Appendix

A SEMANTIC OVERLAP AND SIMILARITIES IN PREVIOUS BENCHMARK

As stated in Section 3, the main assumption of the current OOD-OD benchmark is that no ID category can be present in the OOD datasets. This is what we call the no-overlap condition. If this condition is met, it is ensured that all predictions done by a model trained on the ID datasets can be considered "incorrect" predictions. The non-overlap condition can mainly be enforced by manual inspection of OOD datasets, due to the existence of unlabeled instances of several objects.

Table 6: Semantic overlap: Number of OOD images containing ID classes

ID class	No. Images
Person (or part)	106
Dining table	142
Other	4

A close inspection of the dataset showed that, in fact, the core assumption of no overlap is not met, since there are labeled and unlabeled instances of ID categories in the OOD datasets. The amount of images in the OOD datasets that contain ID categories is shown in Table 6.

Figure 6: Examples of images in the OOD datasets that contain humans or parts of humans. There exists a semantic overlap between ID and OOD datasets. The images must be removed for the benchmark to have consistency.

Figure 7: Examples of images in the OOD datasets that contain dining tables. Some of these contain also humans. There exists a semantic overlap between ID and OOD datasets. The images must be removed for the benchmark to have consistency.

Some examples of images in the OOD datasets that contain humans or parts of humans are shown in Figure 6. Similarly, examples of images containing "dining tables" in the OOD datasets w.r.t. VOC are shown in Figure 7. Table 7 shows the overlapping categories in each OOD dataset.

Table 7: Overlapping categories in each OOD dataset w.r.t. VOC

ID: VOC	COCO	OpenImages
Person	Person	Person, human face, human arm, woman, human head, human hand, human hair, human nose, human ear, human mouth, human nose, human eye, human beard, body part
Dining table	Spoon, fork, pizza, sandwich, cake, hot dog, wine glass, spoon	Salad, plate, broccoli, tableware, fork, baked goods, spoon
Boat	-	Boat
Potted plant	-	Houseplant, flowerpot
Cat	-	Cat

All images containing overlapping classes with the ID ones must be removed for the benchmark to comply with the non-overlap condition. Table 7 presents the detailed list of OOD categories that overlap with the corresponding ID category in each OOD dataset with respect to VOC categories. For BDD100k as ID, only the images containing instances of people or parts of people were removed.

Furthermore, we present a list of OOD categories and their corresponding ID category that are considered semantically or visually *near* w.r.t. VOC in Table 8. All the other categories in the OOD datasets that are not in the *near* list are considered *far* categories when VOC is the ID dataset. It is important to note, as explained in Section 4, that the images were manually checked to ensure the correct assignment into each new split, or removal. Figure 8 show examples of OOD images that contain *near* categories w.r.t. VOC as ID dataset, along with the prediction from Faster-RCNN trained on VOC.

Table 8: Semantically and visually *near* categories in each OOD dataset w.r.t. VOC

VOC category	COCO	OpenImages
Horse	Zebra	-
Cat	-	Jaguar, leopard, cheetah
Chair	Bench	-
Person	-	Clothing
Dining table	Spoon, fork, carrot, orange, apple, cup, bowl	Zucchini, food, knife
Television	Laptop	Tablet computer, laptop
Couch	Bed	-
Dog	Bear	Fox
Potted plant	Vase	-
Various	-	Raccoon, harbor seal, hedgehog, otter, sea lion

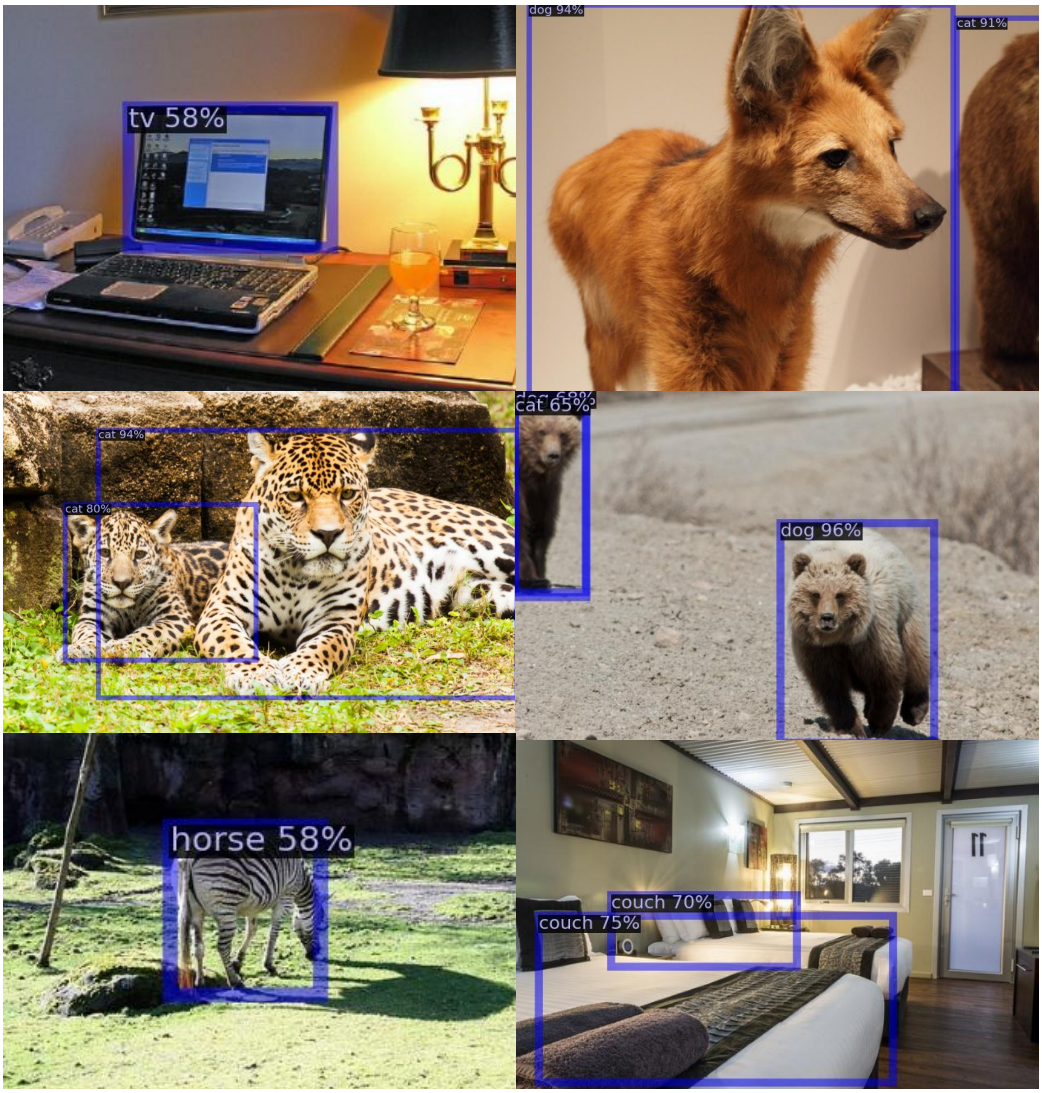


Figure 8: Examples of images in the OOD datasets that contain categories classified as *near* w.r.t. Pascal-VOC as ID dataset. The predictions are made by the Faster RCNN model trained on Pascal-VOC.

B DETAILS ON THE CONSTRUCTION OF THE FMIYC BENCHMARK

Here we provide more details into how the new benchmark was created, in addition to what is already presented in Section 4. Following the observations made in Section A with respect to the semantic overlaps existing in the current OOD-OD benchmark Du et al. (2022b), the first step was to remove the images where semantic overlap exists with the ID categories.

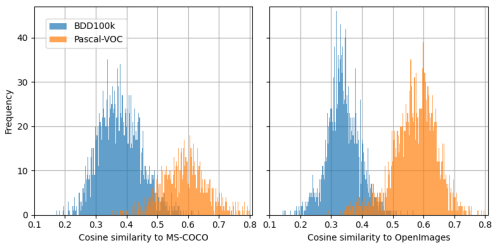
The second step consisted of splitting into *near* and *far* subsets with respect to Pascal-VOC using class names as the criterion. The images containing semantically and visually similar categories from Table 8 were put into the *near* split. The rest were put into the *far* split. The splits were validated using WordNet Miller (1995) and the Wu-Palmer similarity metricWu & Palmer (1994). For each class name in the ID and OOD datasets, the WordNet embedding was obtained. Then, we calculated the highest Wu-Palmer similarity of each OOD class name w.r.t. those of the ID class names. The results in Table 9 show the average WuP similarity for each proposed split, and confirm the stratification. The images were manually inspected to ensure no unlabeled instances of ID categories were present, in which case the image was removed from the benchmark. The manual inspection also ensured the correct assignment of images to each split.

Table 9: Wu-Palmer average similarity scores for the proposed splits. CC=COCO, OI=OpenImages

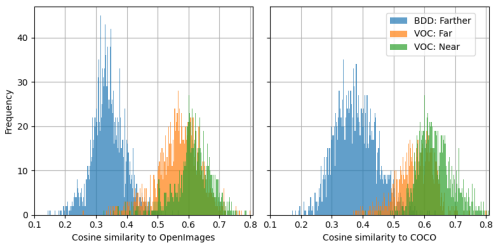
ID	OOD dataset	WuP similarity
VOC	CC Near	0.706 ± 0.225
	OI Near	0.642 ± 0.204
	CC Far	0.683 ± 0.177
	OI Far	0.604 ± 0.193
BDD	CC Farther	0.619 ± 0.158
	OI Farther	0.508 ± 0.175

Next, new images were added to each split. Candidate images from the training sets of COCO and OpenImages were first selected for manual inspection. The candidate images didn't have labeled ID categories, and needed to contain labeled instances of either the *near* or the *far* categories. Candidate images for each split were then manually inspected to ensure also that no ID category was present, and the correct assignment to each split.

For BDD100k as ID, the only modification done to the existing OOD datasets was the removal of images with people, because of overlap with the ID category "pedestrian".



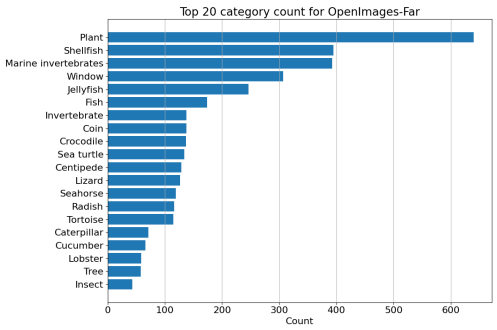
(a) Current benchmark: VOC is semantically and visually more similar to OOD datasets than BDD.



