YolOOD: Utilizing Object Detection Concepts for Multi-Label Out-of-Distribution Detection Supplementary Material

A. Background

A.1. YOLO Object Detector

In this paper, we focus on the state-of-the-art one-stage YOLO object detector and leverage its capabilities. Since the emergence of YOLO's first version [11], other versions have been published to further enhance its performance [1, 5, 9, 10]. However, since YOLOv3 [10], the architectural design of the network has not been altered substantially, and this design serves as the foundation for the recent state-of-the-art versions (*e.g.*, YOLOv4 [1], YOLOv5 [5]). Our proposed approach utilizes the latest version, YOLOv5.

Architecture. YOLO's architecture is comprised of two components: (a) a backbone network used to extract features from the input image, and (b) three detection heads which process the image's features at three different scales. These components are connected using the feature pyramid network (FPN) [8] topology, where feature maps from different blocks of the backbone are concatenated to feature maps of corresponding sizes in the detection heads. The size of a detection head is determined by the size of the input image and the network's stride (downsampling factor) – 32, 16, and 8. This allows the network to detect objects of different sizes: the first detection head (with the largest stride) has a broader context, specializing in the detection of large objects, while the smallest one has finer resolution and specializes in the detection of small objects.

Detection layer. The last layer of each detection head predicts a 3D tensor of size $W \times H \times (4 + 1 + N_c)$, where $W \times H$ is the grid size (W and H are the width and height, respectively) and $4 + 1 + N_c$ (which we refer to as a *candidate*) encodes three parts:

- Bounding box offsets - four coordinate offsets from a predefined anchor box.
- Objectness score - a value that represents the model's confidence that the bounding box contains an object.
- Class scores - N_c confidence scores indicating the presence of specific class categories.

To be more precise, every cell in the grid predicts three bounding boxes (associated with three predefined anchor boxes), resulting in a $3 \times W \times H \times (4+1+N_c)$ prediction.

Training YOLO. Every ground-truth object is associated with a single cell in each detection head. The input image is divided into an $W \times H$ grid, and the responsible cell is determined by the grid cell that the object's center falls in. Since each cell contains three candidates, each of which is associated with a different predefined anchor box, the specific candidate is determined to be the one with the highest intersection over union (IoU) value between the ground-truth bounding box and the candidate's anchor box.

YOLO does not assume that the class categories are mutually exclusive; therefore the class score vector is trained with the multi-label configuration (the sigmoid function is applied on each output neuron).

Postprocessing. YOLO outputs a fixed number of candidates (the amount depends on the size of the input image) which are then filtered in three sequential steps:

- Objectness score filtering - only candidates whose objectness score exceeds a predefined threshold are passed to the next step.
- Class score filtering - only candidates that have at least one unconditional class score ($Pr(Objectness) \cdot Pr(Class)$) that exceeds the predefined threshold are passed to the next step.
- Non-maximum suppression (NMS) - since many candidates can detect the same object, NMS is applied to reduce redundancy.

B. OOD Detection Methods Examined

Let f be a multi-label image classifier with a shared parameter space θ up to the penultimate feature space. For an input x, the output for the n$^{\text{th}}$ class is:

$$f_{y_n}(x) = h(x;\theta) \cdot w_n \tag{1}$$

where $h(x;\theta)$ is the feature vector in the penultimate layer and w_n is a vector that represents the weights corresponding to class n. The predictive probability of a binary label y_n is determined by a binary logistic classifier:

$$p(y_n = 1|x) = \frac{exp(f_{y_n}(x))}{1 + exp(f_{y_n}(x))} \tag{2}$$

where $n \in \{1, .., N_c\}$.

B.1. Baseline Methods

The baselines (MaxLogit [4] and MSP [3]) considered in our paper can be formalized as follows:

$$\textbf{MaxLogit} = \max_n f_{y_n}(x) \tag{3}$$

$$\textbf{MSP} = \max_n \frac{exp(f_{y_n}(x))}{\sum_n^{N_c} exp(f_{y_n}(x))} \tag{4}$$

B.2. State-of-the-Art Methods for OOD Detection in the Multi-Class Domain

We consider two state-of-the-art methods originally designed for the multi-class task: ODIN [7] and Mahalanobis [6]. These methods are adapted to the multi-label case (following the adaption proposed in [13]).

