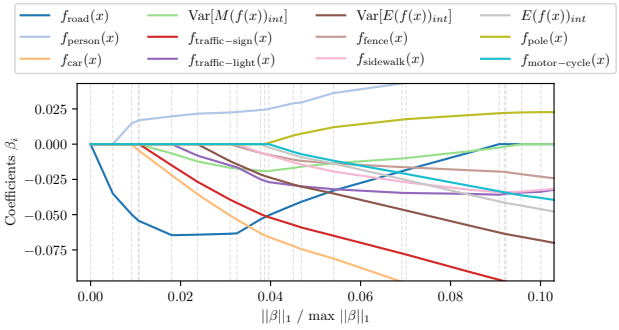
OoD training | OoD training + meta classifier

Figure 18: OoD detection for one scene with different combinations of entropy thresholding for the plain model, entropy thresholding after OoD training and meta classification. For all the OoD predictions the same threshold score of $t = 0.5$ was used. The red segments indicate OoD object predictions.

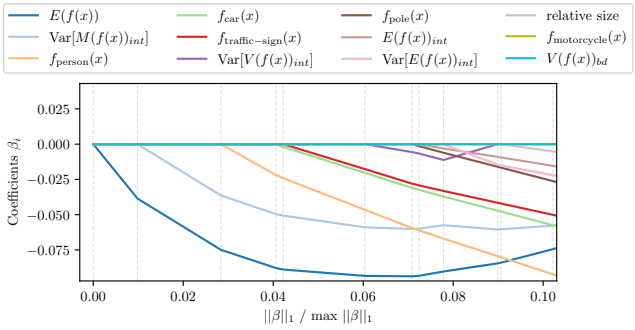
OoD training | OoD training + meta classifier

OoD training | OoD training + meta classifier

Figure 19: OoD detection performed by the OoD-trained network with and without meta classifiers. The red segments indicate OoD object predictions.



Baseline model only



After OoD training

Figure 20: Least angle regression of the meta classifier applied to the softmax output before (left) and after OoD training (right) is applied. The entropy threshold is set to $t = 0.3$ as the experiments showed this threshold yielding the most efficient meta classification models. The 12 meta classification features having the highest correlation are displayed for each model.

K. OoD Detection Methods Description and Run-time comparison

After describing the applied OoD detection methods of our experiments in more detail, we provide a comparison of inference times in order to judge the methods' suitability as an online application.

K.1. Methods

Most methods for OoD detection perform on image-level. However, the state-of-the-art methods for OoD detection can be adapted to semantic segmentation in a straightforward manner. As a reminder, we denote the pixel-wise softmax probability at pixel location $z \in \mathcal{Z}$ with $f^z(x) \in (0,1)^{|\mathcal{C}|}$ for an image $x \in \mathcal{X}$, see also section 3.2.

Maximum softmax probability. The pixel-wise maximum softmax probability is a commonly used baseline for OoD detection. We apply this metric as OoD score for each pixel $z \in \mathcal{Z}$:

$$1 - \max_{j \in \mathcal{C}} f_j^z(x) = 1 - f_{\hat{c}(z)}^z(x),\ x \in \mathcal{X}\ . \qquad (20)$$

ODIN. Let $\tau \in \mathbb{R} \setminus \{0\}$ be a temperature scaling parameter and $\delta \in \mathbb{R}$ a perturbation magnitude. We first add small

perturbations to each pixel $z \in \mathcal{Z}$ of image $x \in \mathcal{X}$:

$$\tilde{x}^z = x^z - \delta \text{sign}\left(-\frac{\partial}{\partial x^z} \log f^z_{j^*}(x)\right) . \quad (21)$$

Then, the OoD score is obtained similar to the maximum softmax probability:

$$1 - \max_{j \in \mathcal{C}} f^z_j(\tilde{x})/\tau . \quad (22)$$

Mahalanobis distance. Let $h(\cdot)$ denote the output of the penultimate layer of a CNN. Under the assumption that $h(\cdot)$ is a class conditional Gaussian, *i.e.*

$$P(h^z(x) \mid y^z(x) = j) = \mathcal{N}(h^z(x) \mid \mu_j, \Sigma_j) \; \forall x \in \mathcal{X} \quad (23)$$

we compute the Mahalanobis distance as OoD score for each pixel $z \in \mathcal{Z}$:

$$\min_{j \in \mathcal{C}} \left\{ (h^z(x) - \hat{\mu}_j)^T \hat{\Sigma}_j^{-1} (h^z(x) - \hat{\mu}_j) \right\} \quad (24)$$

where $\hat{\mu}_j$ and $\hat{\Sigma}_j$ are estimates for class mean μ_j and class covariance Σ_j, respectively, of the latent features in the penultimate layer (see equation (23)).

Monte Carlo dropout. Let $S \in \mathbb{N}$ denote the number of Monte Carlo sampling rounds and let $\hat{p}^z_j(x) = (f^z_j(x))_{s=1}^S$ denote the softmax probabilities of class $j \in \mathcal{C}$ for samples $s \in \{1, \ldots, S\}$. We consider the sum of variances of each class as OoD score for each pixel $z \in \mathcal{Z}$:

$$\sum_{j \in \mathcal{C}} \text{Var}\left(\hat{p}^z_j(x)\right), x \in \mathcal{X} \quad (25)$$

where $\text{Var}(\cdot)$ is the empirical variance function. Regarding Monte Carlo dropout as baseline, we conducted experiments also with the mutual information. However, we observed worse anomaly detection performance compared to the sum over variances.

K.2. Inference Time Comparison

Methods that estimate uncertainty are relevant for many applications involving deep learning. In practice, monitoring systems need to compute uncertainty in real time in order to provide online applicability. Therefore, one crucial factor is the run-time of OoD detection methods. In this subsection we compare the inference time, *i.e.* the time from feeding an image through a model to obtaining pixel-wise OoD scores. We report the results in table 4. For reasons of comparison, we choose the same input and the same segmentation network architecture for all evaluated methods. We observe that our OoD training approach is highly efficient in terms of run time, only being outperformed by the weak maximum softmax probability baseline. However, the gap is less than one second. Compared to the remaining methods, the time difference is more substantial, ranging from 17 seconds for ODIN up to 70 seconds for the Mahalanobis distance.

OoD detection method	time in s ↓ per image
Maximum Softmax	**1.52**
Entropy Thresholding	2.39
ODIN	19.63
Monte Carlo Dropout	33.48
Mahalanobis Distance	72.54
Ours: OoD training + entropy thresholding	2.38

Table 4: Run time comparison of the different OoD detection methods. The inference time for one image is reported in seconds. For all methods the same input as well as the same underlying segmentation network is used.

	LaF test AUPRC	Fishy val AUPRC	City val mIoU
OoD train	**72.58** ± 04.30	**73.51** ± 06.12	**88.95** ± 00.41
Baseline	46.00 ± 00.00	27.54 ± 00.00	90.30 ± 00.00

Table 5: Averaged performance scores over 8 random seeds (for COCO subsampling) and their corresponding standard deviations on LostAndFound, Fishyscapes and Cityscapes. For each run, 10 epochs of OoD training were executed with DeepLabV3+ and $\lambda = 0.9$.

L. Entropy Maximization with Different Seeds

For entropy maximization, a subset of images from the COCO dataset is used. This subset is randomly sampled, *c.f.* section 5. To investigate how different random seeds affect the presented results of this work, we included average performance scores over multiple seeds and the corresponding standard deviations in table 5, where we report the major metric for all considered datasets. In these additional experiments, we even observe scores better than those reported in the paper.