OW-OVD: Unified Open World and Open Vocabulary Object Detection

Supplementary Material

A. Datasets

Tab. 5 presents the partition details of M-OWODB and S-OWODB. M-OWODB combines the COCO[33] and VOC[9] datasets and considers all VOC categories as Task 1, with subsequent tasks organized according to the superclasses listed at the top of Tab. 5. For example, in Task 3, the Sports and Food superclasses are added, with these categories annotated across 39,402 training images, containing 114,452 annotated instances. During testing, the instances of these categories are treated as known, reducing the number of test images and instances to 1,642 and 4,826, respectively. However, OW-DETR[16] notes that M-OWODB exhibits inconsistencies in superclass partitioning, leading to overlap of subclasses within the same superclass across tasks. The similarity among these subclasses introduces potential data challenges. To address this, S-OWODB was proposed. Unlike M-OWODB, S-OWODB includes only COCO dataset categories and organizes them strictly by superclass. This ensures that the detector does not encounter objects from the same superclass in subsequent tasks. For instance, in S-OWODB, the Furniture superclass is assigned to Task 2. Compared to M-OWODB, S-OWODB offers a more consistent partitioning, making it a more generalizable evaluation benchmark.

B. Hyperparameter analysis

When evaluating the impact of hyperparameters on model performance, we introduce additional configurations distinct from the incremental comparisons presented in Table 4. For instance, we include out-of-distribution probability and known uncertainty as part of the baseline settings. In addition, to select appropriate parameters, we hold all other adjustable parameters constant. When evaluating the effect of α, β is set to 0.5 and γ is fixed at 1. When adjusting the β parameter, we use the previously selected α and keep the γ parameter fixed at 1.

α, Eq. (3). Tab. 6 shows the impact of α on model performance. With task-aligned learning (TAL)[12], predictions with high Intersection over Union (IoU) relative to annotated data are assigned higher scores during matching. Thus, unlike traditional classification-oriented learning methods that assign binary scores of 0 (negative sample) or 1 (positive sample), TAL assigns scores on a continuous scale from 0 to 1. To classify samples assigned by TAL, we set a threshold, α, where samples with scores above α are considered positive, while those below α are considered negative. For threshold setting, we follow the COCO evaluation toolkit's verified threshold intervals, using values in the range of 0.5 to 0.95 with an incremental step of 0.05. The results are displayed in Tab. 6. Our model exhibits low sensitivity to threshold settings, likely due to the volume of annotations. Unlike in application scenarios, COCO has a larger volume of annotations, so higher thresholds do not reduce the availability of positive samples. In practical datasets, it may be advisable to lower the threshold. In the M-OWODB dataset, we select a threshold of 0.55, while for S-OWODB, we choose 0.75.

β, Eq. (6). Tab. 7 presents the impact of the parameter β on model performance. To prevent the selected attributes from being too similar, we incorporated a similarity constraint. In each iteration of selection, we combined the distribution similarity index with the average similarity of the current attributes and the selected attributes to filter out attributes similar to the ones already chosen. The balance between the combination of these factors is achieved using a hyperparameter, β. When the value of beta is set too high, the similarity constraint during selection becomes smaller, and vice versa. As shown in Tab. 7, when the attribute constraint is small, the selected attributes are too similar, resulting in significantly lower performance compared to other settings, both in the M-OWODB and S-OWODB benchmarks. As the value of beta is gradually increased, the performance of both benchmarks tends to stabilize. The U-mAP of M-OWODB stabilizes around 7.2, while that of S-OWODB stabilizes around 21. According to the data in the table, we selected a value of 0.2 for M-OWODB and 0.3 for S-OWODB.

γ, Eq. (7). Tab. 8 shows the performance variation of our OW-OVD with respect to the gamma parameter. Using a single attribute yields high recall rates, such as achieving 53.5 U-Recall in M-OWODB. However, using a single attribute increases uncertainty in object descriptions. This single-attribute approach results in the lowest cumulative detection performance (U-mAP), with values of 7.4 in M-OWODB and 21.7 in S-OWODB. By using the weighted sum of multiple attributes (i.e., P_b), the detector yields significant improvement, with U-mAP increasing from 7.4 to 8.6 in M-OWODB and from 21.7 to 23 in S-OWODB. However, the recall capability of the detector drops from 53.5 to 50 in M-OWODB and from 80.4 to 76.2 in S-OWODB. Given that mAP is the widely accepted metric for evaluating object detectors, our parameter selection prioritizes optimizing U-mAP. Therefore, we set the gamma value to 10 for both datasets.

