Supplementary Material
D^{2}F2WOD: Learning Object Proposals for Weakly-Supervised Object Detection via Progressive Domain Adaptation

Yuting Wang
Rutgers University
Piscataway, NJ
yw632@cs.rutgers.edu

Ricardo Guerrero
Samsung AI Center
Cambridge, UK
r.guerrero@samsung.com

Vladimir Pavlovic
Rutgers University
Piscataway, NJ
vladimir@cs.rutgers.edu

1. Details of WSOD Models

Our object proposal generation strategy can be applied to different WSOD methods to boost their detection performance. In the main paper, we show two representative WSOD models – the widely-used OICR [19] and the state-of-the-art CASD [9]. Here, we highlight the main modeling elements of OICR and CASD. Additional details can be found in [19, 9].

As mentioned in Sec. 3.4 of the main paper, we obtain d-dimensional object proposal feature vectors $\mathbf{V}_t \in \mathbb{R}^{d\times M_t}$ for each input image $\mathbf{X}_t$, where M_t is the number of the proposal bounding boxes associated with $\mathbf{X}_t$. These object features are fed into the detection head of OICR [19] or CASD [9] to classify and localize objects. Both models contain a core multiple instance detection network (MIL) and P instance refinement classifiers.

1.1. Core MIL Head

As shown in Fig. 1, the core multiple instance detection network conducts the image-level multiple instance classification supervised by image-level labels $\mathbf{Y}_t^{(w)}$. In the core multiple instance detection network, the object proposal feature vectors $\mathbf{V}_t$ of image $\mathbf{X}_t$ are branched into two parallel classification and detection streams to generate two matrices $\mathbf{x}^{(\text{cls})}$ and $\mathbf{x}^{(\text{det})} \in \mathbb{R}^{C\times M_t}$ by two FC layers, where C is number of classes in $\mathcal{T}$. Then, $\mathbf{x}^{(\text{cls})}$ passes through a softmax layer along the category direction (column-wise), while $\mathbf{x}^{(\text{det})}$ passes through another softmax layer along the proposal direction (row-wise), leading to $\sigma(\mathbf{x}^{(\text{cls})})$ and $\sigma(\mathbf{x}^{(\text{det})})$, respectively. The instance-level classification score for the object proposals is computed as the element-wise product $\mathbf{x}^{(0)} = \sigma(\mathbf{x}^{(\text{cls})}) \odot \sigma(\mathbf{x}^{(\text{det})})$. Finally, the image-level classification score for class c is obtained as $p_c = \sum_{i=1}^{M_t} \mathbf{x}_{c,i}^{(0)}$. We train the core instance classifier using a multi-class cross-entropy loss $\mathcal{L}_{\text{mlc}}$. By using the instance-level classification scores $\mathbf{x}^{(0)}$, we select proposals as detected objects. However, the core MIL head focuses on most discriminative object instances.

1.2. OICR

To address this issue, OICR [19] introduces multi-stage instance refinement classifiers to refine the core instance classifier. As shown in Fig. 1, $\mathbf{V}_t$ is fed into P refinement instance classifiers. Each p-th refinement classifier comprises of an FC and a softmax layers along the category direction, and produces a proposal score matrix $\mathbf{x}^{(p)} \in \mathbb{R}^{(C+1)\times M_t}$, where the $(C+1)$-th category is the background class. We train the p-th refinement instance classifier via a log loss $\mathcal{L}_{\text{ref}}^p$ supervised by instance-level pseudo-labels, which are selected from the top-scoring proposals in the previous stage.

The loss for training the OICR network $\mathcal{L}_{\text{oicr}}$ is defined as

$$\mathcal{L}_{\text{oicr}} = \mathcal{L}_{\text{mlc}} + \lambda_d \sum_{p=1}^{P} \mathcal{L}_{\text{ref}}^p, \tag{1}$$

where λ_d is the trade-off hyperparameter.

1.3. CASD

To further improve OICR, CASD [9] employs an attention-based feature learning method for WSOD model training. In addition to the $\mathcal{L}_{\text{oicr}}$ loss, CASD includes a proposal bounding box smooth $\mathcal{L}_1$ regression loss $\mathcal{L}_{\text{reg}}^p$ for p-th refinement instance classifier by following [23, 14].

To encourage consistent representation learning of the same image under different transformations (horizontal flipping and scaling), we consider the Input-wise CASD following [9]. For each image $\mathbf{X}_t$, we construct a set of images $\mathbf{X}_t^{tr} = \{\mathbf{X}_t^{(s1)}, \mathbf{X}_t^{(\text{flip}(s1))}, \ldots, \mathbf{X}_t^{(sn)}, \mathbf{X}_t^{(\text{flip}(sn))}\}$, where $\mathbf{X}_t^{(si)}$ is its scaled image at si scale, $\mathbf{X}_t^{(\text{flip}(si))}$ is the horizontally flipped image of the scaled image, and n is the number of scales. Then, by feeding the set of images $\mathbf{X}_t^{tr}$ into the same feature extractor FE_5