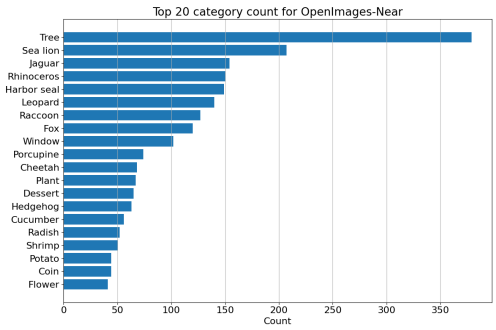
(b) The FMIYC benchmark distinction of *near*, *far* and *farther* splits can be appreciated

Figure 9: Perceptual and semantic (cosine) similarity Mayilvahanan et al. (2023) between ID and OOD datasets using CLIP image encoder embeddings.

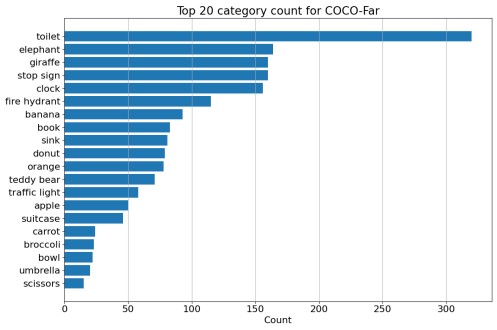
Later, the semantic and visual similarity was assessed using CLIP Radford et al. (2021) embedding space. The embeddings for both ID datasets, and for OOD samples in each split were extracted. Then, following the procedure in Mayilvahanan et al. (2023), we calculated the cosine similarity between ID and their respective OOD datasets. The obtained results before and after creating the splits can be seen in Figure 9. It can be observed that three groups are present. This allowed us to propose the distinction into *near*, *far* and *farther* datasets. *Near* and *far*, are splits that are OOD w.r.t. VOC. Farther are the subsets w.r.t. BDD100k. Each of these subsets exists for COCO and OpenImages, which means that in total, there are six subsets of OOD datasets: COCO-near, COCO-far, OpenImages-near, OpenImages-far w.r.t. VOC; along with COCO-farther and OpenImages-farther w.r.t. BDD100k. The amount of images in each subset is shown in Table 2. In total, there are 7767 images across all splits.



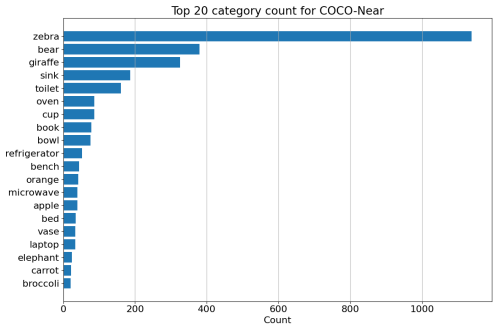
(a) Most frequent labeled categories in OpenImages-Far



(b) Most frequent labeled categories in OpenImages-Near

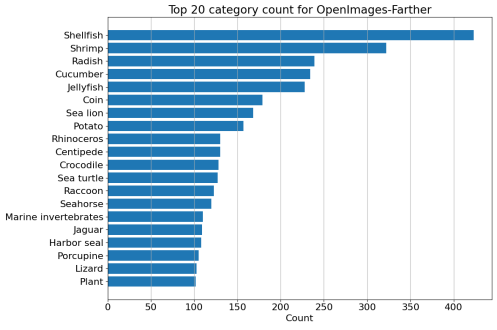


(c) Most frequent labeled categories in COCO-Far

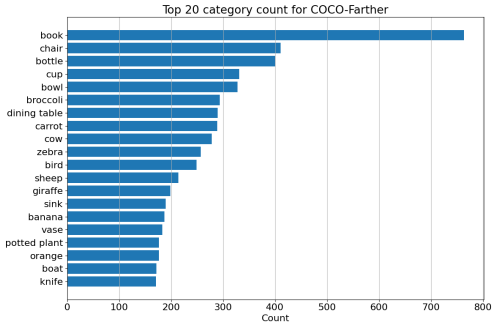


(d) Most frequent labeled categories in COCO-Near

Figure 10: Top 20 category count for OOD datasets w.r.t. Pascal-VOC



(a) Most frequent labeled categories in OpenImages-Farther



(b) Most frequent labeled categories in COCO-Farther

Figure 11: Top 20 category count for OOD datasets w.r.t. BDD100k

Finally, Figure 10 and Figure 11 show the top-20 category count for the images in each split of the new benchmark.

C DETAILS ON THE METRICS USED

This section provides more details about the previous and the newly incorporated metrics.

Previous OOD-OD metrics AUROC and FPR metrics come from binary classification problems. The receiver-operating-characteristic (ROC) curve evaluates the performance of a classifier at varying threshold values. It consists of the plot of the true positive rate (TPR) against the false positive rate (FPR) at each threshold setting. TPR and FPR are defined as follows:

$$\text{FPR} = \frac{FP}{FP + TN} \tag{1}$$

$$\text{TPR} = \frac{TP}{TP + FN} \tag{2}$$

where FP is the number of false positives, TP is the number of true positives, TN is the number of true negatives, and FN is the number of false negatives.

The AUROC is the area under the ROC curve. Since both TPR and FPR are bounded to the interval $[0, 1]$, the AUROC is bounded to the same interval. A perfect classifier would have an AUROC of 1, whereas a random classifier would have an AUROC of 0.5. The value of 0 would mean that the classifier is a perfect misclassifier (predicts negatives as positives and vice-versa). The FPR95 is the false positive rate at 95% true positive rate. The lower the FPR95, the fewer false positives the classifier predicts Lasko et al. (2005).

For the previous OOD-OD benchmark, the main limitation of these two metrics lies in the fact that they have no relation with ground truth (GT) bounding boxes, and rely exclusively on the compliance with the non-overlap assumption, as described in Section 2.2 and Section A. Therefore, AUROC and FPR95 are unable to measure the actual localization of OOD objects. For an illustration of this, see Figure 12.

Moreover, a non-negligible amount of images does not have a single prediction at all, as can be seen in Table 1. AUROC and FPR95 cannot measure that the main objects in Figure 2, Figure 12 and Figure 13 are ignored. They can only take into account the incorrect predictions as in Figure 12. Even if the unknown objects are correctly localized, AUROC and FPR95 are not measuring this since they are unrelated to the GT bounding boxes. For these reasons, we raise the critical question: *are AUROC and FPR95 sufficient metrics to assess the deployment of OOD-OD methods in safety-critical real-world scenarios?*

OSOD metrics The newly proposed metrics for the benchmark exist in the Open Set for object detection (OSOD) community. The metrics were already introduced in Section 4.2. here we give a more detailed definition for each one of them. It is important to note that all of the metrics were calculated using an intersection over union (IoU) threshold of 0.5. This means that one detection is considered as a true positive (TP_U) if the unknown is classified correctly (as unknown or OOD), and its predicted bounding box has an IoU≥ 0.5 with a ground truth (GT) unknown object.

Also, for this case it is important to distinguish two types of false negatives: dismissed or ignored ones, denoted FN^D, and misclassified ones, denoted FN^M. One prediction is considered as FN^D if no predicted bounding box has IoU≥ 0.5 with the GT label. A detection is considered FN^M if a bounding box has IoU≥ 0.5 with a GT unknown but the predicted class is one of the ID categories. The total false negatives for the unknowns are then:

$$FN_U = FN_U^D + FN_U^M \tag{3}$$

The precision of the unknowns P_U is defined in a similar way as the binary classification metric:

$$P_U = \frac{TP_U}{TP_U + FP_U} \tag{4}$$

where all quantities refer to unknowns: TP_U are the true positive predictions, and FP_U are the false positive predictions. Also, let us note that $TP_U + FP_U$ are the total number of predictions

Figure 12: Incorrect predictions of Faster-RCNN trained on BDD100k on images from the OOD datasets in the current benchmark. AUROC and FPR95 cannot measure that the main OOD objects are ignored. They can only take into account the incorrect predictions. OSOD metrics can quantify the dismissal of unknown objects

for the unknown class. Therefore, what P_U is measuring is the ratio of true positives divided by all unknown predictions. In other words, P_U tells the proportion of predictions for unknowns that were actually ground-truth unknowns Powers (2011).

The recall of the unknowns R_U is defined as:

$$R_U = \frac{TP_U}{TP_U + FN_U} \tag{5}$$

where FN_U are the false negatives. Let us note that $TP_U + FN_U$ are the total number of ground-truth unknowns. In other words, R_U tells us the proportion of ground-truth unknowns that were found by the detector.

For the average precision of the unkowns AP_U, it is defined as the area under the precision-recall curve:

$$AP = \int_0^1 p(r)dr \tag{6}$$

which is usually calculated by the interpolation of rectangles of the sampled values:

Figure 13: Absense of predictions of Faster-RCNN trained on BDD100k on images from the OOD datasets in the current benchmark. AUROC and FPR95 cannot measure that all OOD objects in these images are ignored. Dismissing OOD objects is not measurable using the current metrics. OSOD metrics can quantify the dismissal of unknown objects

$$AP = \sum_{m}^{M} (r_{n+1} - r_n) p_{in}(r_{n+1}), \tag{7}$$

$$p_{in}(r_{n+1}) = \max_{\tilde{r} \geq r_{n+1}} p(\tilde{r}) \tag{8}$$

where p_{in} represents the interpolated precision at each detection point, which is obtained by taking the maximum precision whose recall value is greater or equal than (r_{n+1}) Padilla et al. (2020).

Next, usually OSOD works report the absolute open set error (AOSE), that is defined as the total number of unknown objects that are predicted as one of the ID classes (which would correspond to FN_U^M). Since the absolute number of these is not comparable across datasets (because each dataset has a different number of unknown objects), we propose using a metric that we call normalized open set error (nOSE) that is defined as:

$$\text{nOSE} = \frac{FN_U^M}{TP_U + FN_U} \tag{9}$$

where indeed $TP_U + FN_U$ is once more the total number of ground-truth unknown objects. The nOSE is comparable across datasets, and estimates the proportion of OOD objects that are confounded with ID objects.