Incremental learning. Tab. 9 illustrates the performance variations of parameter α in incremental learning. As tasks progress and the detector encounters an increasing

Table 5. Dataset Details. M-OWODB and S-OWODB represent the two divisions proposed by ORE[21] and OW-DETR[16], respectively.

M-OWODB	**Task 1**	**Task 2**	**Task 3**	**Task 4**
Semantic Split	VOC Classes	Outdoor, Accessories Appliances, Truck	Sports, Food	Electronic, Indoor, Kitchen, Furniture
# train images	16551	45520	39402	40260
# test images	4952	1914	1642	1738
# train instances	47223	113741	114452	138996
# test instances	14976	4966	4826	6039
S-OWODB	**Task 1**	**Task 2**	**Task 3**	**Task 4**
Semantic Split	Animals, Persion Vehicles	Appliances, Accessories Outdoor, Furniture	Sports, Food	Electronic, Indoor, Kitchen
# train images	89490	55870	39402	38903
# test images	3793	2351	1642	1691
# train instances	421243	163512	114452	160794
# test instances	17786	7159	4826	7010

Table 6. Impact of hyperparameters in Eq. (3) on performance (α).

Setting	**M-OWODB**			**S-OWODB**		
α	U-mAP	U-Recall	mAP	U-mAP	U-Recall	mAP
0.5	7.2	53	69.3	20.2	79.8	77.4
0.55	7.3	53.5	69.3	20.3	79.8	77.4
0.6	7.3	53.5	69.3	20.2	79.8	77.4
0.65	7.3	53.4	69.3	20.2	79.8	77.4
0.7	7.3	53.5	69.3	20.2	79.8	77.4
0.75	7.3	53.4	69.3	21.5	80.2	77.4
0.8	7.3	53.3	69.3	20.4	80.2	77.3
0.85	7.3	53.3	69.3	21.4	80.1	77.4
0.9	7.2	52.8	69.3	21.3	80.3	77.4
0.95	7.1	52.7	69.3	21.1	80.1	77.4

Table 7. Impact of hyperparameters in Eq. (6) on performance (β).

Setting	**M-OWODB**			**S-OWODB**		
β	U-mAP	U-Recall	mAP	U-mAP	U-Recall	mAP
0.1	5.9	52.6	69.2	17.4	79.4	77.2
0.2	7.4	53.5	69.3	20.6	80.4	77.4
0.3	7.3	53.5	69.3	21.7	80.4	77.4
0.4	7.3	53.4	69.3	21.5	80.2	77.4
0.5	7.3	53.5	69.3	21.4	80.2	77.4
0.6	7.3	53.4	69.3	20.2	79.8	77.4
0.7	7.2	53.0	69.3	21.6	80.1	77.4
0.8	7.1	53.3	69.3	21.6	80.1	77.4
0.9	7.1	53.3	69.3	21.5	80	77.4

Table 8. Impact of hyperparameters in Eq. (7) on performance (γ).

Setting	**M-OWODB**			**S-OWODB**		
γ	U-mAP	U-Recall	mAP	U-mAP	U-Recall	mAP
1	7.4	53.5	69.3	21.7	80.4	77.4
2	7.8	52.8	69.3	21.8	79.7	78.5
3	8.1	52.4	69.3	22.4	79.1	78.6
4	8.3	52.3	69.3	22.7	78.6	78.6
5	8.5	51.9	69.3	22.8	78.2	78.6
6	8.5	51.1	69.4	22.9	77.8	78.6
7	8.5	50.8	69.3	22.9	77.5	78.6
8	8.5	50.6	69.3	22.9	77.1	78.6
9	8.6	50.3	69.3	22.9	76.8	78.6
10	8.6	50	69.4	23	76.2	78.6

Table 9. Impact of hyperparameters on incremental learning (α).

Setting	**M-OWODB**			**S-OWODB**		
α (T2)	U-mAP	U-Recall	mAP	U-mAP	U-Recall	mAP
0.55	4.0	51.4	55.6	17.5	79.8	69.6
0.6	4.1	51.5	55.6	18.1	79.8	69.6
0.65	4.0	51.5	55.6	18.1	79.8	69.6
0.7	4.2	51.6	55.6	18.0	79.8	69.6
0.75	4.2	51.6	55.6	17.8	80.2	69.6
0.8	4.2	51.6	55.6	16.6	80.2	69.6
0.85	4.2	51.6	55.6	16.2	80.1	69.6
0.9	4.3	51.6	55.6	16.1	80.3	69.6
0.95	4.3	51.7	55.6	16.3	80.1	69.6