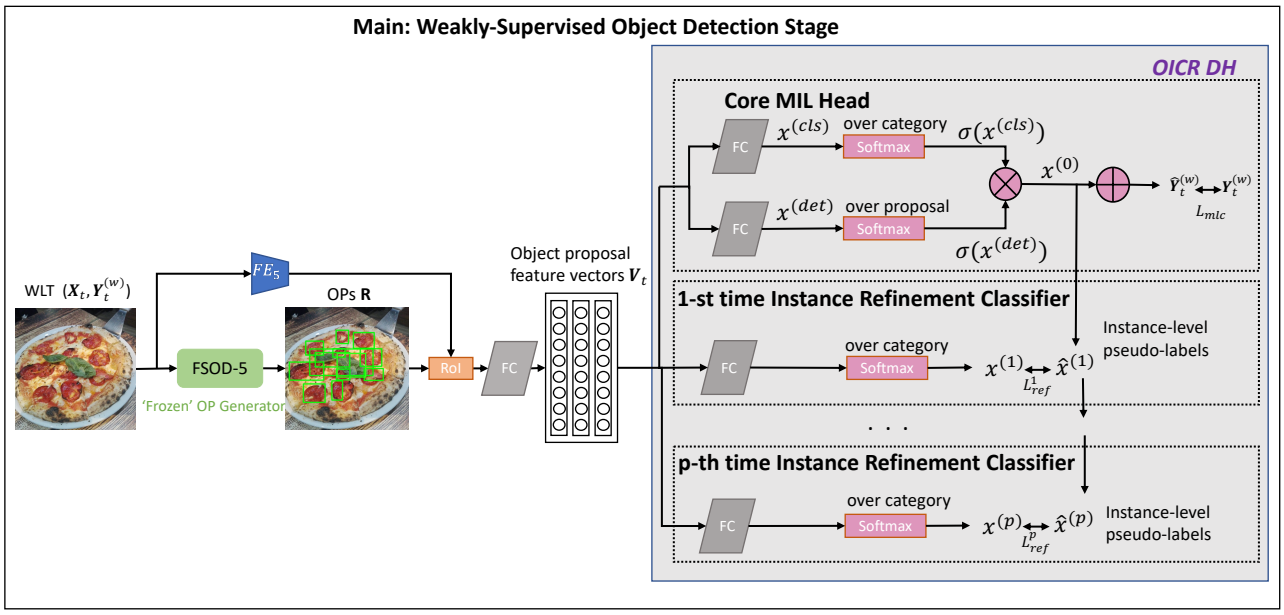


Figure 1: The architecture of our main weakly-supervised object detection stage.

in the WSOD model, we obtain a set of image feature maps $\mathbf{F}_t^{tr} = \{\mathbf{F}_t^{(s1)}, \mathbf{F}_t^{(\text{flip}(s1))}, \ldots, \mathbf{F}_t^{(sn)}, \mathbf{F}_t^{(\text{flip}(sn))}\}$. For each object proposal r, we compute the proposal feature vectors cropped from $\mathbf{F}_t^{tr}$, and use all proposal feature vectors to obtain a set of object proposal attention maps $\mathbf{A}_r^{tr} = \{\mathbf{A}_r^{(s1)}, \mathbf{A}_r^{(\text{flip}(s1))}, \ldots, \mathbf{A}_r^{(sn)}, \mathbf{A}_r^{(\text{flip}(sn))}\}$ by channel-wise average pooling and element-wise sigmoid function. We use the aggregated attention maps $\mathbf{A}_r^{IW} = max(\mathbf{A}_r^{tr})$, where $max(\cdot)$ is the element-wise max operator, to update the feature extractor FE_5 in the p-th refinement step. During training, we add the $\mathcal{L}_{\text{IW}}^p$ loss, which is an $\mathcal{L}_2$ loss between $\mathbf{A}_r^{IW}$ and each object proposal attention map in $\mathbf{A}_r^{tr}$.

To encourage balanced representation learning of the same image produced at different feature extractor layers, we consider the Layer-wise CASD following [9]. The feature extractor FE_5 consists of Q number of convolutional blocks $FE_5 = \{B_1, \ldots, B_Q\}$. The original image $\mathbf{X}_t$ is fed into each of them to output a set of feature map $\mathbf{F}_t^B = \{\mathbf{F}_t^{B_1}, \ldots, \mathbf{F}_t^{B_Q}\}$. For each object proposal r, we compute the proposal feature vector in each block and use all the feature vectors to obtain a set of object proposal attention maps $\mathbf{A}_r^{bl} = \{\mathbf{A}_r^{B_1}, \ldots, \mathbf{A}_r^{B_Q}\}$. Similarly to Input-wise CASD, we obtain the aggregated attention maps $\mathbf{A}_r^{LW} = max(\mathbf{A}_r^{bl})$. To update the feature extractor FE_5 in the p-th refinement step, we add the $\mathcal{L}_{\text{LW}}^p$ loss, which is an $\mathcal{L}_2$ loss between $\mathbf{A}_r^{LW}$ and each object proposal attention map in $\mathbf{A}_r^{bl}$.

The loss for training the CASD network $\mathcal{L}_{\text{casd}}$ is defined as

$$\mathcal{L}_{\text{casd}} = \mathcal{L}_{\text{mlc}} + \sum_{p=1}^{P} (\lambda_d \mathcal{L}_{\text{ref}}^p + \lambda_g \mathcal{L}_{\text{reg}}^p + \lambda_i \mathcal{L}_{\text{IW}}^p + \lambda_i \mathcal{L}_{\text{LW}}^p), \tag{2}$$

where λ_d, λ_g, and λ_i are the trade-off hyperparameters. For additional details please refer to [19, 9].

2. Benchmarks

We evaluate our method on five dual-domain image benchmark pairs: SyntheticPizza10 [12] → RealPizza10 [12], Clipart1K [10] → VOC2007 [4], Watercolor2K [10] → VOC2007-sub, Comic2K [10] → VOC2007-sub, and Clipart1K → MS-COCO-sub [11] datasets. Each synthetic pizza contains up to 10 toppings. The total 16,340 SyntheticPizza10 images are split into 14,802 training and 1,538 testing. RealPizza10 is split into 5,029 training and 552 testing images. Both Clipart1K and VOC2007 contain 20 object classes. Both Watercolor2K and Comic2K contain 6 classes: bike, bird, car, cat, dog, and person, the subset of classes in the VOC2007. The Watercolor2K and Comic2K domains are split into two subsets: 1,000 training and 1,000 testing images. The VOC2007-sub dataset includes 3,487 training and 3,457 testing images. The MS-COCO-sub domain includes 95,279 training images and 4,031 testing images.

Challenge of SyntheticPizza10 → RealPizza10 benchmark. One of our contributions is that we construct the dual-domain benchmark SyntheticPizza10 → RealPizza10 for WSOD. Compared with existing WSOD benchmarks

(VOC2007 and MS-COCO), RealPizza10 is more challenging for the following reasons. First, there are more objects per image. On RealPizza10 each image contains 20 objects on average, while on VOC2007 there are 2 objects per image on average. Second, there is strong variation in appearance. For example, the toppings in Pizza images exhibit complex changes in appearance. Third, there are layered object occlusions due to the topping objects. Such difficulty is further reflected in the low detection performance of baselines on RealPizza10 (e.g., Faster R-CNN achieves only 4.3% mAP on RealPizza10, while 22.8% mAP on VOC2007 and 13.9% mAP on MS-COCO-sub, as shown in Tables 1 and 2 in the main paper). We hope Pizza10 can serve as a new WSOD benchmark.