A summary of the purpose, limitations, and advantages of the used metrics can be found in Table 10.

Table 10: Overall metrics summary

Metric	Purpose	Limitations	Advantages
AUROC, FPR95	Measures the ability of a scoring function to detect incorrect predictions	Cannot take into account ignored objects	Does not depend on GT labels, can detect incorrect predictions that do not overlap with labeled objects
Precision	Measures the percent of correct predictions over the total of predictions	Need good GT labels. Cannot measure unlabeled unknowns.	Measure localization of GT objects
Recall	Measures the percent of found objects divided by the total number of labeled objects		
nOSE	Measures the percent of unknown objects confounded with an ID object		

D Details On Evaluated OOD Detection Methods

We present further details on the OOD detection methods used in the paper. All of the methods come from the Image classification literature Yang et al. (2024), except for VOS Du et al. (2022b).

D.1 Preliminaries.

Using the notation from Section 2.1, let us recall that a trained object detector $\mathcal{M}$ takes as input an image x, along with a confidence threshold t^*, and for each i-th detected object outputs a bounding box $\boldsymbol{b}_i \in \mathbb{R}^4$ and a vector of logits $\boldsymbol{c}_i \in \mathbb{R}^{|\mathbb{C}|}$, with dimension equal to the number of ID classes $\mathbb{C}$. The model output is the set:

$$\mathcal{M}(x; t^*) = \{(\boldsymbol{b}_i, \boldsymbol{c}_i)\}_{i=1}^{D} \tag{10}$$

where D is the number of detections in each image. Each tuple $(\boldsymbol{b}_i, \boldsymbol{c}_i)$ corresponds to one detected object. Note that $D = 0$ is possible, and in such a case the output is empty. Furthermore, the so-called softmax activation is given by:

$$\sigma(c_j) = \frac{e^{c_j}}{\sum_m^{|\mathbb{C}|} e^{c_m}} \tag{11}$$

which transforms the logits vector into a vector of probabilities for each ID class, such that $\sum_j^{|\mathbb{C}|} \sigma(c_j) = 1$. In this notation, the index j denotes the class index, and the index i denotes the object index. An alternative output is then given by the vector of probabilities after softmax: $\mathcal{M}(x; t^*) = \{(\boldsymbol{b}_i, \boldsymbol{p}_i)\}_{i=1}^{D}$, where $p_j = \sigma(c_j)$. The predicted probability of each detected object is the maximum after softmax, let it be denoted by $\hat{p}_i = \max_j p_{ij}$. In any case, to have $D > 0$, there must be at least one prediction such that $\hat{p}_i \geq t^*$.

The OOD detection problem. Is formulated as a binary classification task leveraging a (confidence) scoring function $\mathcal{G}$ for each detected instance $(\boldsymbol{b}_i, \boldsymbol{c}_i)$, so that:

$$\mathcal{G}(x, \boldsymbol{b}_i, \boldsymbol{c}_i) = g_i \in \mathbb{R} \tag{12}$$

The scoring function aims to distinguish between ID and OOD objects, using a thresholding function $\boldsymbol{\Omega}$ with threshold τ as presented in eq. (13).

$$\boldsymbol{\Omega}\Big(g_i, \tau\Big) = \begin{cases} 1 \quad ID & \text{if } \; g_i \geq \tau \\ 0 \quad OOD & \text{if } \; g_i < \tau \end{cases} \tag{13}$$

For the OOD-OD problem, only those detected objects above the threshold t^* are considered. Therefore, if no object is detected in a given image, there is no input for the scoring function $\mathcal{G}$ for such an image. In a general sense, each of the OOD detection methods is a realization of the scoring functions $\mathcal{G}$. Figure 14 presents a depiction of the workflow of OOD-OD scoring functions.

It is important to avoid possible confusion and it can be useful to reiterate here that t^* and τ are two different thresholds. The object detection model $\mathcal{M}$ uses a confidence threshold $t^* \in \mathbb{R}^{[0,1]}$ that is usually the one that maximizes the mAP in the ID test set. This threshold filters the output of the model so that all detected objects satisfy $\hat{p}_i \geq t^*$. On the other hand, the OOD scoring functions $\mathcal{G}$ use each one its own threshold $\tau \in \mathbb{R}$, which corresponds to the one that makes that 95% of the g_i of detected ID objects are above the threshold.

D.2 Evaluated methods

For the adaptation of each method from image classification to object detection, in each case, the score is calculated per each detected object above the threshold t^*. Therefore, there can be zero or several detections per image. Each of the equations in the following section has been adapted to match our notation, and all of them explain the adaptation done to work at the object level.

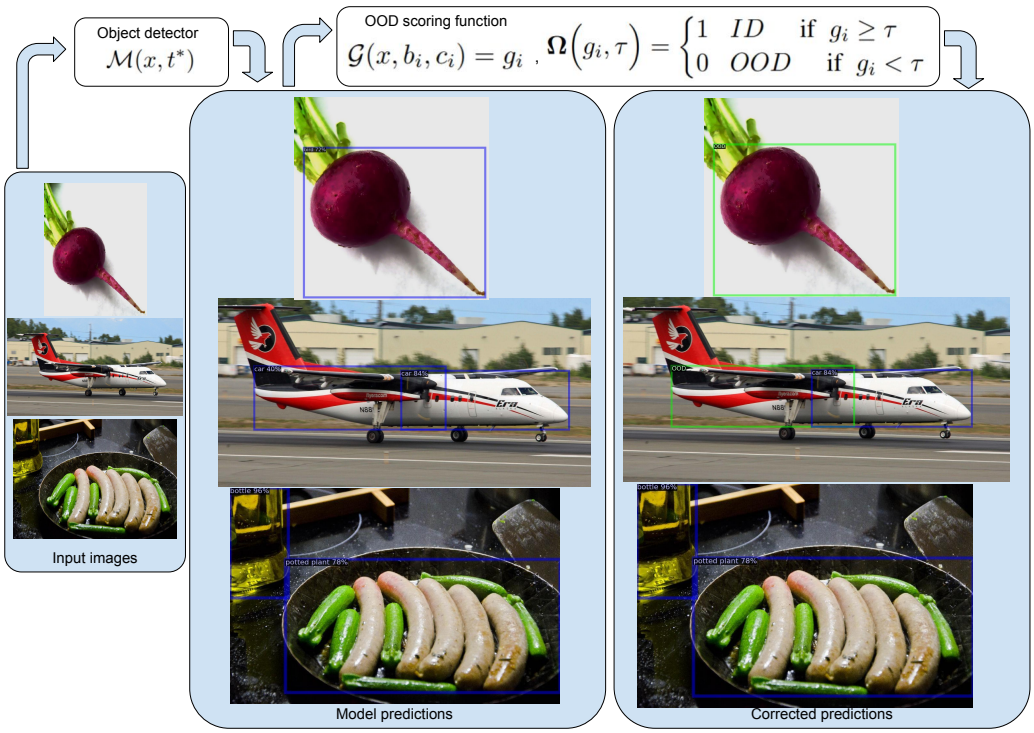


Figure 14: General workflow of OOD-OD scoring functions. The outputs of the base model $\mathcal{M}$ are the inputs to scoring functions $\mathcal{G}$. If the object detector ignores a given object, scoring functions will ignore it, too. The model predictions not marked as OOD, remain with the predicted class.

D.2.1 Output-based methods

Output based methods take either the $\boldsymbol{c}_i$ or the $\boldsymbol{p}_i$ as input to the scoring functions. This family of methods is applicable to all of the architectures tested: Faster-RCNN, Yolov8 and RT-DETR.

Maximum softmax probability (MSP). This is perhaps the most classical baseline in OOD detection for image classification Hendrycks & Gimpel (2016). It consists of directly choosing the maximum softmax value:

$$\max_j p_j = \max_j \frac{e^{c_j}}{\sum_m^{|\mathcal{C}|} e^{c_m}} \tag{14}$$

where e is the Euler number.

Energy score. Proposed by Liu et al. (2020), it calculates the energy score using the activation logits c_i as:

$$E(\boldsymbol{c}_i; T) = -T \log \sum_j^{|\mathbb{C}|} e^{c_j/T} \tag{15}$$

where T is the temperature (usually set to $T = 1$).

Generalized entropy score (GEN). Presented by Liu et al. (2023), the authors propose using the family of generalized entropies:

$$G_\lambda(\boldsymbol{p}_i) = \sum_j p_j^\lambda (1-p_j)^\lambda \tag{16}$$

when $\lambda = 1/2$:

$$G_{1/2}(\boldsymbol{p}_i) = \sum_j \sqrt{p_j(1-p_j)} \tag{17}$$

D.2.2 FEATURE-BASED METHODS

If the model $\mathcal{M}$ has L total layers, and its last layer L is a linear one (also called fully connected), then the activations of the $L-1$ (penultimate) layer are considered the extracted features $\boldsymbol{z}_{L-1} \in \mathbb{R}^d$, where d is the dimension of the feature. Then, for a given input image x, and a detection $(\boldsymbol{b}_i, \boldsymbol{c}_i)$, the features of each detected object are defined as:

$$\boldsymbol{z}_{L-1}^i = \mathcal{M}_{L-1}(x; t^*) \tag{18}$$

where $\mathcal{M}_l$ denotes the latent activation of $\mathcal{M}$ at layer l. To simplify notation, let us denote the per-object feature $\boldsymbol{z}_{L-1}^i$ by $\boldsymbol{z}_i$. In all cases, $\boldsymbol{z}_i^*$ denotes the features of a detected object $(\boldsymbol{b}_i^*, \boldsymbol{c}_i^*)$ from a test image x^*. Feature-based methods considered here need a training phase, and for this phase they take as input the $\boldsymbol{z}_i$ of the training set. At test time, their input is the $\boldsymbol{z}_i^*$ of test samples.

This family of methods is not applicable to Yolov8, since this architecture has no final linear layer: it is fully convolutional. Therefore, it is not possible to associate a set of features to a specific detected object. This family of methods can be used with Faster-RCNN and RT-DETR.

k-Nearest neighbors (kNN). Introduced by Sun et al. (2022), first normalizes the feature for each detected object: $\mathbf{z}_i = \boldsymbol{z}_i / \|\boldsymbol{z}_i\|_2$, where $\|\cdot\|_2$ denotes the L2 norm. Then, the normalized embeddings of the training data are stored: $\bar{\mathbb{Z}}_N = (\mathbf{z_1}, ..., \mathbf{z}_N)$, where N are the number of objects detected in the training set.