number of known object classes, the difficulty of attribute selection intensifies. This is because many attributes possess category-specific characteristics, making the selection process considerably more challenging. As shown in Tab. 9, the threshold α in Eq. (3), used to differentiate positive and negative samples, has a pronounced impact on model performance. In the S-OWODB benchmark, the detector's sensitivity to threshold α increases markedly in later tasks. The performance gap between different settings widens to 2.0 U-mAP, compared to only 1.3 in Task 1. Additionally, in the M-OWODB benchmark, a higher threshold is required. This is attributed to the detector's comparatively poorer performance in M-OWODB relative to S-OWODB, indicating that OW-OVD struggles more with distinguishing unknown classes and thus necessitates more precise attribute matching. Conversely, in the S-OWODB benchmark, the detector exhibits superior performance in handling unknown classes, allowing for a lower threshold setting. Therefore, we select 0.95 and 0.6 as the parameter settings for M-OWODB and S-OWODB, respectively.

C. Addition comparison

C.1. Implementation Details

M-OWODB and S-OWODB. GT denotes the use of unknown class names as prompts, representing the upper limit of the OVD model's capacity to detect unknown classes. FOMO/LLM employs a large language model (LLM) to predict potential unknown object class names. Given a list of known class names, the LLM is tasked with inferring the names of unknown object classes. FOMO/IN utilizes class names from the Imagenet[6] as unknown object class names. To prevent potential leakage, any class names that overlap with genuinely unknown objects are excluded. To ensure minimal modification, we evaluate using the class names provided by FOMO. FOMO training consists of three phases: attribute selection, attribute adaptation, and attribute refinement. During the attribute selection phase, a weighted sum of attribute similarities is used to predict known objects (as shown in Fig. 3, left). After training on known classes, linear layer weights are used to select attributes most similar to each known class. During the attribute adaptation phase, attribute embeddings are adjusted to reduce their distance to the mean embedding. In the attribute refinement phase, the focus is on training the distance between attribute embeddings and labels. Due to the use of batch normalization (BN) in YOLO-World's contrastive learning head, which differs from conventional configurations, the processing flows for visual and image regions are incompatible. Consequently, vision-guided finetuning is not applicable. In the attribute adaptation phase, instead of minimizing the distance between average similarities, we directly train adaptive embeddings.

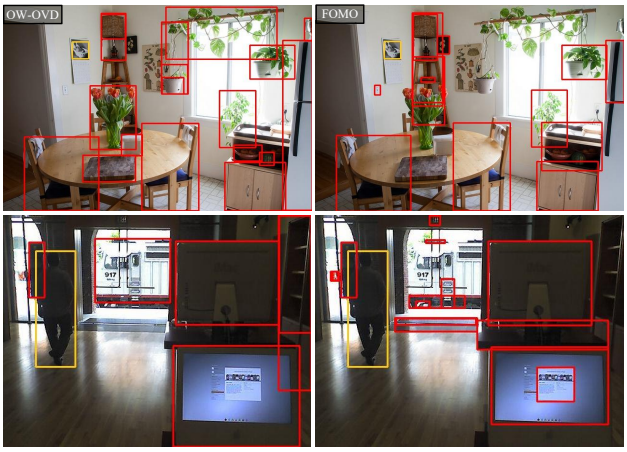

Figure 5. Visualization. The first column presents the performance of our OW-OVD, while the second column displays the results of FOMO. All images are sourced from the S-OWODB test set.

Incremental learning. Similar to the OWOD setting, we employ zero-shot testing to assess the generalization capability of the OVD detector under incremental learning tasks. The incremental learning training pipeline for FOMO differs from the OWOD benchmark, as we only employ the attribute refinement approach. Since the number of attributes generated in VOC is significantly lower than in M-OWODB and S-OWODB, we omit the original attribute selection and adaptation stages. In fact, applying the original three-stage process does not yield good performance because FOMO relies on attribute predictions for known classes. It is noteworthy that during the attribute refinement stage, we simultaneously train both the attribute embeddings and the linear weights, a modification that leads to better outcomes compared to the original setup. Additionally, because FOMO's original implementation does not support incremental learning, it continues to use annotations from the previous task in Task 2. Consequently, in our incremental learning experiments, we trained separate models for each task and then merged their linear weights and attribute embeddings to support incremental learning tasks.