3. Additional Implementation Details

Implementation Details. All models and experiments were implemented in Pytorch. The VGG16 [17] and ResNet-50 [6] models pre-trained on ImageNet [16] were used as FSOD feature extractor and WSOD feature extractor. VGG16 was used as Faster R-CNN feature extractor and ResNet-50 was used as Sparse DETR feature extractor. Because VOC2007 and MS-COCO lack clean background, we run experiments on Clipart1K $\rightarrow$ VOC2007, Watercolor2K $\rightarrow$ VOC2007-sub, Comic2K $\rightarrow$ VOC2007-sub, and Clipart1K $\rightarrow$ MS-COCO-sub without the 3rd adaptation step FSOD-3 on augmented intermediate domain $\mathcal{G}_2$ in our warm-up stage. In the main stage, the maximum number of training iteration was set to be 100K for all target domains.

Copy-Paste Augmentation. As stated in Sec. 3.3 of the main paper, we employ an object-aware data augmentation method based on copy-paste [22] to map images $\mathbf{X}_{g_1}$ to $\mathbf{X}_{g_2}$. For each image $\mathbf{X}_{g_1}$, we randomly copy several foreground object instances from $\mathcal{G}_1$, with resizing and flipping transformations, and paste them onto the real-world target background images from $\mathcal{T}$ to generate $\mathbf{X}_{g_2}$. The flipping transformation includes horizontal and vertical flipping transformations. The resize ratio is a random value between 0.8 to 1.2. The PL step can be in principle performed for K times to generate instance-level pseudo-annotations. In our experiments, we found that running the PL step twice has achieved satisfactory performance. After the second PL step, we apply the copy-paste augmentation with resizing and flipping transformations for the pseudo-labeled target images. According to the statistic information of Clipart1k $\rightarrow$ VOC2007 reported in Table 1, for each class, there is a maximum of two objects in an image. Therefore, we copy each pseduo-labeled object and randomly paste it 0 or 1 time onto the original image. On SyntheticPizza10 $\rightarrow$ RealPizza10, according to the statistic information reported in Table 2, we copy each pseduo-labeled object and randomly paste it maximum 20 times onto the original image. All the pasted objects and original objects have no overlapping.

CycleGAN. We trained CycleGAN [24] with the learning rate of 1.0×10^{-5} for the first ten epochs and a linear decaying rate to zero over the next ten epochs following [10] to generate intermediate images. We followed the original paper [24] for remaining hyperparameters.

Faster R-CNN. We trained Faster R-CNN [13] on images of a single scale. The short edge of input images was rescaled to 600, and the longest image edge was capped to 1000. We employed a learning rate, which is the same as the final learning rate for the previous step, to progressively fine-tune Faster R-CNN on (1) a transfer-labeled intermediate domain $\mathcal{G}_1$, (2) augmented transfer-labeled intermediate domain $\mathcal{G}_2$ and then on (3) the pseudo-labeled target domain $\mathcal{T}$ and (4) augmented pseudo-labeled target domain $\mathcal{T}$.

Sparse DETR. The number of object queries is 300 and we only use 10% of the encoder tokens on all benchmarks. We followed the original paper [15] for the other hyperparameters. Each adaptation step was conducted with the learning rate equal to the final learning rate of the prior step.

OICR and CASD. We followed the original papers [19, 9] for the hyperparameters.

Object Proposals. The number of instances for each class in SyntheticPizza10 $\rightarrow$ RealPizza10 and Clipart1K $\rightarrow$ VOC2007 is unbalanced and the statistics information is reported in Table 1 and Table 2, respectively. The statistics information of object proposals mentioned in the main paper is shown in Table 3.

Cost of Training and Computing Resources. We train $\texttt{D}^2\texttt{F2WOD}$ based on Faster R-CNN on a Tesla K80 GPU. As shown in Table 4, we have the following observations. (1) The warm-up stage takes much less time than the main stage, indicating our domain adaptation to be lightweight. (2) Standard CycleGAN training brings in the most additional computation overhead.

4. Additional Main Results

In Sec. 4.1 of the main paper we show the main results based on mAP values. Here, we list the whole mAP with per class AP values. Table 5a and Table 5b summarize the detection results on Clipart1K $\rightarrow$ VOC2007 and SyntheticPizza10 $\rightarrow$ RealPizza10 based on Faster R-CNN FSOD backbone, respectively. $\texttt{D}^2\texttt{F2WOD}$ incorporated with OICR is denoted as $\texttt{D}^2\texttt{F2WOD}_{\text{oicr}}$, with CASD is

Table 1: Statistics of Clipart1k → VOC2007 (train+test): number of images (#img), number of instances (#ins), and relative size of human-labeled object instances on average (%size).

Name	Clipart			VOC			Name	Clipart			VOC		
	#img	#ins	%size	#img	#ins	%size		#img	#ins	%size	#img	#ins	%size
Aero	41	73	14.7	442	591	26.3	Table	106	115	19.7	390	421	33.3
Bike	27	36	20.1	482	690	22.0	Dog	51	54	9.2	839	999	34.1
Bird	135	265	9.4	612	945	20.2	Horse	46	79	17.4	561	710	30.2
Boat	88	129	12.1	353	553	17.2	Mbike	16	17	51.2	467	664	28.3
Bottle	60	121	3.5	456	974	5.8	Person	521	1185	14.5	4015	9218	16.5
Bus	20	21	31.7	360	442	28.9	Plant	100	178	5.6	469	994	11.6
Car	103	202	11.0	1434	2451	19.8	Sheep	27	76	8.1	193	499	13.0
Cat	43	50	8.4	659	734	41.9	Sofa	42	52	19.1	452	487	35.6
Chair	181	340	8.6	862	1554	12.2	Train	45	46	40.3	520	579	36.9
Cow	30	46	19.1	268	503	18.4	Tv	65	80	8.4	485	632	12.8
Total								1000	3165	12.9	9963	24640	19.9

Table 2: Statistics of SyntheticPizza10 (train+test) → RealPizza10 (only test set is annotated, so here we report test statistic): number of images (#img), number of instances (#ins), and relative size of human-labeled object instances on average (%size).

Dataset		Pepperoni	Mushroom	Pepper	Olive	Basil	Bacon	Broccoli	Pineapple	Tomato	Onion	Total
Synthetic	#img	3741	3894	3901	3729	3868	3725	3904	3638	3979	3931	16340
	#ins	29113	31723	29336	29001	36846	36113	31853	28675	31374	29561	313595
	%size	2.4	2.2	2.4	2.4	1.4	1.3	2.2	2.3	2.3	2.7	2.1
Real	#img	197	96	94	70	152	24	10	8	161	76	552
	#ins	3638	1455	1239	841	1145	289	110	152	1697	706	11272
	%size	1.3	1.0	1.0	0.6	1.4	1.8	1.5	0.9	1.3	1.2	1.2

Table 3: Statistical information of the number of object proposals per image generated by the Faster R-CNN backbone in source domains.