ODIN. The adapted ODIN score takes the maximum of the calibrated label-wise predictions and is formulated as follows:

$$\textbf{ODIN} = \max_n \frac{exp(f_{y_n}(x)/T)}{1 + exp(f_{y_n}(x)/T)} \tag{5}$$

Since ODIN also proposes input preprocessing (by adding small adversarial perturbations), the input is calculated using $\hat{x} = x - \epsilon \cdot \text{sign}(-\nabla \ell_{\hat{y}_n})$, where $\ell_{\hat{y}_n}$ is the BCE loss for the label $\hat{y}_n$ with the largest output, such that $\hat{y}_n = \text{argmax}_n(p(y_n = 1|x))$.

Mahalanobis. The Mahalanobis score is represented in the following way:

$$\mathbf{M}(x) = \max_n -(\phi(x) - \hat{\mu}_{y_n})^\intercal \hat{\Sigma}^{-1} (\phi(x) - \hat{\mu}_{y_n}) \tag{6}$$

where the feature embedding $\phi(x)$ is extracted for a given sample, $\hat{\mu}_{y_n}$ is the class mean for label y_n, and $\hat{\Sigma}^{-1}$ is the covariance matrix. Furthermore, we consider the calibration techniques proposed in the original paper [6]: (a) input preprocessing, and (b) feature ensemble. For preprocessing, the perturbed sample is defined as:

$$\begin{aligned} \hat{x} &= x + \epsilon \,\text{sign}(\nabla_x M(x)) = \\ &= x - \epsilon \,\text{sign}(\nabla(\phi(x) - \hat{\mu}_{y_n})^\intercal \hat{\Sigma}^{-1} (\phi(x) - \hat{\mu}_{y_n})) \end{aligned} \tag{7}$$

For feature ensemble, the scores are extracted from each block and integrated by $\sum_\ell \alpha_\ell M_\ell(\hat{x})$, where ℓ denotes the ℓ-th block. In our evaluation, we use equal weights for all layers (*i.e.*, $\alpha_\ell = 1$).

B.2.1 Hyperparameter Selection

Since ODIN and Mahalanobis both require validation sets for hyperparameter selection, we consider the following setup proposed in [13] (corrupted in-distribution samples into OOD data):

- Pixel-wise arithmetic mean of random pairs of in-distribution images. Given two images x_1 and x_2, the value of a pixel at location (i, j) in the new image is:

$$x_{\text{arth mean}}(i,j) = \frac{x_1(i,j) + x_2(i,j)}{2} \tag{8}$$

- Geometric mean of random pairs of in-distribution images. Given two images x_1 and x_2, the value of a pixel at location (i, j) in the new image is:

$$x_{\text{geom mean}}(i,j) = \sqrt{x_1(i,j) \cdot x_2(i,j)} \tag{9}$$

- Random permutation of 16 equally sized patches of an in-distribution image.

These datasets are used to fine-tune the hyperparameters: magnitude of noise ϵ for both and temperature T for ODIN. For ODIN, $T \in \{1, 10, 100, 1000\}$, and ϵ is chosen from 21 evenly spaced in the range of $[0, 0.004]$. For Mahalanobis, ϵ is chosen from $[0, 0.0005, 0.001, 0.002, 0.005]$. The optimal ϵ and T values are the ones that minimize the FPR95.

B.3. State-of-the-Art Methods for OOD Detection in the Multi-Label Domain

JointEnergy. We also consider the state-of-the-art multi-label OOD detection method JointEnergy [13]:

$$\begin{aligned} E_{y_n}(x) &= -\log(1 + exp(f_{y_n}(x))) \\ E_{\text{joint}}(x) &= \sum_{n=1}^{N_c} -E_{y_n}(x) \end{aligned} \tag{10}$$

C. Proposed Datasets

C.1. In-Distribution Dataset

Objects365$_{\text{in}}$. We propose a new in-distribution benchmark for OOD detection in the multi-label domain, which is a subset of the Objects365 dataset [12]. The specific classes, which are chosen according to their frequency in the dataset (*i.e.*, the 20 most common classes that do not overlap with the OOD dataset), are: *person, chair, car, boat, wild bird, bench, sailboat, bottle, potted plant, cup, handbag/satchel, dog, bus, train, umbrella, cow, airplane, cat, truck, and horse*. We divided the images into three sets consisting of 68,723, 5,000, and 10,000 images for the training, validation, and test sets, respectively.