C.2. Performance Comparison

U-mAP. We conducted a performance comparison between OW-OVD and the FOMO series, with detailed results presented in Tab. 10. The FOMO series includes two models: FOMO/LLM and FOMO/IN. FOMO/LLM relies on a large language model to predict potential object class names; however, due to the inherent limitations in the language model's prediction accuracy, the overall performance is suboptimal. In Task 1, regardless of whether the M-OWODB or S-OWODB dataset was used, FOMO/LLM did not exceed 1 U-mAP. FOMO/IN, on the other hand, employs ImageNet[6] class names as prompts for unknown

Table 10. Comparison of open-world object detection performance. GT represents zero-shot test results using all unknown categories as prompts. FOMO/LLM denotes the prediction of potential objects using a large language model. FOMO/IN indicates the prediction of unknowns based on ImageNet[6] categories. Our OW-OVD demonstrates a significant lead with remarkable performance.

Task IDs(→)	**Task 1**		**Task 2**				**Task 3**				**Task 4**		
Method	U-mAP (↑)	mAP (↑) Current Known	U-mAP (↑)	mAP(↑) Previously Known	mAP(↑) Current Known	mAP(↑) Both	U-mAP (↑)	mAP(↑) Previously Known	mAP(↑) Current Known	mAP(↑) Both	mAP(↑) Previously Known	mAP(↑) Current Known	mAP(↑) Both
Base+GT	23.8	69.0	20.0	69.0	40.5	54.8	18.0	54.8	29.8	46.5	46.5	24.1	40.9
FOMO/LLM	0.4	67.9	0.3	67.9	40.6	54.2	0.2	54.2	29.8	46.1	46.5	24.1	40.9
FOMO/IN	0.5	67.5	0.4	67.4	39.6	53.5	0.3	53.5	**29.9**	45.7	46.5	24.1	40.9
FOMO	2.6	68.4	1.6	67.4	40.4	53.9	1.3	53.5	26.8	44.6	46.5	24.1	40.9
Ours:OW-OVD	**8.6**	**69.4**	**4.3**	**69.5**	**41.7**	**55.6**	**4.0**	**55.5**	29.8	**47.0**	**47.0**	**25.2**	**41.6**
Base+GT	54.6	76.9	54.4	77.0	56.9	66.4	54.9	66.4	53.8	62.2	62.2	54.0	60.2
FOMO/LLM	0.2	76.9	0.4	76.9	56.9	66.4	0.3	66.4	53.8	62.2	62.2	54.0	60.2
FOMO/IN	0.3	76.9	0.4	76.9	56.8	66.4	0.3	66.3	53.7	62.2	62.2	54.0	60.2
FOMO	9.5	75.3	5.6	72.5	54.6	63.1	4.7	63.5	48.1	58.4	56.3	47.5	54.1
Ours:OW-OVD	**23.0**	**78.6**	**18.1**	**78.5**	**61.5**	**69.6**	**16.9**	**69.6**	**55.1**	**64.7**	**64.8**	**56.3**	**62.7**

Table 11. Comparison of incremental object detection performance. The table compares the performance of OW-OVD with FOMO in incremental learning on the PASCAL VOC dataset[9]. The table shows three category splits: 10+10, 15+5, and 19+1. The grey areas indicate the new categories introduced in the second task. mAP represents the mean average precision at the end of all training tasks.

10+10 setting	aero	cycle	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	bike	person	plant	sheep	sofa	train	tv	mAP
Base+GT	97.7	96.3	91.3	62.7	73.6	96.2	92.5	95.3	74.4	90.6	77.5	94.1	97.6	95.7	91.8	52.7	91.7	80.6	95.4	81.3	86.4
FOMO	52.2	91.9	67.7	44.8	51.3	73.0	77.2	91.6	62.1	67.8	85.5	90.6	72.9	91.2	73.6	45.4	65.2	85.2	87.4	77.1	72.7
Ours: OW-OVD	97.3	96.1	90.9	73.4	77.1	95.7	92.3	95.3	75.2	89.3	79.2	94.0	97.3	94.9	91.5	53.7	90.3	82.5	95.3	82.3	**87.2**
15+5 setting	aero	cycle	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	bike	person	plant	sheep	sofa	train	tv	mAP
Base+GT	97.7	96.3	91.3	62.7	73.6	96.2	92.5	95.3	74.4	90.6	77.5	94.1	97.6	95.7	91.8	52.7	91.7	80.6	95.4	81.3	86.4
FOMO	84.4	94.5	76.7	64.9	46.4	91.5	75.7	93.2	64.3	81.9	83.6	90.6	76.9	93.5	78.2	49.9	72.1	76.1	91.9	51.6	76.9
Ours: OW-OVD	97.1	96.0	90.6	74.5	76.8	95.7	92.4	95.0	74.7	87.6	79.6	94.4	97.4	94.4	91.5	54.9	89.0	81.4	95.3	79.3	**86.9**
19+1 setting	aero	cycle	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	bike	person	plant	sheep	sofa	train	tv	mAP
Base+GT	97.7	96.3	91.3	62.7	73.6	96.2	92.5	95.3	74.4	90.6	77.5	94.1	97.6	95.7	91.8	52.7	91.7	80.6	95.4	81.3	86.4
FOMO	82.8	94.3	79.7	47.3	55.2	92.6	77.0	93.7	54.0	74.6	85.1	92.5	93.1	92.8	77.2	47.7	76.0	88.2	91.9	56.1	77.6
Ours: OW-OVD	97.0	96.0	90.8	75.1	76.3	95.3	92.4	95.0	75.0	86.3	78.9	94.3	97.2	94.3	91.6	52.1	90.0	82.2	95.4	80.4	**86.8**