Datasets	Min	Max	Mean	Std
PASCAL VOC 2007 train	300.0	1800.0	437.3	195.5
PASCAL VOC 2007 test	300.0	1500.0	424.8	185.0
RealPizza10 2007 train	300.0	1800.0	441.0	225.7
RealPizza10 2007 test	300.0	1500.0	482.6	236.7

Table 4: Training time for different stages on Clipart1K → VOC2007 (Clip → VOC), SyntheticPizza10 → RealPizza10 (SPizza → RPizza), Watercolor2K → VOC2007-sub (Water → VocS), and Comic2K → VOC2007-sub (Comi → VocS) based on Faster R-CNN FSOD backbone. '-' denotes that we run experiments without the 3rd adaptation step FSOD-3, since VOC2007 and MS-COCO lack clean background.

Training time	Dataset	Warm-Up Stage					Main Stage	CycleGAN
Stage		FSOD-1	FSOD-2	FSOD-3	FSOD-4	FSOD-5	CASD	
hour	SPizza → RPizza	14	10	13	2	2	79	216
	Clip → Voc	7	6	-	3	3	102	144
	Water → VocS	8	8	-	2	2	81	144
	Comi → VocS	4	4	-	8	8	81	144

denoted as $\texttt{D}^2\texttt{F2WOD}_{\text{casd}}$, and with CASD+W2N is denoted as $\texttt{D}^2\texttt{F2WOD}_{\text{casd+w2n}}$. The results of our warm-up stage are denoted as $\texttt{D}^2\texttt{F2WOD}_{\text{warm-up}}$.

The detection performance does not benefit from using OICR or CASD twice (once for proposal and once for object detection), since doing so does not improve the generated proposals. The result of training CASD two times is denoted as CASD^2. As shown in Table 5a, on Clipart1K → VOC2007, $\texttt{D}^2\texttt{F2WOD}_{\text{casd}}$ reaches 64.8% mAP, outperforming the original CASD by 7.8% mAP, while CASD^2 reaches 57.4% mAP, outperforming the original CASD by 0.4% mAP. The detection result of using OICR twice will be the same as using OICR once, because OICR directly uses the proposals produced by selective search without refinement on their bounding boxes. $\texttt{D}^2\texttt{F2WOD}_{\text{casd+w2n}}$ reaches 66.9% mAP, outperforming the original CASD+W2N by

1.5% mAP. This further validates that *D²F2WOD is a general framework that can be combined with different WSOD methods* to improve their object proposal generation and thus overall performance.

5. Additional Ablation Study

In Sec. 4.2 of the main paper we show the ablation study results based on the mAP values. Here, we list the whole mAP values with per class AP values.

- **Effectiveness of Progressive Adaptation.** As shown in Table 6, each adaptation step in our warm-up stage is not only helpful in terms of mAP, but it also benefits for each class.
- **Impact of Adaptation Order.** Table 7 shows that our *progressive* adaptation order that gradually reduces domain gap achieves the best performance – it is better to first fine-tune the FSOD on intermediate images or pseudo-labeled images, and then fine-tune on the augmented images.
- **Generalizability of the Warm-up Stage across FSODs.** As shown in Table 8, our $\texttt{D}^2\texttt{F2WOD}_{\text{warm-up}}$ and $\texttt{D}^2\texttt{F2WOD}_{\text{casd}}$ based on Sparse DETR yield 0.6% and 1.1% improvement in terms of mAP, respectively, compared with Faster R-CNN backbone.
- **Main Stage Configurations.** As shown in Table 9, two key components of our method are both effective and complementary to each other. Note that the current transformer-based detectors rely on relatively large amounts of annotate data; therefore, we found that it was difficult to train Sparse DETR on Clipart1K with only 500 training images; by contrast, Sparse DETR worked reasonably well on SyntheticPizza10 with 14,802 training images. Accordingly, for the experiment on the Clipart1K → VOC2007 datasets, we mainly focus on Faster R-CNN. We leave the investigation of Sparse DETR on Clipart1K → VOC2007 as interesting future work, by either exploring additional synthetic data to increase the synthetic training dataset size or leveraging more data-efficient transformer-based detectors.
- **Identifying Object Detection Errors.** We use TIDE [3] to analyse the classification, localization, both Cls and Loc, duplicate detection, background, and missed GT errors in DT+PL, CASD, and our model. Each chart shows the relative percentage of each type of error. As shown in Fig. 2, $\texttt{D}^2\texttt{F2WOD}$ effectively reduces the localization error compared with other two baselines. Here classification error indicates object localized correctly but misclassified; localization error indicates object classified correctly but mislocalized; both Cls and Loc error indicates object misclassified and mislocalized; duplicate detection error indicates object matched with a GT which has already matched with another higher confidence scoring prediction; background error indicates background detected as foreground; missed GT error indicates ground-truth that not matched with any predictions.

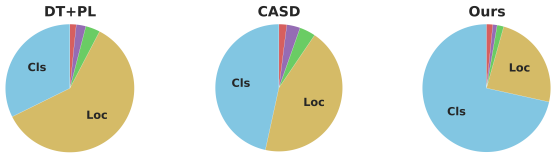


Figure 2: Summary of errors on DT+PL, CASD and our method.

6. Additional Qualitative Analysis

Fig. 3 shows the representative images generated by CycleGAN on different benchmarks.

Fig. 4 illustrates some examples used for FSOD-3 training on SyntheticPizza10 → RealPizza10.

Fig. 5 and Fig. 6 illustrate detection results produced by our $\texttt{D}^2\texttt{F2WOD}$ and CASD on RealPizza10 and VOC2007 datasets, respectively. There, it can be observed that $\texttt{D}^2\texttt{F2WOD}$ does not only locate most objects, but that it also produces more accurate bounding boxes. Specifically, in the RealPizza10 images it can be appreciated bounding boxes provided by our method (left) closely align with the objects of interest, while for CASD (right) bounding boxes are often imprecise (either wrong shape or big/small). Similar observations can be made for VOC2007 where CASD often fails to locate objects or produces spurious bounding boxes.