C.2. OOD Datasets

We propose two new benchmarks constructed from datasets that contain images associated with multiple class categories and instances, thus reflecting the complexity of the multi-label setting: a subset from the Objects365 [12] dataset and a subset from the NUS-WIDE dataset [2].

Objects365$_{\text{out}}$. A subset of 241 classes which do not overlap with any of the classes present in the in-distribution

datasets, and specifically with the classes present in the Objects365$_{\text{in}}$ subset presented above, containing a total of 11,669 images. For example, The top-10 most frequent classes are (sorted in descending order): *lamp, street lights, storage box, picture/frame, cabinet/shelf, flag, air conditioner, sneakers, trash bin can, fish*. The full list of the class can be found online.[1]

NUS-WIDE$_{\text{out}}$. A subset of 54 classes that do not overlap with any of the classes in the in-distribution datasets containing 13,149 images. The specific classes are: *beach, bridge, buildings, castle, cityscape, clouds, coral, earthquake, elk, fire, fish, flags, flowers, fox, frost, garden, glacier, grass, harbor, house, lake, leaf, map, military, moon, mountain, nighttime, ocean, plants, police, railroad, rainbow, reflection, road, rocks, sand, sign, snow, sports, statue, street, sun, sunset, temple, tiger, tower, town, toy, tree, valley, water, waterfall, whales, window*.

D. Results

D.1. Effect of Responsible Grid Cell Percentage p_k

As discussed in Section 4.2 in the paper, we characterize the effect of the percentage $\{p_k | k \in \{1, 2, 3\}\}$ of responsible cells (described in Section 3.1), where p_1 (resp. p_3) represents the smallest (resp. largest) detection head. We perform an extensive evaluation to examine the effect of different p_k combinations, where p_k is selected from 11 evenly spaced numbers in the range $[0, 1]$. To limit the number of possible combinations, we set a constraint such that $p_3 > p_2 > p_1$, based on the fact that the grid's resolution increases in each subsequent detection head, resulting in 165 different combinations. Figure 1 presents the mAP results for the different combinations of p_k across the different datasets. It is interesting to observe the consistent pattern present across all datasets, where we can see that p_1 benefits most from the lower range ($\sim [0.0 - 0.2]$), p_2 benefits from slightly higher range ($\sim [0.1 - 0.4]$), and p_3 benefits from the upper range ($\sim [0.3 - 0.7]$). Furthermore, as stated in Section 3.1, the results confirm that since bounding boxes do not accurately segment the object's area, setting $p_k = 1$ degrades the model's performance, likely due to the background areas included in the annotated bounding boxes. To obtain the optimal combination, after training, we sort all the models according to their in-distribution mAP and select the 20 best-performing ones. Then, we count the number of occurrences for all p_k values (each p_k is counted independently, not as a triplet) and select the most frequent values. We aggregate the results over all of the in-distribution datasets and find that the best configuration is: $(p_1, p_2, p_3) = (0.0, 0.1, 0.5)$. We recommend using this configuration for all datasets, *i.e.*, p_k is not a hyperparameter that should be tuned.