objects. Nevertheless, because it does not include any accurate unknown class names, the theoretical detection accuracy is expected to be near zero. In contrast, FOMO leverages attribute-based predictions for unknown objects, which provides a notable advantage over using exact class names. Attributes serve as more generalized representations, enhancing the OVD detector's ability to recognize objects in unseen scenarios. On the M-OWODB benchmark, FOMO achieved a U-mAP of 2.6, and on the S-OWODB benchmark, it reached 9.5 U-mAP, significantly outperforming both FOMO/LLM and FOMO/IN. Despite FOMO's improvements, our OW-OVD model consistently demonstrated a marked performance advantage across most metrics. In the M-OWODB benchmark, OW-OVD surpassed FOMO by 6 U-mAP (an increase of 230%), and in the S-OWODB benchmark, it outperformed FOMO by 13.5 U-mAP (a 142% increase). Furthermore, in subsequent tasks, OW-OVD continued to lead. In Task 2 and Task 3, OW-OVD achieved U-mAP scores of 18.1 and 16.9, respectively, compared to FOMO's scores of 5.6 and 4.7, resulting in a substantial advantage of 12 U-mAP and 12.2 U-mAP. Our OW-OVD model employs HAUF rather than a linear layer for unknown object prediction, thereby retaining the zero-shot detection capabilities of OVD. Conversely, FOMO relies on a linear layer for known class predictions, which causes a decline in performance as the number of known classes increases. This reliance can even lead to worse performance compared to zero-shot testing (60.2 vs. 54.7). In contrast, OW-OVD overcomes this limitation, maintaining robust performance (60.2 vs. 62.7).