Fig. 7 illustrates some challenging cases on the RealPizza10 dataset where the performance of both our $\texttt{D}^2\texttt{F2WOD}$ and the baseline CASD still lags. We hypothesize this is because in these cases, the corresponding categories (e.g., bacon, broccoli and pineapple) have significantly smaller number of training examples.

Table 5: Results (AP in %) for different methods on Clipart1K → VOC2007 and SyntheticPizza10 → RealPizza10. We denote as Upper-Bound the FSOD (Faster R-CNN or Sparse DETR) results, trained and tested on *fully-annotated* target domain to indicate the weak upper-bound performance of our methods. Our warm-up stage is compared with CD models and our main stage is compared with SD models. The upper part shows the results using CD models. The lower part shows the results using SD methods. Faster R-CNN in CD means we trained our network on fully-annotated source and test on fully-annotated target domains. The best and second best results for D^{2}F2WOD compared with baselines are shown in red and blue.

(a) Clipart1K → VOC2007.

Type	Method	mAP	Aero	Bike	Bird	Boat	Bottle	Bus	Car	Cat	Chair	Cow	Table	Dog	Horse	Mbike	Person	Plant	Sheep	Sofa	Train	Tv
	Upper-Bound [13]	*69.9*	*69.8*	*79.1*	*67.5*	*56.5*	*54.4*	*77.2*	*82.2*	*80.9*	*50.1*	*78.4*	*64.5*	*78.4*	*83.7*	*72.3*	*77.2*	*38.3*	*70.9*	*66.6*	*77.5*	*71.9*
CD	Faster R-CNN [13]	22.8	10.7	39.7	30.5	8.6	19.3	27.4	48.0	4.5	23.7	21.2	7.9	19.0	21.9	21.5	45.0	17.4	16.1	22.5	25.5	25.0
	DT+PL [10]	34.6	18.8	55.8	33.2	20.4	18.8	47.2	56.2	15.8	27.4	45.5	10.7	25.9	54.1	54.3	47.6	10.6	35.4	42.3	47.0	25.9
	PADOD [7]	24.2	13.3	40.3	28.8	12.6	20.2	32.2	46.7	7.5	25.9	24.0	13.8	19.3	21.2	17.5	44.0	17.6	17.2	24.4	28.5	29.4
Ours	D^{2}F2WOD$_{\text{warm-up}}$	37.3	20.7	60.8	37.2	19.4	25.0	51.1	59.7	17.2	30.4	44.4	17.3	27.4	55.8	56.8	47.6	12.9	38.4	45.5	50.3	27.6
SD	WSDDN [2]	34.8	39.4	50.1	31.5	16.3	12.6	64.5	42.8	42.6	10.1	35.7	24.9	38.2	34.4	55.6	9.4	14.7	30.2	40.7	54.7	46.9
	OICR [19]	41.2	58.0	62.4	31.1	19.4	13.0	65.1	62.2	28.4	24.8	44.7	30.6	25.3	37.8	65.5	15.7	24.1	41.7	46.9	64.3	62.6
	PCL [18]	43.5	54.4	69.0	39.3	19.2	15.7	62.9	64.4	30.0	25.1	52.5	44.4	19.6	39.3	67.7	17.8	22.9	46.6	57.5	58.6	63.0
	WeakRPN [20]	45.3	57.9	70.5	37.8	5.7	21.0	66.1	69.2	59.4	3.4	57.1	57.3	35.2	64.2	68.6	32.8	28.6	50.8	49.5	41.1	30.0
	C-MIL [21]	50.5	62.5	58.4	49.5	32.1	19.8	70.5	66.1	63.4	20.0	60.5	52.9	53.5	57.4	68.9	8.4	24.6	51.8	58.7	66.7	63.5
	WSOD2(+Reg) [23]	53.6	65.1	64.8	57.2	39.2	24.3	69.8	66.2	61.0	29.8	64.6	42.5	60.1	71.2	70.7	21.9	28.1	58.6	59.7	52.2	64.8
	Pred Net [1]	52.9	66.7	69.5	52.8	31.4	24.7	74.5	74.1	67.3	14.6	53.0	46.1	52.9	69.9	70.8	18.5	28.4	54.6	60.7	67.1	60.4
	C-MIDN [5]	52.6	53.3	71.5	49.8	26.1	20.3	70.3	69.9	68.3	28.7	65.3	45.1	64.6	58.0	71.2	20.0	27.5	54.9	54.9	69.4	63.5
	MIST(+Reg) [14]	54.9	68.8	77.7	57.0	27.7	28.9	69.1	74.5	67.0	32.1	73.2	48.1	45.2	54.4	73.7	35.0	29.3	64.1	53.8	65.3	65.2
	CASD [9]	57.0	67.2	71.5	57.8	41.5	23.4	72.9	70.3	75.5	21.5	64.8	53.8	71.8	65.0	72.5	32.6	25.0	56.6	58.5	69.5	68.2
	CASD^2	57.4	56.1	64.6	61.4	60.9	35.1	59.9	76.9	56.6	27.8	73.6	51.2	60.1	70.5	72.5	34.4	54.0	70.5	45.1	53.1	63.7
	CASD+W2N [8]	65.4	74.0	81.7	71.2	48.9	51.0	78.6	82.3	83.5	29.1	76.9	51.5	82.1	76.9	79.1	28.5	34.3	65.0	64.2	75.2	74.8
Ours	D^{2}F2WOD$_{\text{casd}}$	64.8	62.7	64.9	69.9	47.9	57.9	74.3	85.7	59.6	43.4	82.2	39.6	67.2	84.0	77.8	74.0	50.6	74.6	48.8	66.7	64.6
	D^{2}F2WOD$_{\text{casd+w2n}}$	66.9	58.6	69.1	77.9	49.3	78.1	73.2	89.0	64.9	39.6	83.5	33.0	77.7	95.2	77.0	75.9	50.7	74.4	44.8	66.6	61.0

(b) SyntheticPizza10 → RealPizza10.