[1] https://github.com/AlonZolfi/YolOOD

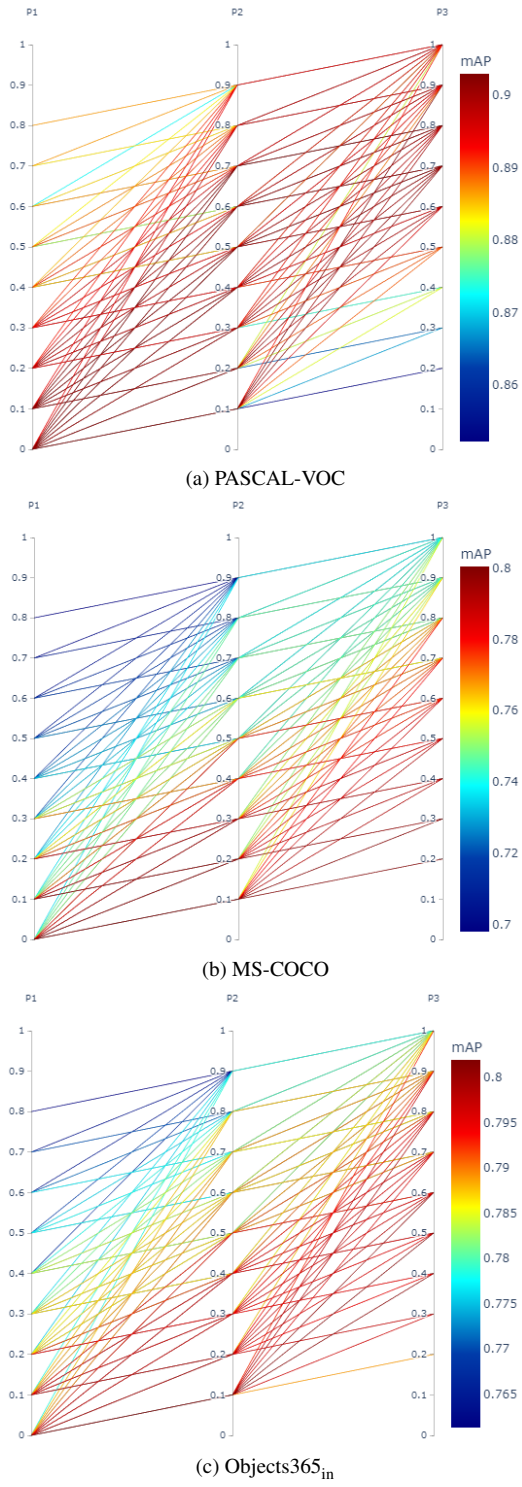

Figure 1. Models' mAP for different p_k combinations on the (a) PASCAL-VOC, (b) MS-COCO, and (c) Objects365$_{\text{in}}$ datasets. Each line represents a model trained with a single combination.

D.2. In-Distribution mAP

For a comprehensive comparison of OOD detection performance, we also provide the in-distrubtion mAP results for the YOLO-cls and YolOOD models. Table 1 presents the results on the various in-distribution datasets, showing that YolOOD's mAP is on par with that of YOLO-cls. Despite the slight differences, in the OOD detection task, YolOOD outperforms all state-of-the-art OOD detection methods (which use YOLO-cls), demonstrating the superiority of our proposed approach. It should be noted that when training the networks from "scratch" (*i.e.*, no pre-trained weights are used), YolOOD achieves substantially better results on both the in-distribution mAP and OOD detection metrics. However, for a fair comparison between the different methods, we followed the training scheme of using a pre-trained backbone.

Model \ $\mathcal{D}_{\text{in}}$	PASCAL-VOC	MS-COCO	Objects365$_{\text{in}}$
YOLO-cls	91.09	81.83	81.53
YolOOD-a[1]	88.64	76.81	78.48
YolOOD-o[2]	89.15	78.97	79.71

Table 1. In-distribution mAP comparison on the YOLO-cls and YolOOD models on the in-distribution datasets. [1]Trained using the auto-generated annotations. [2]Trained using the original annotations.

D.3. Additional Results

In addition to the OOD detection results presented in the paper, we provide the full results of all the experiments conducted in our evaluation, including both the mean and standard deviation.

Table 2 presents the OOD detection performance of YolOOD vs. state-of-the-art OOD detection methods.

Table 3 presents the OOD detection performance when using different combinations of aggregation functions for YolOOD's detection heads and class scores output vector.

Table 4 presents the OOD detection performance when using JointEnergy on YolOOD and YOLO-cls networks.

Table 5 presents the OOD detection performance between YolOOD and a standard YOLO object detector.

Table 2. Comparison of the OOD detection performance of YolODD vs. state-of-the-art methods. ↓ indicates that lower values are better, and ↑ indicates that higher values are better.
[1]Trained using the auto-generated annotations. [2]Trained using the original annotations.