During testing, the normalized features $\mathbf{z}_i^*$ are derived, and the euclidean distances $\|\mathbf{z}_i^* - \mathbf{z}_j\|_2$ are calculated with respect to the train embeddings $\mathbf{z}_j \in \mathbb{Z}_N$. Afterward, the embeddings are reordered according to the increasing distance $\|\mathbf{z}_i^* - \mathbf{z}_j\|_2$. The reordered embedding sequence is $\bar{\mathbb{Z}}'_N = (\mathbf{z}_{(\mathbf{1})}, \mathbf{z}_{(\mathbf{2})}, ..., \mathbf{z}_{(N)})$. The scoring function is defined as:

$$r_k(\mathbf{z}_i^*) = \|\mathbf{z}_i^* - \mathbf{z}_{(k)}\|_2 \tag{19}$$

which corresponds to the distance to the k-th nearest neighbor in the normalized feature space Sun et al. (2022).

Mahalanobis distance. Proposed by Lee et al. (2018), the Mahalanobis score calculates the distance to the centroids of a class-conditional Gaussian distribution. The predicted class per detected object is denoted y_c^i and corresponds to the index of the $\max$ value of either the c_i or the p_i. Then the empirical class mean and covariance matrix of training samples are estimated:

$$\hat{\mu}_c = \frac{1}{N_c} \sum_{j:y_c} \boldsymbol{z}_j, \quad \hat{\Sigma} = \frac{1}{N} \sum_c^{\mathcal{C}} \sum_{j:y_c} (\boldsymbol{z}_j - \hat{\mu}_c)(\boldsymbol{z}_j - \hat{\mu}_c)^\top \tag{20}$$

where N_c denotes the total number of objects of class y_c detected in the training set, N is the total number of detected objects in the training set in all classes, and j are de indexes of detected objects of class y_c. Then the Mahalanobis confidence score is defined as the Mahalanobis distance between the features z_i^*, and the closest class-conditional Gaussian distribution:

$$M(\boldsymbol{z}_i^*) = \max_c -(\boldsymbol{z}_i^* - \hat{\mu}_c)\hat{\Sigma}^{-1}(\boldsymbol{z}_i^* - \hat{\mu}_c)^\top \tag{21}$$

which corresponds to the log of the probability of the test sample Lee et al. (2018).

Deep deterministic uncertainty (DDU). A work by Mukhoti et al. (2023), fits a Gaussian mixture model (GMM) on the feature space, then computes the density under the GMM. Similar to Equation (20), the mean per class $\hat{\mu}_c$ and the covariance matrix $\hat{\Sigma}$ are computed for the features z_i of each detected object (b_i^*, c_i^*). Then the weights of the GMM are computed as:

$$\pi_c = \frac{1}{N} \sum y_c \tag{22}$$

which denotes the proportion of detected objects for each class y_c over the total N detected objects in the training dataset. During inference time, the density under the GMM is computed for the features $\boldsymbol{z}_i^*$ of a detected object $(\boldsymbol{b}_i^*, \boldsymbol{c}_i^*)$ from a test image x^*:

$$q(\boldsymbol{z}_i^*) = \sum_{y_c} q(\boldsymbol{z}_i^*|y_c)\pi_c, \quad \text{where} \quad q(\boldsymbol{z}_i^*|y_c) \sim \mathcal{N}(\mu_c; \sigma_{y_c}) \tag{23}$$

D.2.3 Output-feature (mixed) based methods

This family of methods takes both the outputs (either the c_i or the p_i) and the features z_i for each detected object (b_i, c_i) as inputs to the scoring functions. This family of methods was not applicable to Yolov8 for the same reasons as for the previous family of methods.

Activation shaping (ASH). Showcased by Djurisic et al. (2022), involves a reshaping of the feature z_i, and subsequent use of the energy score from Equation (15). The reshaping is done by first calculating a threshold t that corresponds to the p-th percentile of the entire set of the detected objects representations of the training set:

$$\mathbb{Z}_N = (z_1, ..., z_N) \tag{24}$$

Afterward, we calculate $s_1 = \sum_j z_j$. Then all values below t are set to 0 to obtain a pruned version of the features $\mathbb{Z}_N^p = (z_1^p, ..., z_N^p)$. Using the $\mathbb{Z}_N^p$, we calculate $s_2 = \sum_j z_j^p$. Finally, all non-zero values in $\mathbb{Z}_N^p$ are multiplied with $\exp(s_1/s_2)$, to obtain the pruned and reshaped features:

$$\begin{aligned} \mathbb{Z}_N^r &= \mathbb{Z}_N^p \exp(s_1/s_2) \\ &= (z_1^p \exp(s_1/s_2), ..., z_N^p \exp(s_1/s_2)) \\ &= (z_1^r, ..., z_N^r) \end{aligned} \tag{25}$$

Finally, the pruned and reshaped features are passed through the final fully connected layer L to obtain the logit activations c_i, which are passed to the energy score calculation as in Equation (15). The authors found that the method works best when using a pruning percentile of about 90% Djurisic et al. (2022).

Directed sparsification (DICE). Introduced by Sun & Li (2022), the authors consider the weight matrix of the final fully connected layer $\mathbf{W} \in \mathbb{R}^{d \times |\mathcal{C}|}$, where d is the dimension of the feature z_i, and $|\mathcal{C}|$ is the number of ID categories. This matrix is then subject to sparsification, to preserve the most important weights in it. The contribution is measured by a matrix $\mathbf{V} \in \mathbb{R}^{d \times |\mathcal{C}|}$, where each column $\mathbf{v}_c \in \mathbb{R}^d$ is given by:

$$\mathbf{v}_c = \mathbb{E}_{z_j \in \mathbb{Z}_N}[\mathbf{w}_c \odot z_j] \tag{26}$$

where $\odot$ represents the element-wise multiplication, $\mathbf{v}_c$ indicates the weight vector for class y_c, and $\mathbb{Z}_N$ is as defined in Equation (24). Then the top-k weights are selected from the largest values of $\mathbf{V}$, to obtain a sparsified matrix $\mathbf{W}'$. This matrix is now used as the final layer weights instead of the $\mathbf{W}$. Finally, the obtained c_i are passed to the energy scoring function from Equation (15) Sun & Li (2022).

Rectified activations (ReAct). Proposed by Sun et al. (2021), it performs a clipping operation on the features z_i, and the calculation of the energy score. The rectification (or clipping) is performed as:

$$\bar{\boldsymbol{z}}_i = \min(\boldsymbol{z}_i, t) \tag{27}$$

where each element of $\boldsymbol{z}_i$ is truncated to be at most equal to the threshold t. This threshold is calculated so that a given percentile of the activations is less than the threshold. For instance, at percentile $p = 90$, 90% of ID train activations are below the threshold t. The authors found that a percentile of 90 works best. Then, the $\bar{\boldsymbol{z}}_i$ are passed as inputs to the final layer to obtain the outputs $\boldsymbol{c}_i$, which are then used to calculate the energy score as in Equation (15) Sun et al. (2021).

Virtual logit matching (ViM). A method inspired by a thorough geometrical analysis of the space of the matrix $\mathbf{Z}$, whose rows are the $\boldsymbol{z}_i$ for all detected objects in the training set. Let $\mathbf{X}$ denote a centered version of $\mathbf{Z}$, obtained by offsetting the $\boldsymbol{z}_i$ by a vector $\mathbf{o} = -(\mathbf{W}^\top)^+\mathbf{b}$, where $(\cdot)^+$ denotes the Moore-Penrose inverse, $\mathbf{W}$ is the final layer weight matrix and $\mathbf{b}$ is the final layer bias. The eigendecomposition of the matrix $\mathbf{X}^\top\mathbf{X}$ is:

$$\mathbf{X}^\top\mathbf{X} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^{-1} \tag{28}$$

where eigenvalues $\mathbf{\Lambda}$ are ordered decreasingly. The first D columns of $\mathbf{Q}$ are called the D-dimensional principal subspace P. The residual subspace $P^\perp$ is spanned by the remaining $D+1$ to the last columns of $\mathbf{Q}$, and is represented by the matrix $\mathbf{R} \in \mathbb{R}^{N\times(N-D)}$, where N is the number of detected objects in the train set. Then $\boldsymbol{z}_i^{P^\perp}$ denotes the projection of $\boldsymbol{z}_i$ onto $\mathbf{R}$: $\boldsymbol{z}_i^{P^\perp} = \mathbf{R}\mathbf{R}^\top z_i$. The virtual logit c_0 is calculated as:

$$c_0 = \alpha\|\boldsymbol{z}_i^{P^\perp}\| = \alpha\sqrt{\boldsymbol{z}_i^\top\mathbf{R}\mathbf{R}^\top\boldsymbol{z}_i} \tag{29}$$

which corresponds to the norm of the residual $z_i^{P^\perp}$ rescaled by a constant α. This constant is calculated as:

$$\alpha = \frac{\sum_j^K \max_{m=1,\ldots,|\mathcal{C}|}\{c_m^j\}}{\sum_{j=1}^K \|\boldsymbol{z}_i^{P^\perp}\|} \tag{30}$$

where $\boldsymbol{z}_1, \boldsymbol{z}_2, ..., \boldsymbol{z}_K$ are uniformly sampled K training examples, and c_m^j is the m-th logit of c_j. This constant scales the virtual logit to the average maximum of the original logits. Finally, the ViM score is calculated as:

$$\text{ViM}(z_i) = \alpha\|z_i^{P^\perp}\| - \ln\sum_{j=1}^{|\mathbb{C}|} e^{c_j} \tag{31}$$

which, in summary, is the virtual logit minus the energy score of the rest of the logits. For the hyperparameter D, the authors recommend using $D = 1000$ if the dimension of the feature $d > 1000$, or use $D = 512$ otherwise Wang et al. (2022).

D.2.4 LATENT SPACE METHODS

In this family we find methods that take as input other latent activations inside the network. We took inspiration from Arnez et al. (2024); Wilson et al. (2023) and built a method based on the latent space convolutional activations. In our case, we used directly the latent activations without doing Monte Carlo dropout sampling of entropy estimation as in Arnez et al. (2024), nor using a surrogate model or the generation of adversarial examples as in Wilson et al. (2023).