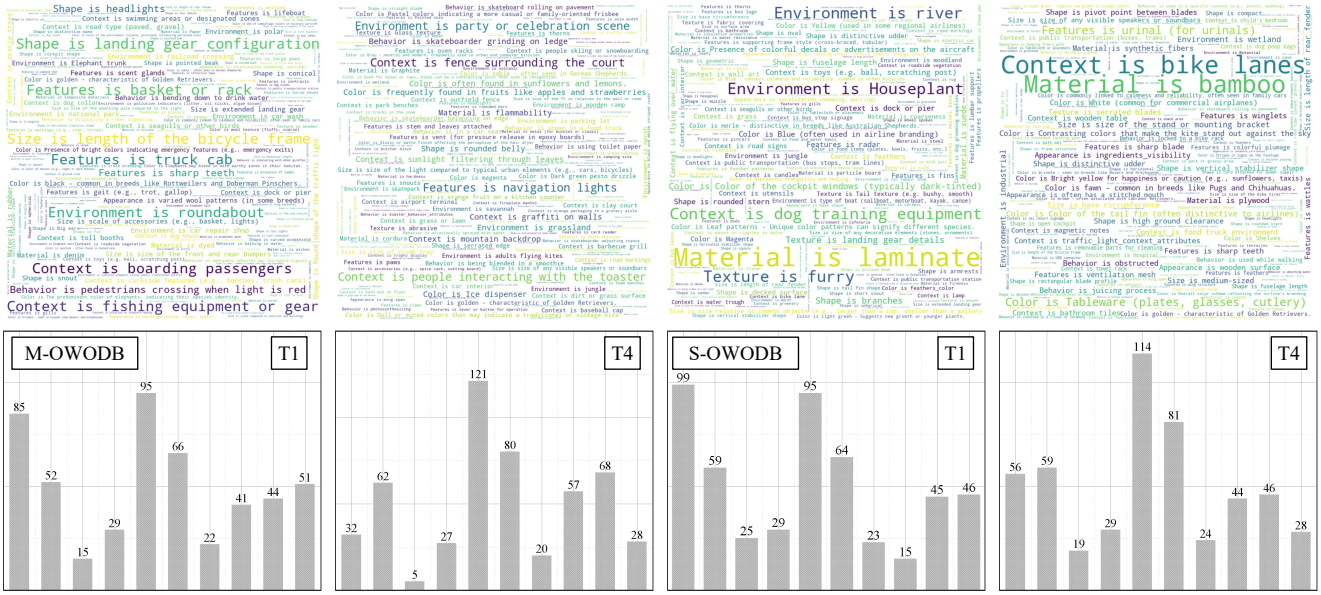

Figure 6. Attribute analysis. The word cloud at the top illustrates the selection order of attributes, where larger font sizes indicate higher selection priority. The bar chart at the bottom presents the distribution of the number of occurrences for each attribute category after selection. The horizontal axis of the bar chart represents the following categories: Shape, Color, Texture, Size, Context, Features, Appearance, Behavior, Environment, and Material. The first two columns on the left correspond to Tasks 1 and 4 of M-OWODB, while the two columns on the right pertain to S-OWODB. To ensure clarity, only the top 500 attributes are displayed in the figure.

incremental learning. Tab. 11 presents a comparative analysis of the performance of our proposed OW-OVD and FOMO in incremental learning. In zero-shot testing, the OVD detector exhibited remarkable generalization capabilities, achieving an mAP of 86.4. In contrast, FOMO, which utilized the same OVD detector, demonstrated inferior performance, recording an mAP of only 72.7 in the 10+10 setup, trailing behind the Base+GT by 13.7 mAP. However, in the 15+5 and 19+1 configurations, FOMO's performance improved, with the gap narrowing to 9.5 mAP and 8.8 mAP, respectively. Notably, FOMO displayed significant performance fluctuations across the three tasks, increasing from 72.7 mAP in the 10+10 setting to 77.6 mAP in the 19+1 setup. This pronounced variability indicates that FOMO's performance on known categories is influenced by the number of unknown categories present. Such limitations were also observed in earlier detectors. Our OW-OVD retains the performance advantages of OVD, consistently outperforming in all three task divisions. Furthermore, the performance of OW-OVD remains almost constant across these three task settings, with only a 0.4 mAP difference. This insensitivity to the number of newly introduced categories in subsequent tasks gives OW-OVD a distinct advantage in open-world scenarios.

Qualitative analysis. Fig. 5 presents a qualitative comparison between our proposed OW-OVD and FOMO on the S-OWODB benchmark. Our OW-OVD demonstrates superior recall capability for unknown objects. In the first row, OW-OVD successfully recalls the board and tissue on the desk, whereas FOMO fails to detect them. In the second row, OW-OVD successfully identifies the vehicle outside the door as an unknown object, while FOMO fails to detect it. Regarding detection accuracy, FOMO exhibits significant shortcomings. In the first row, similar to SAM-based methods, FOMO erroneously separates different parts of the ladder and generates redundant predictions for the desk lamp. In the second row, FOMO produces separate predictions for objects under the car and incorrectly predicts the pages on the computer screen. In contrast, our OW-OVD handles these situations correctly.

D. Attribute analysis

Fig. 6 illustrates the attribute variations of the detector following the application of the method for selecting features with the greatest similarity between positive and negative samples. In the M-OWODB benchmark, a notable change is observed in the increase in the number of shape and material categories. Specifically, the number of Shape attributes nearly doubled, rising from 56 to 99, while the Material attributes increased from 28 to 46. This trend indicates that as task complexity increases, the VSAS mechanism tends to prioritize more general and stable geometric and material features. This behavior also indicates that these features exhibit higher similarity between positive and