Type	Method	mAP	Pepperoni	Mushroom	Pepper	Olive	Basil	Bacon	Broccoli	Pineapple	Tomato	Onion
CD	Faster R-CNN [13]	4.3	12.1	0.4	9.6	5.0	3.4	0.3	1.0	1.0	9.7	0.9
	DT+PL [10]	14.9	30.7	4.3	11.6	25.3	42.7	1.3	3.6	2.4	21.4	5.2
	PADOD [7]	8.1	19.5	0.2	3.4	11.8	30.2	0.2	1.1	0.5	13.3	0.8
Ours	D^{2}F2WOD$_{\text{warm-up}}$	17.9	31.0	8.3	11.8	28.1	45.5	0.8	9.8	9.8	21.5	12.7
SD	OICR [19]	4.7	0.2	1.3	4.5	0.1	0	8.8	19.4	11.0	1.0	0.8
	CASD [9]	12.9	12.7	19.5	14.8	10.5	13.7	10.4	10.1	14.5	11.7	10.7
Ours	D^{2}F2WOD$_{\text{casd}}$	25.1	43.9	35.1	14.9	27.3	41.8	9.2	12.5	8.5	28.4	29.2

Table 6: Effectiveness of progressive adaptation: each adaptation step in our warm-up stage is helpful not only in terms of mAP, but also benefits for each class.

FSOD	Step	mAP	Pepperoni	Mushroom	Pepper	Olive	Basil	Bacon	Broccoli	Pineapple	Tomato	Onion
Faster R-CNN	FSOD-1	4.3	12.1	0.4	9.6	5.0	3.4	0.3	1.0	1.0	9.7	0.9
	FSOD-2	9.7	22.3	0.8	9.6	14.5	33.2	0.8	0.7	0.8	13.4	1.0
	FSOD-3	10.3	23.0	1.0	10.1	15.4	35.7	0.8	0.8	0.6	14.4	1.1
	FSOD-4	15.0	28.7	4.6	11.0	22.2	39.9	9.3	1.8	9.6	16.6	6.3
	FSOD-5	17.9	31.0	8.3	11.8	28.1	45.5	0.8	9.8	9.8	21.5	12.7
Sparse DETR	FSOD-1	3.7	10.0	0.2	0.8	20.2	0.7	0.2	0.4	0.6	1.0	2.6
	FSOD-2	9	27.0	1.7	0.6	24.5	19.1	0.1	2.3	2.5	8.3	4.0
	FSOD-3	10.3	30.40	0.9	1.2	28.1	27.7	0.0	1.2	1.3	9.2	2.5
	FSOD-4	18.2	47.7	4.7	5.9	39.8	46.0	0.0	1.7	3.1	25.1	7.8
	FSOD-5	18.5	48.3	6.0	5.3	43.0	43.1	0.1	1.4	2.4	26.1	9.2

Table 7: Impact of adaptation order (Faster R-CNN backbone).

Adaptation Domain	mAP	Pepperoni	Mushroom	Pepper	Olive	Basil	Bacon	Broccoli	Pineapple	Tomato	Onion
$\mathcal{S}$	4.3	12.1	0.4	9.6	5.0	3.4	0.3	1.0	1.0	9.7	0.9
$\mathcal{S} \rightarrow \mathcal{G}_1 \rightarrow \mathcal{G}_2$	10.3	23.0	1.0	10.1	15.4	35.7	0.8	0.8	0.6	14.4	1.1
$\mathcal{S} \rightarrow \mathcal{G}_2 \rightarrow \mathcal{G}_1$	5.9	5.8	1.1	9.7	6.9	20.4	0	2.3	0.5	9.1	3.2
$\mathcal{G}_2 \rightarrow \mathcal{T} \rightarrow$ Aug.$\mathcal{T}$	17.9	31.0	8.3	11.8	28.1	45.5	0.8	9.8	9.8	21.5	12.7
$\mathcal{G}_2 \rightarrow$ Aug.$\mathcal{T} \rightarrow \mathcal{T}$	17.3	32.6	6.3	8.3	28.4	47.3	0.6	4.9	10.4	22.4	11.3

Table 8: Generalizability of the warm-up stage across FSODs.

FSOD	Stage	mAP	Pepperoni	Mushroom	Pepper	Olive	Basil	Bacon	Broccoli	Pineapple	Tomato	Onion
Faster R-CNN	$\texttt{D}^2\texttt{F2WOD}_{\text{warm-up}}$	17.9	31.0	8.3	11.8	28.1	45.5	0.8	9.8	9.8	21.5	12.7
	$\texttt{D}^2\texttt{F2WOD}_{\text{casd}}$	25.1	43.9	35.1	14.9	27.3	41.8	9.2	12.5	8.5	28.4	29.2
Sparse DETR	$\texttt{D}^2\texttt{F2WOD}_{\text{warm-up}}$	18.5	48.3	6.0	5.3	43.0	43.1	0.1	1.4	2.4	26.1	9.2
	$\texttt{D}^2\texttt{F2WOD}_{\text{casd}}$	26.2	51.9	35.5	18.9	33.1	47.4	11.2	9.6	5.4	29.3	20.1

Table 9: Ablation study of $\texttt{D}^2\texttt{F2WOD}$ main configurations on (a) Clipart1K → VOC2007 (Faster R-CNN backbone), (b) SyntheticPizza10 → RealPizza10 (Faster R-CNN backbone) and (c) SyntheticPizza10 → RealPizza10 (Sparse DETR backbone). "**FE**" and "**OP**" denote the domain specific pre-trained feature extractor and weakly-supervised object proposal generator, respectively.

(a) Clipart1K → VOC2007 (Faster R-CNN backbone).

Type	Method	mAP	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
SD	OICR	41.2	58.0	62.4	31.1	19.4	13.0	65.1	62.2	28.4	24.8	44.7	30.6	25.3	37.8	65.5	15.7	24.1	41.7	46.9	64.3	62.6
$\texttt{D}^2\texttt{F2WOD}_{\text{oicr}}$	**+FE**	44.7	53.8	52.5	41.1	37.4	27.8	53.9	63.5	39.1	30.5	59.5	40.7	42.6	47.6	52.1	23.5	36.1	55.9	40.0	45.1	50.7
	+OP	47.2	23.4	54.4	46.9	34.6	46.5	69.4	78.0	10.1	44.7	65.6	27.7	25.9	52.8	64.3	65.3	32.5	54.7	42.1	52.8	52.3
	+FE+OP	52.7	39.1	60.6	56.2	37.4	48.0	67.8	81.0	18.6	51.8	67.5	38.1	31.3	72.0	67.8	70.2	40.0	60.6	40.9	56.4	49.3
SD	CASD	57.0	67.2	71.5	57.8	41.5	23.4	72.9	70.3	75.5	21.5	64.8	53.8	71.8	65.0	72.5	32.6	25.0	56.6	58.5	69.5	68.2
$\texttt{D}^2\texttt{F2WOD}_{\text{casd}}$	**+FE**	60.0	51.0	71.1	72.1	38.1	27.5	76.1	71.6	74.0	27.2	64.3	58.8	81.6	88.1	71.0	63.1	19.5	53.2	58.7	69.4	64.0
	+OP	60.1	38.3	67.6	63.2	45.4	62.0	77.7	88.9	24.1	56.3	76.9	44.9	41.4	76.8	77.4	75.1	42.2	68.6	52.4	62.2	61.4
	+FE+OP	64.8	62.7	64.9	69.9	47.9	57.9	74.3	85.7	59.6	43.4	82.2	39.6	67.2	84.0	77.8	74.0	50.6	74.6	48.8	66.7	64.6

(b) SyntheticPizza10 → RealPizza10 (Faster R-CNN backbone).