Latent representation density (LaRD). We start by considering a convolutional feature map $z_{i,l} \in \mathbb{R}^{N_c \times W \times H}$, where N_c is the number of channels, W is the width and H is the height of the latent activation, extracted at layer l. Then it is possible to use the predicted bounding boxes b_i and the feature maps as inputs for the ROIAlign (RA) algorithm He et al. (2017), which can extract the corresponding portion of the feature maps per each predicted object:

$$o_{i,l} = \text{RA}(z_{i,l}, b_i), \text{where } o_{i,l} \in \mathbb{R}^{N_c \times R \times R} \tag{32}$$

Where R is the parameter that fixes the size of the output of the RA algorithm, that outputs crops of the feature map $z_{i,l}$ with a given fixed-sized for all objects, independently of their aspect ratio or actual size in the image. Then an average per channel is taken to reduce the dimensionality of these representations:

$$\bar{o}_{i,l} = \frac{1}{HW} \sum_{h=1}^{H} \sum_{w=1}^{W} o_{i,l}(c, h, w), \text{ where } \bar{o}_{i,l} \in \mathbb{R}^{N_c} \tag{33}$$

The set $O_l = \{\bar{o}_{i,l}, y_i\}_{d=1}^{D}$ consists of all the averaged latent representations at layer l of each object found by the object detector in one image, along with the predicted class y_i. Then, we also want to build a density estimator, by making a forward pass through the training set to obtain the set of all the ID objects latent representations: $\mathcal{O}_{train,l} = \{O_l\}_{x=1}^{N_t}$, where N_t is the size of the training set. Afterward, we use the methodology as in the Mahalanobis distance baseline to obtain a scoring function for each of the detected objects. We used a hyperparameter of $R = 9$ for all experiments. For Faster-RCNN, the chosen latent layer was the RPN intermediate convolutional layer as in Arnez et al. (2024); for Yolov8, it was the final layer of the backbone, after evaluation of each layer. For RT-DETR the chosen hidden layer was the first encoder module, similarly, after evaluation of each layer.

E Details on the training of architectures

This section provides details on the training of Yolov8 Sohan et al. (2024) and RTDETR Zhao et al. (2024). Both architectures were trained on a single GPU Nvidia A100 40G. The achieved mAP by both models in each ID dataset is found in Table 3.

E.1 Yolov8

We trained the nano version of Yolov8 for both ID datasets (BDD100k and Pascal-VOC). We used the same hyperparameters for both models. Most of them corresponded to the default hyperparameters. They were trained for 100 epochs, using the AdamW optimizer with momentum of 0.937 and weight decay of 5×10^{-4}. The learning rate was 10^{-3}, and was controlled by a cosine scheduler. The batch size was 16, and we used the copy-paste augmentation, on top of the mosaic, translate, scale, erase, and flip-lr default augmentations. For the training, we used the Ultralytics library Jocher et al. (2023).

E.2 Real-Time DETR

We fine-tuned a version of RT-DETR that was pre-trained on COCO for both ID datasets (BDD100k and Pascal-VOC). The pretrained version can be found in Huggingface: RT-DETR. Both versions used early stopping with a patience of 16 epochs. The hyperparameters for both models can be found in Table 11.

Table 11: Hyperparameters for training RT-DETR whith ID datasets BDD100k and Pascal-VOC

Parameter	ID: BDD	ID: VOC
Batch size	8	8
Inference threshold	0.25	0.25
Learning rate backbone	4×10^{-6}	2×10^{-6}
Max epochs	60	60
Num queries	100	100
Random seed	40	40
Learning rate	4×10^{-5}	2×10^{-5}

F DETAILED RESULTS PER METHOD AND ARCHITECTURE

This section provides detailed results per architecture and per method on all metrics. First, the results for previous metrics are presented. Afterward, the results for the new metrics are detailed. Finally, a study of the correlations among previous and new metrics is presented.

F.1 DETAILED RESULTS ON THE PREVIOUS OOD-OD METRICS

Table 12: OOD detection performance for FasterRCNN (Vanilla) on various OOD splits (ID: PascalVOC). Metrics are AUC↑ (%) and FPR95↓ (%). LaRD represents best of (Mahalanobis PCA, KNN PCA, GMM PCA). Best result per metric column is in **bold**. [B]Indicates the primary scoring method of the VOS (Virtual Outlier Synthesis).

	FasterRCNN (Vanilla) — PascalVOC							
	COCO-Near (OOD)		**COCO-Far (OOD)**		**OpImg-Near (OOD)**		**OpImg-Far (OOD)**	
Method	AUC↑	FPR95↓	AUC↑	FPR95↓	AUC↑	FPR95↓	AUC↑	FPR95↓
ViM	75.7	85.5	77.8	87.4	73.0	87.1	74.9	91.6
Mahalanobis	59.8	95.9	64.9	95.5	59.7	94.6	60.3	95.9
MSP	73.8	88.3	77.3	88.0	70.5	90.4	75.4	87.9
Energy	86.5	45.5	82.3	56.2	81.5	57.9	81.8	52.6
ASH	82.9	49.9	74.5	66.6	78.7	59.9	74.8	60.8
DICE	82.7	62.0	78.2	76.7	79.1	67.3	76.7	71.6
ReAct	85.1	58.1	75.2	82.5	83.1	66.0	73.4	83.0
GEN	87.4	44.8	84.5	55.0	82.8	**56.2**	83.7	52.1
DICE+ReAct	66.3	89.8	56.0	94.8	71.4	88.9	48.3	99.0
DDU	64.0	97.6	68.3	97.0	70.4	97.2	66.3	98.3
VOS[B](Energy)	**90.0**	**44.6**	**89.1**	**44.9**	**84.4**	60.0	**86.0**	**49.1**
LaRD	73.8	81.7	68.6	88.0	70.0	88.4	70.0	89.2

Table 13: OOD detection performance for FasterRCNN enhanced with VOS (Virtual Outlier Synthesis) on various OOD splits (ID: PascalVOC). Metrics are AUC↑ (%) and FPR95↓ (%). LaRD represents best of (Mahalanobis PCA, KNN PCA, GMM PCA). Best result per metric column is in **bold**. [B]Indicates the primary scoring method of the VOS (Virtual Outlier Synthesis).

	FasterRCNN (VOS) — PascalVOC							
	COCO-Near (OOD)		**COCO-Far (OOD)**		**OpImg-Near (OOD)**		**OpImg-Far (OOD)**	
Method	AUC↑	FPR95↓	AUC↑	FPR95↓	AUC↑	FPR95↓	AUC↑	FPR95↓
ViM	77.4	87.7	80.3	85.9	73.4	89.8	77.2	92.2
Mahalanobis	60.9	95.9	65.5	94.9	60.3	95.5	64.8	95.5
MSP	69.1	91.5	75.1	89.2	65.6	91.1	72.6	88.2
ASH	**90.2**	44.1	87.4	51.4	84.8	59.8	82.5	56.2
DICE	88.0	56.5	88.3	53.4	82.7	67.8	80.8	59.0
ReAct	87.1	57.1	79.9	72.2	**85.6**	64.5	77.1	76.3
GEN	89.7	**42.9**	**89.3**	45.7	85.3	**58.2**	**86.0**	50.7
DICE+ReAct	74.9	84.8	67.3	88.1	74.8	88.5	58.6	98.9
DDU	67.5	99.2	70.0	96.9	72.5	99.3	72.7	98.3
VOS[B](Energy)	90.0	44.6	89.1	**44.9**	84.4	60.0	86.0	**49.1**
LaRD	75.1	77.5	68.1	87.8	67.8	87.2	67.8	89.2

Table 14: OOD detection performance for FasterRCNN variants on Farther OOD splits (ID: BDD). LaRD represents best of (Mahalanobis PCA, KNN PCA, GMM PCA). Higher AUC is better (↑), lower FPR95 is better (↓). Best result per metric column is in **bold**. [B]For the FasterRCNN (VOS) architecture, this indicates the primary scoring method of the VOS (Virtual Outlier Synthesis).

	FasterRCNN (Vanilla) — ID: BDD				**FasterRCNN (VOS) — ID: BDD**			
	COCO-Farther (OOD)		OpImg-Farther (OOD)		COCO-Farther (OOD)		OpImg-Farther (OOD)	
Method	AUC↑ (%)	FPR95↓ (%)	AUC↑ (%)	FPR95↓ (%)	AUC↑ (%)	FPR95↓ (%)	AUC↑ (%)	FPR95↓ (%)
ViM	91.4	39.3	91.6	39.3	92.9	32.3	93.1	31.5
Mahalanobis	89.5	48.8	89.0	51.5	91.1	43.3	90.6	46.7
MSP	80.0	77.7	81.2	76.8	79.1	79.4	80.0	76.6
Energy	72.4	64.4	73.3	60.3	—	—	—	—
ASH	48.9	81.0	49.0	77.3	67.6	70.6	71.7	61.4
DICE	68.3	69.2	69.3	65.0	77.7	57.9	71.6	49.0
ReAct	65.7	95.1	58.8	97.4	79.6	71.2	77.0	76.4
GEN	78.8	62.7	79.6	58.9	86.6	52.7	89.5	47.8
DICE+ReAct	57.9	97.7	48.5	98.9	66.8	90.5	59.4	95.5
DDU	90.8	41.6	91.5	42.6	92.2	37.2	92.9	40.1
VOS[B](Energy)	84.8	49.1	88.1	38.5	84.8	49.1	88.1	38.5
LaRD	**96.6**	**15.8**	**97.7**	**8.6**	**96.6**	**15.8**	**97.4**	**10.9**

Table 15: OOD detection performance for YOLOv8 (ID: PascalVOC). LaRD represents results from available PCA methods (KNN PCA 32 only in provided data). Higher AUC is better (↑), lower FPR95 is better (↓). Best result per metric column is in **bold**.

	YOLOv8 — PascalVOC							
	COCO-Near (OOD)		COCO-Far (OOD)		OpImg-Near (OOD)		OpImg-Far (OOD)	
Method	AUC↑ (%)	FPR95↓ (%)	AUC↑ (%)	FPR95↓ (%)	AUC↑ (%)	FPR95↓ (%)	AUC↑ (%)	FPR95↓ (%)
MSP	**85.2**	**64.0**	81.4	73.7	**85.1**	**67.4**	82.0	74.4
Energy	57.0	95.2	66.1	91.3	51.6	96.1	65.6	92.4
GEN	81.3	65.0	79.5	**67.2**	81.0	68.9	**82.3**	**59.1**
LaRD	78.6	76.4	**82.0**	68.8	71.4	85.7	80.9	75.7

Table 16: OOD detection performance on Farther OOD splits (ID: BDD). LaRD for RT-DETR represents best of (Mahalanobis PCA, KNN PCA, GMM PCA). Higher AUC is better (↑), lower FPR95 is better (↓). Best result for each metric column is in **bold**. '—' indicates data not available.