	$\mathcal{D}_{\text{in}}$	PASCAL-VOC	MS-COCO	Objects365$_{\text{in}}$
$\mathcal{D}_{\text{out}}$	**Method**	**FPR95 ↓ / AUROC ↑ / AUPR ↑**		
Objects365$_{\text{out}}$	MaxLogit [4]	28.91 ± 0.71 / 94.96 ± 0.11 / 95.32 ± 0.16	16.39 ± 0.58 / 96.90 ± 0.09 / 99.17 ± 0.03	29.95 ± 0.81 / 94.33 ± 0.21 / 94.38 ± 0.30
	MSP [3]	50.78 ± 1.50 / 88.36 ± 0.37 / 88.61 ± 0.39	46.25 ± 0.16 / 86.78 ± 0.28 / 95.63 ± 0.13	65.20 ± 0.88 / 83.99 ± 0.42 / 84.13 ± 0.59
	ODIN [7]	28.91 ± 0.71 / 94.96 ± 0.11 / 95.32 ± 0.16	16.39 ± 0.58 / 96.90 ± 0.09 / 99.17 ± 0.03	29.95 ± 0.81 / 94.33 ± 0.21 / 94.38 ± 0.30
	Mahalanobis [6]	73.34 ± 0.45 / 73.90 ± 0.18 / 70.94 ± 0.25	87.69 ± 0.13 / 52.13 ± 0.15 / 77.53 ± 0.11	83.30 ± 0.23 / 63.23 ± 0.15 / 56.69 ± 0.19
	JointEnergy [13]	27.90 ± 1.29 / 95.37 ± 0.18 / 96.04 ± 0.13	14.80 ± 0.40 / 97.16 ± 0.07 / 99.28 ± 0.02	23.13 ± 0.42 / 95.84 ± 0.12 / 96.20 ± 0.17
	YolOOD-a[1]	18.37 ± 0.51 / 96.10 ± 0.20 / 95.85 ± 0.32	11.70 ± 0.15 / 97.21 ± 0.01 / 99.19 ± 0.01	18.40 ± 0.66 / 95.76 ± 0.10 / 95.15 ± 0.13
	YolOOD-o[2]	16.38 ± 0.70 / 96.60 ± 0.22 / 96.50 ± 0.28	11.53 ± 0.22 / 97.30 ± 0.05 / 99.23 ± 0.02	17.24 ± 0.33 / 95.97 ± 0.06 / 95.42 ± 0.06
NUS-WIDE$_{\text{out}}$	MaxLogit [4]	23.60 ± 1.27 / 95.99 ± 0.21 / 96.05 ± 0.19	12.16 ± 0.27 / 97.53 ± 0.06 / 99.24 ± 0.02	38.07 ± 2.62 / 92.62 ± 0.37 / 91.48 ± 0.39
	MSP [3]	47.34 ± 1.61 / 89.34 ± 0.42 / 88.71 ± 0.41	40.89 ± 0.62 / 88.33 ± 0.22 / 95.53 ± 0.10	78.08 ± 1.20 / 78.42 ± 0.63 / 76.91 ± 0.62
	ODIN [7]	23.60 ± 1.27 / 95.99 ± 0.21 / 96.05 ± 0.19	12.16 ± 0.27 / 97.53 ± 0.06 / 99.24 ± 0.02	38.07 ± 2.62 / 92.62 ± 0.37 / 91.48 ± 0.39
	Mahalanobis [6]	77.23 ± 0.31 / 73.76 ± 0.35 / 67.76 ± 0.53	88.74 ± 0.13 / 60.35 ± 0.22 / 80.85 ± 0.14	88.54 ± 0.14 / 62.34 ± 0.17 / 54.12 ± 0.21
	JointEnergy [13]	20.19 ± 1.78 / 96.53 ± 0.22 / 96.76 ± 0.18	8.29 ± 0.23 / 97.90 ± 0.04 / 99.39 ± 0.01	24.46 ± 1.55 / 95.34 ± 0.21 / 94.96 ± 0.24
	YolOOD-a[1]	21.24 ± 0.78 / 96.29 ± 0.09 / 96.08 ± 0.16	7.62 ± 0.37 / 98.13 ± 0.07 / 99.43 ± 0.03	12.19 ± 0.73 / 97.64 ± 0.12 / 97.29 ± 0.14
	YolOOD-o[2]	18.48 ± 0.76 / 96.85 ± 0.16 / 96.77 ± 0.22	4.40 ± 0.12 / 98.57 ± 0.03 / 99.58 ± 0.01	9.54 ± 0.57 / 98.01 ± 0.05 / 97.61 ± 0.06

Table 3. OOD detection performance when using different combinations of aggregation functions on YolOOD's detection heads and class scores output vector. ↓ indicates that lower values are better, and ↑ indicates that higher values are better.