Type	Method	mAP	Pepperoni	Mushroom	Pepper	Olive	Basil	Bacon	Broccoli	Pineapple	Tomatoes	Onion
SD	OICR	4.7	0.2	1.3	4.5	0.1	0	8.8	19.4	11.0	1.0	0.8
$\texttt{D}^2\texttt{F2WOD}_{\text{oicr}}$	**+FE**	8.5	4.4	12.5	12.2	7.2	6.1	7.4	8.6	13.2	5.1	7.9
	+OP	12.6	23.0	18.5	8.5	14.7	20.8	5.0	2.3	3.9	13.9	15.8
	+FE+OP	13.8	24.3	19.7	10.0	15.2	21.9	3.7	7.5	3.6	16.3	15.4
SD	CASD	12.9	12.7	19.5	14.8	10.5	13.7	10.4	10.1	14.5	11.7	10.7
$\texttt{D}^2\texttt{F2WOD}_{\text{casd}}$	**+FE**	14.8	10.2	11.3	14.6	10.1	10.0	19.0	30.9	21.0	10.9	10.5
	+OP	24.0	45.8	36.7	14.8	25.8	37.4	3.2	12.3	3.5	32.1	28.3
	+FE+OP	25.1	43.9	35.1	15.0	27.3	41.8	9.2	12.5	8.5	28.4	29.2

(c) SyntheticPizza10 → RealPizza10 (Sparse DETR backbone).

Type	Method	mAP	Pepperoni	Mushroom	Pepper	Olive	Basil	Bacon	Broccoli	Pineapple	Tomatoes	Onion
SD	OICR	5.9	12.1	9.9	2.8	5.4	12.7	1.0	0.3	0.1	5.5	9.4
$\texttt{D}^2\texttt{F2WOD}_{\text{oicr}}$	**+FE**	10.8	21.7	17.1	9.3	11.6	15.2	0.4	4.5	0.8	15.6	12.1
	+OP	13.4	22.1	16.8	9.0	13.5	23.7	10.9	5.5	2.2	15.6	14.3
	+FE+OP	15.4	28.0	13.2	9.5	21.5	24.1	8.0	14.0	7.7	19.6	8.3
SD	CASD	13.4	25.2	17.0	8.3	14.9	21.0	10.0	5.9	0.5	18.0	12.8
$\texttt{D}^2\texttt{F2WOD}_{\text{casd}}$	**+FE**	15.8	26.7	22.1	14.3	16.6	20.2	5.4	9.5	5.8	20.6	17.1
	+OP	25.1	44.5	35.0	14.6	28.7	46.2	8.6	8.5	3.8	31.7	29.1
	+FE+OP	26.2	51.9	35.5	18.9	33.1	47.4	11.2	9.6	5.4	29.3	20.1

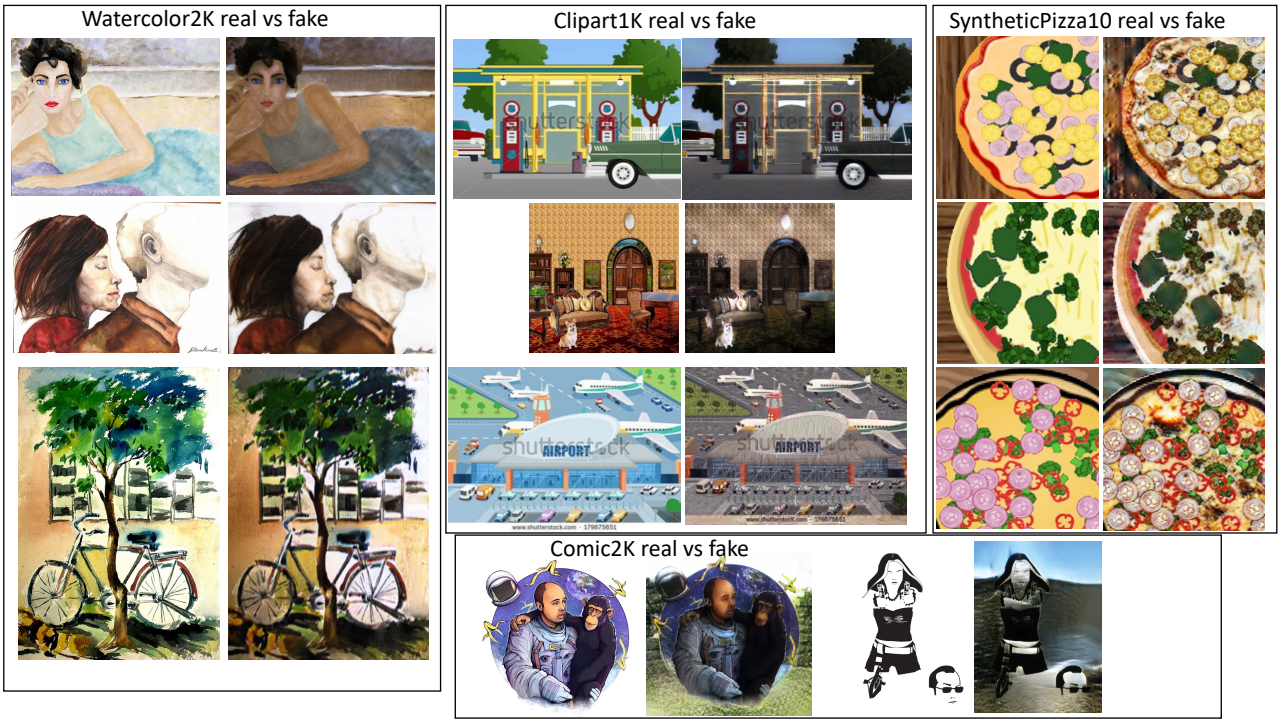


Figure 3: Representative images generated by CycleGAN.