	YOLOv8 — ID: BDD				**RT-DETR — ID: BDD**			
	COCO-Farther (OOD)		OI-Farther (OOD)		COCO-Farther (OOD)		OI-Farther (OOD)	
Method	AUC↑ (%)	FPR95↓ (%)	AUC↑ (%)	FPR95↓ (%)	AUC↑ (%)	FPR95↓ (%)	AUC↑ (%)	FPR95↓ (%)
ViM	—	—	—	—	89.5	30.7	95.2	15.2
Mahalanobis	**98.2**	**7.8**	**99.6**	**1.3**	**99.1**	5.0	**99.7**	1.1
MSP	69.4	77.1	69.4	75.4	79.4	60.9	85.1	57.2
Energy	64.8	91.1	62.8	91.5	57.9	97.4	64.4	96.2
ASH	—	—	—	—	33.1	98.6	35.4	99.2
DICE	—	—	—	—	60.7	90.8	58.1	96.4
ReAct	—	—	—	—	56.5	96.8	63.2	95.0
GEN	63.8	71.9	66.8	68.8	77.1	67.9	83.8	63.3
DICE+ReAct	—	—	—	—	59.3	92.7	57.0	97.3
DDU	—	—	—	—	99.1	**3.5**	99.6	**0.6**
LaRD	—	—	—	—	98.8	5.3	99.4	1.4

Table 17: OOD detection performance for RT-DETR (ID: PascalVOC). LaRD represents best of (Mahalanobis PCA, KNN PCA, GMM PCA). Higher AUC is better (↑), lower FPR95 is better (↓). Best result per metric column is in **bold**.

RT-DETR — PascalVOC								
	COCO-Near (OOD)		COCO-Far (OOD)		OpenImages-Near (OOD)		OpenImages-Far (OOD)	
Method	AUC↑ (%)	FPR95↓ (%)	AUC↑ (%)	FPR95↓ (%)	AUC↑ (%)	FPR95↓ (%)	AUC↑ (%)	FPR95↓ (%)
ViM	**96.8**	10.9	**90.0**	**35.7**	74.1	59.7	87.7	39.7
Mahalanobis	96.6	**10.8**	87.2	42.4	**91.7**	32.6	**92.0**	**29.9**
MSP	94.2	21.7	84.5	58.3	62.7	79.0	76.7	67.3
Energy	68.1	97.7	70.8	92.6	50.1	96.3	62.8	96.7
ASH	64.7	86.2	57.5	92.9	46.8	96.6	49.3	94.3
DICE	63.4	89.7	70.7	83.0	81.9	73.7	81.3	78.2
ReAct	66.3	96.0	71.5	90.8	50.9	97.5	61.9	98.1
GEN	74.9	97.7	75.6	94.7	53.2	96.0	69.9	90.1
DICE+ReAct	68.0	90.0	70.1	84.1	81.5	75.4	79.0	83.0
DDU	96.4	11.9	86.7	45.2	91.2	**32.2**	91.4	31.5
LaRD	91.8	26.6	83.3	48.8	77.8	76.2	81.2	76.0

The evaluation using traditional OOD metrics (AUC/FPR95) reveals a significant method-architecture interaction effect on OOD discrimination performance. While certain methods like GEN demonstrate robust OOD separation on specific architectures (e.g., FasterRCNN), their efficacy is not universally transferable. Conversely, density-based methods like Mahalanobis show high sensitivity to the feature space, achieving exceptional discrimination in some contexts (e.g., YOLOv8/RT-DETR on BDD) but underperforming in others. This variability underscores that current OOD scoring functions often exploit specific architectural properties or data distributions rather than embodying a generalizable principle of OOD detection.

Across the presented experiments, traditional OOD detection metrics like AUC and FPR95 generally indicated that distinguishing out-of-distribution objects becomes less challenging as their semantic distance from the in-distribution data increases. This broad trend falsely suggests that greater dissimilarity simplifies the OOD object detection task. However, these metrics, while useful for gauging overall separability, offer limited insight into if these unknown objects are actually found, or the precision of their identification within an object detection framework.

F.2 Detailed results on the newly incorporated OSOD metrics

Table 18: OOD detection performance comparison on COCO splits (ID: PascalVOC). Lower nOSE is better (↓), higher AP_U/P_U/R_U is better (↑). Best result per metric column is in **bold**.

YOLOv8 — PascalVOC								
	COCO-Near (OOD)				COCO-Far (OOD)			
Method	nOSE↓	AP_U↑	P_U↑	R_U↑	nOSE↓	AP_U↑	P_U↑	R_U↑
MSP	32.3	8.5	62.0	11.0	18.7	4.5	**61.1**	5.7
Energy	43.6	1.3	44.3	3.0	23.2	1.0	34.8	2.7
GEN	27.2	11.7	64.7	16.5	14.2	6.3	59.3	10.1
LaRD	**24.9**	**13.7**	**67.8**	**18.8**	**11.4**	**7.0**	52.5	**12.7**

Table 19: OOD detection performance comparison on OpenImages splits (ID: PascalVOC). Lower nOSE is better (↓), higher $AP_U/P_U/R_U$ is better (↑). Best result per metric column is in **bold**.

	YOLOv8 — PascalVOC							
	OpenImages-Near (OOD)				OpenImages-Far (OOD)			
Method	nOSE↓	AP_U↑	P_U↑	R_U↑	nOSE↓	AP_U↑	P_U↑	R_U↑
MSP	26.2	6.2	**62.6**	7.6	13.8	2.1	52.3	3.1
Energy	34.3	0.9	41.2	2.0	15.9	0.8	42.0	1.8
GEN	**23.4**	**7.5**	62.2	**11.0**	**9.5**	4.0	52.1	6.9
LaRD	26.9	5.5	60.1	8.2	9.8	**4.1**	**52.8**	**7.0**

Table 20: OOD detection performance comparison on Far OOD sets (ID: BDD). Lower nOSE is better (↓), higher $AP_U/P_U/R_U$ is better (↑). Best result per metric column is in **bold**.

	YOLOv8 — BDD							
	COCO-Farther (OOD)				OpenImages-Farther (OOD)			
Method	nOSE↓	AP_U↑	P_U↑	R_U↑	nOSE↓	AP_U↑	P_U↑	R_U↑
MSP	4.6	0.3	31.4	1.0	4.0	0.6	**36.5**	1.2
Energy	5.3	0.1	26.0	0.4	5.0	0.1	22.6	0.3
GEN	3.9	0.6	**34.3**	1.7	3.2	0.8	36.0	2.0
LaRD	**0.1**	**1.6**	31.1	**4.8**	**0.0**	**1.4**	28.3	**4.7**

Table 21: OOD detection performance for FasterRCNN (Vanilla) on COCO splits (ID: PascalVOC). Lower nOSE is better (↓), higher $AP_U/P_U/R_U$ is better (↑). Best result per metric column is in **bold**. [B]Indicates the primary scoring method of the VOS (Virtual Outlier Synthesis).

	FasterRCNN (Vanilla) — PascalVOC							
	COCO-Near (OOD)				COCO-Far (OOD)			
Method	nOSE↓	AP_U↑	P_U↑	R_U↑	nOSE↓	AP_U↑	P_U↑	R_U↑
ViM	38.3	5.0	69.0	6.3	17.9	1.9	56.5	2.6
Mahalanobis	44.6	0.2	85.7	0.2	20.6	0.1	**100.0**	0.1
MSP	33.5	7.4	65.4	10.2	15.2	2.8	49.5	5.2
KNN	39.6	4.2	77.0	5.0	18.9	1.0	53.2	1.7
Energy	16.0	22.3	75.9	24.8	9.8	8.0	66.3	9.9
ASH	21.0	18.1	76.4	20.5	13.5	5.6	71.2	6.6
DICE	26.7	14.2	77.4	16.2	15.3	3.9	66.2	4.9
ReAct	33.3	10.0	**86.5**	10.8	19.0	1.3	83.3	1.5
GEN	**14.3**	**23.2**	73.8	**26.1**	**8.7**	8.6	65.2	11.0
DICE+ReAct	43.0	1.3	69.6	1.8	20.2	0.3	72.7	0.4
DDU	44.3	0.3	40.5	0.5	20.2	0.3	40.0	0.4
VOS^B(Energy)	20.5	21.5	72.1	24.6	9.6	**8.3**	55.6	**11.3**
LaRD	39.9	3.3	65.5	4.3	17.5	2.6	71.8	3.1

Table 22: OOD detection performance for FasterRCNN (Vanilla) on OpenImages splits (ID: PascalVOC). Lower nOSE is better (↓), higher AP_U/P_U/R_U is better (↑). Best result per metric column is in **bold**. [B]Indicates the primary scoring method of the VOS (Virtual Outlier Synthesis).

	FasterRCNN (Vanilla) — PascalVOC							
	OpenImages-Near (OOD)				OpenImages-Far (OOD)			
Method	nOSE↓	AP_U↑	P_U↑	R_U↑	nOSE↓	AP_U↑	P_U↑	R_U↑
ViM	30.6	3.1	66.0	4.1	11.7	0.8	59.7	1.1
Mahalanobis	35.0	0.2	**100.0**	0.2	12.7	0.0	0.0	0.0
MSP	28.4	4.1	59.9	6.1	9.7	1.7	53.0	2.9
KNN	33.5	1.0	57.5	1.7	11.7	0.9	62.0	1.1
Energy	18.1	12.9	73.6	15.2	5.8	5.3	70.1	6.6
ASH	21.1	10.7	75.3	12.5	7.8	3.8	69.9	4.6
DICE	23.0	9.7	75.8	11.0	9.0	2.9	70.0	3.5
ReAct	28.4	5.7	86.4	6.1	11.8	0.8	**78.7**	0.9
GEN	**15.9**	**14.1**	72.0	**16.9**	**5.4**	5.4	68.0	6.9
DICE+ReAct	33.4	1.5	82.4	1.7	12.7	0.0	50.0	0.0
DDU	34.5	0.5	51.6	0.6	12.6	0.1	46.2	0.1
VOS[B](Energy)	22.3	10.3	64.1	12.8	6.3	**5.5**	67.3	**7.1**
LaRD	31.1	3.4	74.6	3.7	10.0	2.2	68.8	2.6

Table 23: OOD detection performance for FasterRCNN (Vanilla) on Far OOD sets (ID: BDD). Lower nOSE is better (↓), higher AP_U/P_U/R_U is better (↑). Best result per metric column is in **bold**. [B]Indicates the primary scoring method of the VOS (Virtual Outlier Synthesis).