negative sample regions. Conversely, the number of Context and Behavior categories decreased significantly, especially for Behavior, which dropped from 44 to 15, representing a pronounced reduction. The count of Context attributes decreased from 114 to 95, implying that the model progressively relies more on context features when distinguishing between positive and negative samples, as these features exhibit greater differentiation. In Task 1 of S-OWODB, the numbers for Context and Behavior categories were relatively high, at 95 and 41, respectively. This observation suggests that these categories encapsulate common context and behavior features shared across positive and negative samples, leading the detector to retain them as generalized attributes since they are challenging to distinguish effectively. In Task 4, the Context category saw a substantial increase to 121, and the Behavior category rose to 57. This upward trend indicates that as task complexity intensifies, the detector recognizes more context and behavior features as exhibiting commonality between positive and negative samples, resulting in these attributes being prioritized during selection. This choice likely reflects the increased environmental complexity in new tasks, where contextual information becomes more similar between sample groups. As the number of known categories gradually expands, initial attributes are increasingly discarded during selection. For instance, in M-OWODB, the attribute, feature is truck cab, progressively diminishes and even vanishes by Task 4. Meanwhile, in S-OWODB, the attribute environment is river disappears in Task 4, whereas attribute, material is bamboo, becomes a primary selected attribute. These changes suggest that with the addition of known categories, attributes like shape gain greater importance in the detector's decision-making process.

E. Pseudocode

The presented pseudocode describes our method for selecting visual similarity attributes in a way that balances generalizability and redundancy. In Step 1, we generate descriptive attributes using a large language model, ensuring a rich set of features. Step 2 encodes these features and class names into a suitable representation for comparison. Steps 3 and 4 construct and normalize distributions of similarities between positive and negative samples, which are crucial for assessing the discriminative power of each attribute. Finally, Step 5 iteratively selects attributes by minimizing a combined measure of JSD and redundancy, ensuring that the selected features are both effective and diverse. This iterative selection process continues until our stopping criteria are satisfied, which ensures that the algorithm efficiently converges on a robust set of attributes. Consistent with prior work[63], we select 25 attributes for each category.

The presented pseudocode outlines our Hybrid Attribute-Uncertainty Fusion (HAUF) inference method, designed to improve the detection of unknown objects while preserving the model's capability for zero-shot detection. The algorithm operates in four key steps to achieve an effective balance between detecting unknown objects and accurately

Algorithm 1 Attribute Generation and Visual Similarity Attribute Selection (VSAS)

Require: CategoryNames, TrainingSet, LLM, TextEncoder, VisualEncoder, BoxHead, ContrastiveHead, JSD, β
Ensure: Selected Attributes $\hat{E}_{att}$

Step 1: Attribute Generation
$E_{att} \leftarrow \emptyset$
for each category name in CategoryNames **do**
  features $\leftarrow$ LLM.GenerateFeatures(category name)
  sentences $\leftarrow$ ConstructSentences(features)
  $E_{att} \leftarrow E_{att} \cup$ TextEncoder.Encode(sentences)
end for
Step 2: Class Embedding
$E_C \leftarrow$ TextEncoder.Encode(CategoryNames)
Step 3: Distribution Construction
$E^+_{vis} \leftarrow \emptyset$, $E^-_{vis} \leftarrow \emptyset$
for each image I in TrainingSet **do**
  $E_{vis} \leftarrow$ VisualEncoder.Encode(I)
  $P_{cls} \leftarrow$ BoxHead.Decode(E_{vis})
  $P_{box} \leftarrow$ ContrastiveHead.Decode(E_{vis})
  matchedScores $\leftarrow$ MatchScores(P_{cls}, P_{box}, GT)
  for each score in matchedScores **do**
    if score $\geq \alpha$ **then**
      $E^+_{vis} \leftarrow E^+_{vis} \cup E_{vis}[i]$
    else
      $E^-_{vis} \leftarrow E^-_{vis} \cup E_{vis}[i]$
    end if
  end for
end for
Step 4: Similarity Calculation
$D^+_i \leftarrow \emptyset$, $D^-_i \leftarrow \emptyset$
for each e_{att_i} in E_{att} **do**
  $d^+_i \leftarrow$ CalculateSimilarity(E^+_{vis}, e_{att_i})
  $d^-_i \leftarrow$ CalculateSimilarity(E^-_{vis}, e_{att_i})
  $D^+_i \leftarrow$ NormalizeDistribution(d^+_i)
  $D^-_i \leftarrow$ NormalizeDistribution(d^-_i)
end for
Step 5: Iterative Attribute Selection
$\hat{E}_{att} \leftarrow \emptyset$
repeat
  $i \leftarrow \arg\min \Big(\beta \cdot \text{JSD}(D^+_i, D^-_i) + (1-\beta) \cdot \text{RedundancyPenalty}(E_{att}[i], \hat{E}_{att}) \Big)$
  $\hat{E}_{att} \leftarrow \hat{E}_{att} \cup \{E_{att}[i]\}$
until SelectionCriteriaMet$(\hat{E}_{att})$
return $\hat{E}_{att}$