		$\mathcal{D}_{\text{in}}$	PASCAL-VOC	MS-COCO	Objects365$_{\text{in}}$
$\mathcal{D}_{\text{out}}$	**Class Agg.**	**Head Agg.**	**FPR95 ↓ / AUROC ↑ / AUPR ↑**		
Objects365$_{\text{out}}$	Max	Max	20.06 ± 0.68 / 96.29 ± 0.16 / 96.52 ± 0.20	8.54 ± 0.18 / 97.97 ± 0.03 / 99.46 ± 0.01	20.55 ± 0.62 / 96.01 ± 0.02 / 95.96 ± 0.03
		Multiply	23.19 ± 1.92 / 95.11 ± 0.50 / 94.89 ± 0.57	15.62 ± 0.39 / 96.69 ± 0.08 / 99.06 ± 0.03	26.05 ± 1.01 / 94.56 ± 0.25 / 93.99 ± 0.23
		Sum	16.38 ± 0.70 / 96.60 ± 0.22 / 96.50 ± 0.28	11.53 ± 0.22 / 97.30 ± 0.05 / 99.23 ± 0.02	17.24 ± 0.33 / 95.97 ± 0.06 / 95.42 ± 0.06
	Sum	Max	40.04 ± 1.80 / 82.03 ± 1.14 / 77.46 ± 1.54	53.25 ± 0.61 / 82.31 ± 0.26 / 93.63 ± 0.15	38.86 ± 0.85 / 84.24 ± 0.16 / 78.66 ± 0.29
		Multiply	23.15 ± 1.79 / 95.11 ± 0.51 / 94.94 ± 0.58	14.76 ± 0.43 / 96.98 ± 0.07 / 99.17 ± 0.02	25.71 ± 1.14 / 94.70 ± 0.25 / 94.27 ± 0.24
		Sum	37.31 ± 2.19 / 86.73 ± 1.05 / 85.04 ± 1.19	41.27 ± 0.71 / 89.57 ± 0.18 / 96.71 ± 0.07	35.22 ± 0.80 / 90.85 ± 0.19 / 89.25 ± 0.19
NUS-WIDE$_{\text{out}}$	Max	Max	28.14 ± 1.03 / 94.97 ± 0.37 / 94.70 ± 0.52	8.55 ± 0.21 / 98.05 ± 0.04 / 99.40 ± 0.01	21.48 ± 0.64 / 95.40 ± 0.19 / 94.19 ± 0.26
		Multiply	16.32 ± 0.53 / 96.92 ± 0.19 / 96.60 ± 0.25	5.48 ± 0.20 / 98.49 ± 0.04 / 99.55 ± 0.01	10.40 ± 0.32 / 97.90 ± 0.10 / 97.36 ± 0.13
		Sum	18.48 ± 0.76 / 96.85 ± 0.16 / 96.77 ± 0.22	4.40 ± 0.12 / 98.57 ± 0.03 / 99.58 ± 0.01	9.54 ± 0.57 / 98.01 ± 0.05 / 97.61 ± 0.06
	Sum	Max	49.09 ± 2.06 / 74.59 ± 1.79 / 64.33 ± 2.83	56.15 ± 0.46 / 78.87 ± 0.24 / 90.78 ± 0.16	31.66 ± 1.28 / 87.09 ± 0.65 / 79.90 ± 0.86
		Multiply	16.01 ± 0.61 / 96.97 ± 0.19 / 96.66 ± 0.24	4.45 ± 0.13 / 98.64 ± 0.03 / 99.60 ± 0.01	9.39 ± 0.39 / 98.08 ± 0.09 / 97.61 ± 0.12
		Sum	42.90 ± 2.29 / 83.08 ± 1.37 / 78.62 ± 1.84	37.73 ± 0.45 / 90.83 ± 0.12 / 96.72 ± 0.05	17.13 ± 1.48 / 96.51 ± 0.26 / 95.54 ± 0.31

Table 4. OOD detection performance comparison when applying JointEnergy on YolOOD and YOLO-cls networks. ↓ indicates lower values are better, and ↑ indicates higher values are better.