	FasterRCNN (Vanilla) — BDD							
	COCO-Farther (OOD)				OpenImages-Farther (OOD)			
Method	nOSE↓	AP_U↑	P_U↑	R_U↑	nOSE↓	AP_U↑	P_U↑	R_U↑
ViM	1.1	1.2	22.9	3.9	0.7	0.9	18.3	3.3
Mahalanobis	2.0	1.0	21.4	3.1	2.4	0.4	11.8	1.8
MSP	3.3	0.3	17.8	1.9	2.4	0.3	14.9	1.7
KNN	1.9	1.0	23.3	3.2	0.7	1.1	20.5	3.3
Energy	2.0	0.9	22.9	3.0	0.6	1.2	22.1	3.4
ASH	3.3	0.5	20.5	1.8	2.1	0.6	19.0	2.1
DICE	2.3	0.8	22.7	2.8	1.0	1.0	21.4	3.0
ReAct	4.0	0.4	17.9	1.2	3.6	0.1	7.7	0.6
GEN	2.0	1.0	22.9	3.0	0.7	1.2	21.8	3.3
DICE+ReAct	4.3	0.1	14.4	0.9	3.7	0.0	7.3	0.5
DDU	3.2	0.6	19.7	2.0	3.2	0.1	9.4	1.0
VOS[B](Energy)	1.8	**1.8**	**26.7**	**4.7**	**0.6**	**2.2**	**26.2**	**5.6**
LaRD	**0.7**	1.3	21.0	4.2	**0.6**	0.8	16.5	3.4

Table 24: OOD detection performance for FasterRCNN (VOS) on COCO splits (ID: PascalVOC). Lower nOSE is better (↓), higher AP_U/P_U/R_U is better (↑). Best result per metric column is in **bold**. [B]Indicates the primary scoring method of the VOS (Virtual Outlier Synthesis).

	FasterRCNN (VOS) — PascalVOC							
	COCO-Near (OOD)				COCO-Far (OOD)			
Method	nOSE↓	AP_U↑	P_U↑	R_U↑	nOSE↓	AP_U↑	P_U↑	R_U↑
ViM	45.5	3.1	64.0	4.3	20.3	1.3	48.9	2.2
Mahalanobis	50.0	0.0	0.0	0.0	22.6	0.0	0.0	0.0
MSP	39.6	7.0	66.6	9.6	17.6	2.6	44.5	4.7
KNN	36.3	10.3	73.9	12.2	14.9	4.7	55.7	6.9
ASH	17.6	23.3	71.3	26.5	9.5	8.7	60.4	11.4
DICE	33.0	12.4	73.6	15.2	14.1	5.1	56.6	7.4
ReAct	42.4	6.6	**83.6**	6.8	20.7	1.3	**66.0**	1.7
GEN	**15.9**	**24.1**	69.4	**27.8**	**8.0**	**9.1**	54.9	**12.7**
DICE+ReAct	50.0	0.0	0.0	0.0	22.6	0.0	0.0	0.0
DDU	49.8	0.2	46.7	0.2	22.3	0.2	25.0	0.3
VOS[B](Energy)	20.5	21.5	72.1	24.6	9.6	8.3	55.6	11.3
LaRD	42.1	6.0	70.7	7.1	19.9	2.1	64.5	2.5

Table 25: OOD detection performance for FasterRCNN (VOS) on OpenImages splits (ID: PascalVOC). Lower nOSE is better (↓), higher AP_U/P_U/R_U is better (↑). Best result per metric column is in **bold**. [B]Indicates the primary scoring method of the VOS (Virtual Outlier Synthesis).

	FasterRCNN (VOS) — PascalVOC							
	OpenImages-Near (OOD)				OpenImages-Far (OOD)			
Method	nOSE↓	AP_U↑	P_U↑	R_U↑	nOSE↓	AP_U↑	P_U↑	R_U↑
ViM	34.9	1.5	49.1	2.2	13.2	0.4	46.3	0.6
Mahalanobis	37.3	0.0	0.0	0.0	13.8	0.0	0.0	0.0
MSP	31.7	3.0	53.2	5.4	10.9	1.4	49.1	2.7
KNN	31.6	3.7	58.7	5.3	8.9	3.5	63.5	4.6
ASH	20.7	11.9	65.5	14.1	7.1	4.8	65.5	6.3
DICE	28.9	6.0	62.9	7.6	9.4	3.1	63.3	4.2
ReAct	32.2	4.4	**84.2**	4.7	12.7	0.9	**72.1**	1.1
GEN	**18.4**	**12.9**	63.3	**15.9**	**5.9**	**5.7**	66.1	**7.5**
DICE+ReAct	37.3	0.0	0.0	0.0	13.8	0.0	0.0	0.0
DDU	37.3	0.0	0.0	0.0	13.7	0.0	21.4	0.1
VOS[B](Energy)	22.3	10.3	64.1	12.8	6.3	5.5	67.3	7.1
LaRD	32.8	3.7	75.0	4.1	11.3	1.9	73.1	2.3

Table 26: OOD detection performance for FasterRCNN (VOS) on Far OOD sets (ID: BDD). Lower nOSE is better (↓), higher AP_U/P_U/R_U is better (↑). Best result per metric column is in **bold**. [B]Indicates the primary scoring method of the VOS (Virtual Outlier Synthesis).

	FasterRCNN (VOS) — BDD							
	COCO-Farther (OOD)				OpenImages-Farther (OOD)			
Method	nOSE↓	AP_U↑	P_U↑	R_U↑	nOSE↓	AP_U↑	P_U↑	R_U↑
ViM	1.1	1.7	24.1	5.3	0.8	1.7	23.1	5.5
Mahalanobis	2.3	1.3	22.1	4.3	3.3	0.8	17.8	3.4
MSP	4.4	0.5	19.9	2.4	3.9	0.6	21.9	2.9
KNN	2.3	1.5	24.8	4.2	0.8	2.0	26.6	5.5
ASH	3.6	1.0	23.7	3.0	2.4	1.6	26.7	4.1
DICE	2.8	1.4	25.9	3.8	1.1	**2.3**	**29.1**	5.2
ReAct	3.6	1.1	24.0	3.1	4.7	0.5	17.6	2.1
GEN	1.6	1.8	26.3	4.8	0.6	2.1	25.5	5.6
DICE+ReAct	5.4	0.3	16.7	1.4	5.9	0.1	13.4	1.0
DDU	3.8	0.7	20.6	2.9	4.8	0.3	14.9	2.1
VOS[B](Energy)	1.8	1.8	**26.7**	4.7	0.6	2.2	26.2	5.6
LaRD	**0.4**	**2.0**	23.5	**6.0**	**0.0**	1.8	21.8	**6.3**

Table 27: OOD detection performance for RT-DETR on COCO splits (ID: PascalVOC). Lower nOSE is better (↓), higher AP_U/P_U/R_U is better (↑). Best result per metric column is in **bold**.

	RT-DETR — PascalVOC							
	COCO-Near (OOD)				COCO-Far (OOD)			
Method	nOSE↓	AP_U↑	P_U↑	R_U↑	nOSE↓	AP_U↑	P_U↑	R_U↑
MSP	2.9	20.0	96.4	20.8	2.3	4.4	87.7	5.1
ViM	4.2	18.9	96.4	19.6	1.5	5.5	92.1	5.9
Mahalanobis	0.9	21.3	93.6	22.8	0.7	5.5	83.9	6.6
KNN	**0.8**	**21.4**	93.0	**23.0**	**0.4**	**5.6**	80.8	**6.8**
Energy	23.8	0.0	0.0	0.0	7.5	0.0	0.0	0.0
ASH	22.2	1.5	89.5	1.6	7.4	0.0	16.7	0.1
DICE	23.7	0.1	**100.0**	0.1	7.2	0.3	**100.0**	0.3
ReAct	23.8	0.0	0.0	0.0	7.4	0.1	**100.0**	0.1
GEN	23.8	0.0	0.0	0.0	7.5	0.0	0.0	0.0
DICE+ReAct	23.6	0.2	**100.0**	0.2	6.9	0.5	90.9	0.5
DDU	1.2	21.2	94.1	22.5	1.0	5.5	86.3	6.4
LaRD	5.7	17.1	95.4	17.9	3.2	3.2	75.7	4.2

Table 28: OOD detection performance for RT-DETR on OpenImages splits (ID: PascalVOC). Lower nOSE is better (↓), higher AP_U/P_U/R_U is better (↑). Best result per metric column is in **bold**.

	RT-DETR — PascalVOC							
	OpenImages-Near (OOD)				OpenImages-Far (OOD)			
Method	nOSE↓	AP_U↑	P_U↑	R_U↑	nOSE↓	AP_U↑	P_U↑	R_U↑
MSP	24.6	4.8	69.6	6.9	10.8	3.1	58.0	5.3
ViM	23.1	6.7	77.9	8.5	8.8	5.7	72.0	7.3
Mahalanobis	**6.0**	**21.8**	78.5	**24.8**	**2.9**	**9.6**	66.3	**12.6**
KNN	7.8	19.7	76.7	23.0	3.3	9.0	62.9	12.2
Energy	31.7	0.0	0.0	0.0	16.4	0.0	0.0	0.0
ASH	31.6	0.0	8.7	0.1	16.1	0.1	31.6	0.3
DICE	29.5	2.1	**91.5**	2.2	15.1	1.1	**81.0**	1.3
ReAct	31.7	0.0	0.0	0.0	16.4	0.0	0.0	0.0
GEN	31.7	0.0	0.0	0.0	16.4	0.0	0.0	0.0
DICE+ReAct	28.5	3.1	90.9	3.2	15.1	1.0	76.9	1.2
DDU	7.3	20.3	77.2	23.5	3.9	9.0	66.2	11.7
LaRD	27.3	3.0	66.5	4.2	13.2	1.6	47.9	3.0

Table 29: OOD detection performance for RT-DETR on Far OOD sets (ID: BDD). Lower nOSE is better (↓), higher AP_U/P_U/R_U is better (↑). Best result per metric column is in **bold**.