Algorithm 2 Hybrid Attribute-Uncertainty Fusion (HAUF) Inference

Require: e_{vis}: Visual embedding, h_{cls}: Class scoring function, $\hat{E}_{att}$: Attribute embeddings, P_C: Known class confidence scores, γ: Number of top attribute scores, k: Number of known classes

Ensure: P_u: Out-of-distribution probability

Step 1: Compute attribute-based scores
$s \leftarrow h_{cls}(e_{vis}, \hat{E}_{att})$
$s_\gamma \leftarrow$ Top-γ scores from s
Step 2: Calculate foreground probability P_b
$P_b \leftarrow \frac{1}{\gamma} \sum_\gamma \text{softmax}(s_\gamma) \cdot s_\gamma$
Step 3: Compute uncertainty probability P_{un}
$P_{un} \leftarrow \frac{1}{k} \sum_k \left(-p_k \log(p_k) - (1-p_k)\log(1-p_k)\right)$
Step 4: Compute unknown object probability P_u
$P_u \leftarrow \frac{1}{2}(P_b + P_{un}) \cdot (1 - \max(P_C))$
return P_u

classifying known categories. Step 1 computes attribute-based scores. The top γ attribute scores, s_γ, are then selected to focus on the most relevant features for distinguishing unknown objects. Step 2 calculates the foreground probability, P_b, which measures the likelihood of an object being distinguishable from the background. This probability is computed using a weighted average of the top γ attribute scores, scaled by their softmax values. Step 3 computes the uncertainty probability, P_{un}, which assesses the model's confusion regarding the current object's classification relative to known categories. Step 4 combines both probabilities, P_b and P_{un}, to compute the final unknown object probability, P_u. This probability is further modulated by $1 - \max(P_C)$, which reduces the likelihood of misclassifying a known object as unknown if the model has high confidence in a known class.

F. Incremental learning analysis

The qualitative results of our incremental learning analysis are illustrated in Fig. 7. The primary objective of incremental learning is to enable detectors to function effectively in open-world scenarios. Specifically, the goal of OWOD is to empower detectors to actively discover objects of interest during the inference stage and iteratively refine their detection capabilities through annotator-assisted incremental training. By continual fine-tuning, the detector incrementally acquires the ability to recognize new categories, meeting the practical demands of open-world applications.

In the unknown object discovery phase, both the recall rate and detection precision for unknown objects are critical metrics. These indicators directly influence the efficiency of annotators and significantly impact the outcomes of subsequent incremental learning stages. Compared to our OW-OVD, detection methods relying on Segment Anything Model (SAM) exhibit subpar performance in detecting unknown objects. For instance, the KTCN erroneously segments the hand and nose of a teddy bear as separate objects, while the SGROD method generates multiple redundant predictions for the ground. Such errors impose unnecessary burdens on annotators, reducing efficiency. In subsequent tasks, OW-OVD demonstrates superior accuracy in detecting known objects, outperforming KTCN and SGROD by a significant margin. Furthermore, OW-OVD shows notable advantages in the recall of unknown objects. For example, while MEPU fails to recall the plate on the table and SKDF misses the bottle, OW-OVD not only successfully recalls these objects but also correctly classifies them as known categories in the subsequent incremental learning stages. In contrast, MEPU and SKDF exhibit clear limitations in these aspects. Overall, OW-OVD outperforms other methods in both the precision of known object detection and the recall of unknown objects. This superiority not only reduces the workload of annotators but also significantly enhances the applicability of incremental learning in open-world scenarios, providing robust support for subsequent tasks.

Figure 7. Incremental learning analysis. This image presents a comprehensive evaluation of incremental learning performance, with visualizations systematically arranged from top to bottom for the methods KTCN, SGROD, MEPU, and SKDF. Each row contains four images: the first two display the outcomes from other SOTA methods, while the latter two represent our OW-OVD. Specifically, the first column illustrates the performance of these methods on the test set after the completion of training for Task 1. The second column, on the other hand, demonstrates the results after training has been completed for Task 4, reflecting their effectiveness at later stages of incremental learning. The third and fourth columns are dedicated to showcasing ours OW-OVD. These columns provide a direct comparison of our method's performance on the test set following the completion of training for Task 1 and Task 4, respectively. To ensure clarity and focus in the presentation, only predictions for unknown categories are visualized for Task 1, while for Task 4, we emphasize predictions for known categories, aligning with the progression of incremental learning tasks. Additionally, the images corresponding to the KTCN method are drawn from the M-OWODB test set, whereas the results for other methods are obtained from the S-OWODB test set.