	$\mathcal{D}_{\text{in}}$	PASCAL-VOC	MS-COCO	Objects365$_{\text{in}}$
$\mathcal{D}_{\text{out}}$	**Model**	**FPR95 ↓ / AUROC ↑ / AUPR ↑**		
Objects365$_{\text{out}}$	YOLO-cls	27.90 ± 1.29 / 95.37 ± 0.18 / 96.04 ± 0.13	14.80 ± 0.40 / 97.16 ± 0.07 / 99.28 ± 0.02	23.13 ± 0.42 / 95.84 ± 0.12 / 96.20 ± 0.17
	YolOOD	17.41 ± 0.81 / 96.77 ± 0.16 / 97.04 ± 0.18	7.10 ± 0.13 / 98.20 ± 0.01 / 99.54 ± 0.01	15.91 ± 0.41 / 96.93 ± 0.03 / 96.97 ± 0.03
NUS-WIDE$_{\text{out}}$	YOLO-cls	20.19 ± 1.78 / 96.53 ± 0.22 / 96.76 ± 0.18	8.29 ± 0.23 / 97.90 ± 0.04 / 99.39 ± 0.01	24.46 ± 1.55 / 95.34 ± 0.21 / 94.96 ± 0.24
	YolOOD	23.66 ± 1.40 / 95.88 ± 0.34 / 95.72 ± 0.43	4.65 ± 0.24 / 98.57 ± 0.02 / 99.58 ± 0.01	11.51 ± 0.66 / 97.77 ± 0.09 / 97.37 ± 0.11

Table 5. OOD detection performance comparison between YolOOD and a regular YOLO detector. ↓ indicates lower values are better, and ↑ indicates higher values are better.

	$\mathcal{D}_{\text{in}}$	PASCAL-VOC	MS-COCO	Objects365$_{\text{in}}$
$\mathcal{D}_{\text{out}}$	Model	**FPR95 ↓ / AUROC ↑ / AUPR ↑**		
Objects365$_{\text{out}}$	YOLO	40.57 ± 1.02 / 92.16 ± 0.26 / 92.66 ± 0.25	20.03 ± 0.13 / 96.40 ± 0.05 / 99.02 ± 0.02	28.64 ± 0.86 / 94.06 ± 0.10 / 93.66 ± 0.12
	YolOOD	16.38 ± 0.70 / 96.60 ± 0.22 / 96.50 ± 0.28	11.53 ± 0.22 / 97.30 ± 0.05 / 99.23 ± 0.02	17.24 ± 0.33 / 95.97 ± 0.06 / 95.42 ± 0.06
NUS-WIDE$_{\text{out}}$	YOLO	29.90 ± 2.29 / 94.54 ± 0.34 / 94.24 ± 0.32	7.20 ± 0.23 / 98.27 ± 0.02 / 99.48 ± 0.01	17.99 ± 0.86 / 96.43 ± 0.11 / 95.60 ± 0.11
	YolOOD	18.48 ± 0.76 / 96.85 ± 0.16 / 96.77 ± 0.22	4.40 ± 0.12 / 98.57 ± 0.03 / 99.58 ± 0.01	9.54 ± 0.57 / 98.01 ± 0.05 / 97.61 ± 0.06

References

[1] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020. 1

[2] Tat-Seng Chua, Jinhui Tang, Richang Hong, Haojie Li, Zhiping Luo, and Yantao Zheng. Nus-wide: a real-world web image database from national university of singapore. In *Proceedings of the ACM international conference on image and video retrieval*, pages 1–9, 2009. 2

[3] Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. In *International Conference on Learning Representations*, 2017. 2, 5

[4] Dan Hendrycks, Steven Basart, Mantas Mazeika, Mohammadreza Mostajabi, Jacob Steinhardt, and Dawn Song. Scaling out-of-distribution detection for real-world settings. *arXiv preprint arXiv:1911.11132*, 2019. 2, 5

[5] Glenn Jocher. ultralytics/yolov5: v6.0 - yolov5n 'nano' models, roboflow integration, tensorflow export, opencv dnn support, 2021. 1

[6] Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. *Advances in neural information processing systems*, 31, 2018. 2, 5

[7] Shiyu Liang, Yixuan Li, and R Srikant. Enhancing the reliability of out-of-distribution image detection in neural networks. In *6th International Conference on Learning Representations, ICLR 2018*, 2018. 2, 5

[8] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017. 1

[9] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271, 2017. 1

[10] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018. 1

[11] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016. 1

[12] Shuai Shao, Zeming Li, Tianyuan Zhang, Chao Peng, Gang Yu, Xiangyu Zhang, Jing Li, and Jian Sun. Objects365: A large-scale, high-quality dataset for object detection. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 8430–8439, 2019. 2

[13] Haoran Wang, Weitang Liu, Alex Bocchieri, and Yixuan Li. Can multi-label classification networks know what they don't know? *Advances in Neural Information Processing Systems*, 34:29074–29087, 2021. 2, 5