	RT-DETR — BDD							
	COCO-Farther (OOD)				OpenImages-Farther (OOD)			
Method	nOSE↓	AP_U↑	P_U↑	R_U↑	nOSE↓	AP_U↑	P_U↑	R_U↑
MSP	15.4	4.2	35.6	11.8	7.2	1.9	18.3	10.1
ViM	14.1	4.8	34.5	12.8	5.4	1.7	14.4	11.6
Mahalanobis	**0.2**	11.2	33.0	**25.0**	**0.0**	2.3	12.4	**14.9**
KNN	0.4	11.4	33.0	24.8	**0.0**	2.4	12.5	**14.9**
Energy	28.6	0.0	0.0	0.0	20.6	0.0	0.0	0.0
ASH	28.5	0.0	19.1	0.1	20.6	0.0	10.5	0.0
DICE	27.9	0.5	**77.5**	0.7	20.5	0.0	**36.4**	0.1
ReAct	28.6	0.0	7.1	0.0	20.6	0.0	25.0	0.0
GEN	27.9	0.4	54.2	0.7	20.0	0.2	33.9	0.4
DICE+ReAct	27.7	0.7	70.9	0.9	20.6	0.0	25.0	0.1
DDU	0.6	**11.5**	34.1	24.7	**0.0**	**2.4**	12.6	**14.9**
LaRD	0.8	10.7	32.3	24.4	**0.0**	2.3	12.4	**14.9**

Looking at the results, we don't find a universally best method, neither across architecture nor across semantic distance, e.g: GEN frequently demonstrates strong performance on FasterRCNN (Vanilla

and VOS) and YOLOv8 when PascalVOC is the ID, often achieving leading nOSE, AP_U, and R_U values. However, its efficacy sharply declines on the RT-DETR architecture with PascalVOC as the ID. Energy, particularly its VOS variant on FasterRCNN and for Far OOD scenarios on BDD, shows competitive results but generally struggles on YOLOv8 and RT-DETR (ID: PascalVOC), characterized by high nOSE and poor recall of unknowns (R_U). LaRD's performance is more varied; it excels on YOLOv8 (especially for Far OOD BDD splits) and demonstrates strength on FasterRCNN for BDD Far OOD detection tasks, often leading in nOSE, AP_U, and R_U. Conversely, its effectiveness is less prominent on FasterRCNN and RT-DETR architectures when trained on PascalVOC. This work also highlights the performance volatility of OOD-OD methods and offers a comprehensive comparative analysis across architectures and semantic similarity.

The introduction of OSOD metrics (nOSE, AP_U, P_U, R_U) provides a much more nuanced understanding of performance related to semantic distance. These metrics reveal that even if general OOD discrimination (AUC/FPR95) seems satisfactory, the actual ability to comprehensively find OOD objects remains unknown. This challenges the intuition that greater dissimilarity inherently makes all aspects of OOD object detection easier.

F.3 Correlations among metrics

Additionally, in Figure 15 it is possible to find the empirical matrix of correlations among all (old and new) metrics. This matrix is calculated from the overall results previously presented. It shows correlations among metrics across all methods, architectures, and OOD datasets. The figure indicates in general significant but moderate correlations between old metrics and new ones, meaning that the AUROC and FPR95 can be indicative of the performance of OOD-OD methods for finding unknown objects. However the correlations don't have a high absolute value (minimum 0.56 an maximum 0.70), which means that new information is added by the new metrics.

Moreover the results indicate that there is no correlation found between old metrics (AUC & FPR95) and P_U. This means the P_U is orthogonal to the previous metrics, and therefore the information measured by P_U is invisible to the old metrics. This reinforces the utility of adding OSOD metrics to the benchmark.

G Further discussion on the similarities and differences between OOD-OD and OSOD methods

Building upon the detailed presentation of how Out-of-Distribution Object Detection (OOD-OD) methods operate in Section 2 and Section D, which draws from previous works Du et al. (2022b); Wilson et al. (2023); Ammar et al. (2024); Han et al. (2022), we can conclude that the two approaches for handling unknown objects in object detectors are distinct yet they are like two sides of the same coin.

In simpler terms, the current formulation of OOD-OD serves as a monitoring function for the base object detector. It aims to verify that the detected objects are indeed In-Distribution (ID) categories, rather than actively seeking out unknown objects in images. Nevertheless, it *can* identify unknown objects and label them as Out-of-Distribution (OOD). The ability of OOD-OD methods to detect objects was not assessable in the previous benchmark, but it can now be quantified precisely using the new FMIYC benchmark, which employs OSOD metrics calculated with respect to the ground truth labels.

Conversely, open set object detection (OSOD) methods do not rely on monitoring functions. Instead, they incorporate an "unknown" class directly into the object detector, adding specific loss terms and usually training with labeled or pseudo-labeled examples of "unknowns" Joseph et al. (2021); Dhamija et al. (2020); Gupta et al. (2022). OSOD has developed several metrics, already presented in Section 4.2 and Section C, to measure how well OSOD methods can identify and localize both unknown and known objects simultaneously. The OOD-OD community lacks this type of evaluation, which we believe can significantly enrich the field and is provided by the present benchmark.

We believe the OOD-OD field has substantial potential for future developments, particularly in enhancing a method's ability to localize unknown objects. The main bottleneck is perhaps the filtering

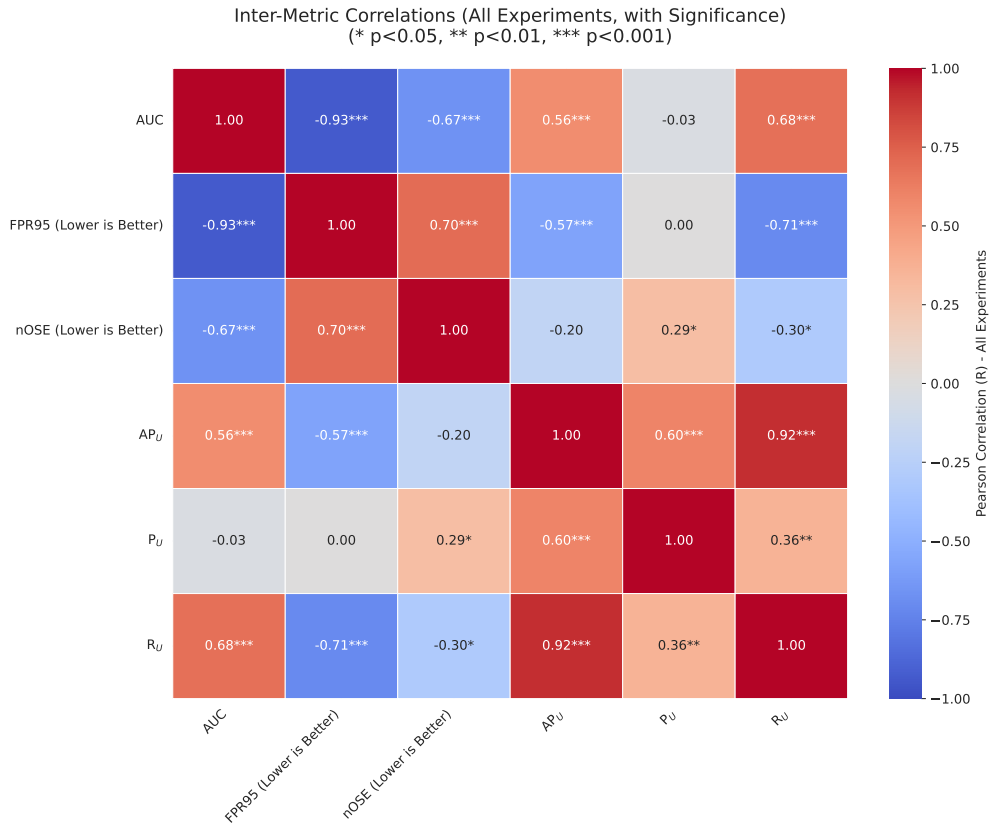


Figure 15: Empirical correlations among old and new metrics.

of predictions by the confidence threshold in the base model $\mathcal{M}$ because the model is trained to ignore unknown objects. Therefore, finding ways to encourage models to retrieve more predictions that will be post-processed anyway by OOD scoring functions can be an interesting research direction. This could be done perhaps by adjusting the confidence threshold t^* so that a model can retrieve more objects, rather than just maximizing the mAP of the ID test dataset.

Another research direction that may impact the field is the development of OOD-Od or OSOD methods for VLMs, which have broader semantic knowledge and, therefore, may be able to localize several categories of objects beyond a definite set of ID classes. In any case, precise detection of unknown objects must be rigorously evaluated, since this capability is crucial for applications beyond identifying incorrect predictions. Without proper evaluation, OOD-OD methods lack a realistic assessment of their performance for real-world scenarios.

H Societal Impact

This work fosters positive societal impacts by enhancing the safety and trustworthiness of object detection systems in safety-critical applications like autonomous driving and medical imaging. By providing a more rigorous benchmark and nuanced metrics for evaluating how well systems detect out-of-distribution objects, it helps prevent overconfidence in deployed models and pushes the field towards developing AI that is more trustworthy and reliable. However, as systems improve in identifying "unknown" or "novel" entities through enhanced evaluations like this, there are several potential downsides to consider. Enhanced capabilities in detecting unspecified "unknowns" could inadvertently enable more pervasive or intrusive surveillance systems, potentially tracking atypical (though not necessarily illicit) activities or objects without clear justification. Furthermore, if the definition of "known" within the training data or benchmark inherently contains biases, such as curation biases, objects or individuals deviating from these biased norms might be disproportionately

flagged as"unknown," leading to unfair scrutiny or misclassification for certain groups. There's also a risk that an over-reliance on these improved systems, even with better benchmarking, could lead to a false sense of safety & security, potentially delaying human intervention when truly critical and unanticipated failures occur, or encouraging the deployment of systems in environments where the range of true "unknowns" far exceeds what any benchmark can capture *i.e.*, existence of *unknown-unknowns* in the wild real-word that cannot be foreseeing by any evaluation benchmark.

I DATASHEET FOR DATASETS

Upon acceptance, we will provide the dataset datasheet as suggested by Gebru et al. (2021).