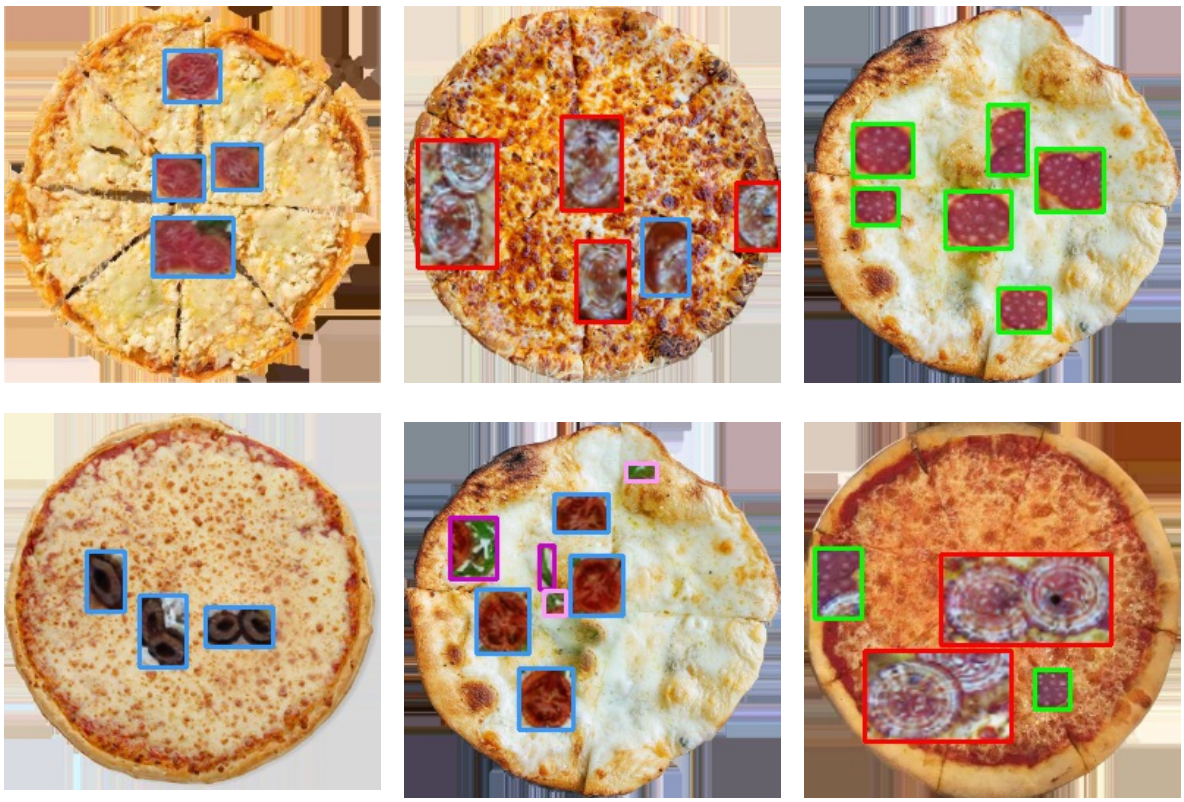
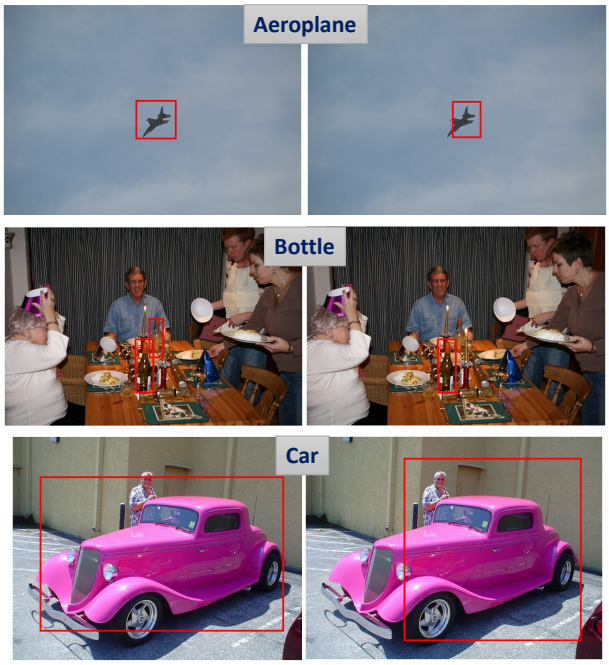


Figure 4: Representative images used for FSOD-3 training.

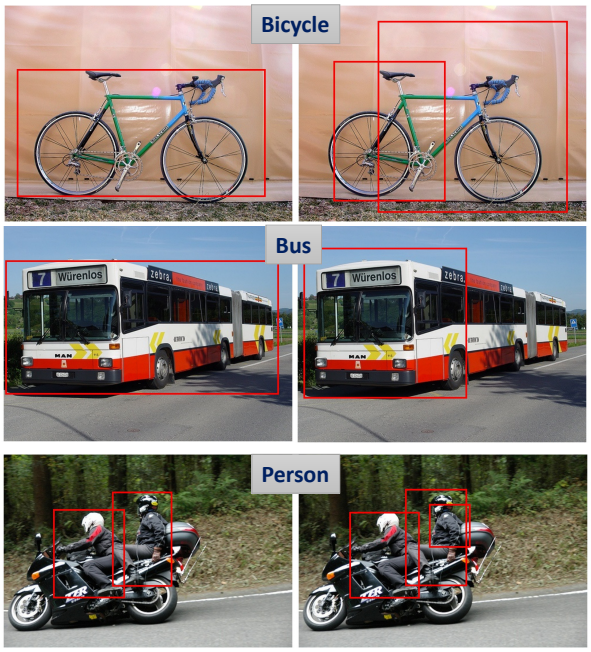
Ours vs. CASD

Ours vs. CASD

Figure 5: Example of success cases for our $\texttt{D}^2\texttt{F2WOD}_{\text{casd}}$ vs. CASD in the test set of RealPizza10 dataset. We only show instances with scores over 0.3 to maintain visibility.



Ours vs. CASD



Ours vs. CASD

Figure 6: Example of success cases for our $\texttt{D}^2\texttt{F2WOD}_{\text{casd}}$ vs. CASD in the test set of VOC2007 dataset. We only show instances with scores over 0.3 to maintain visibility.

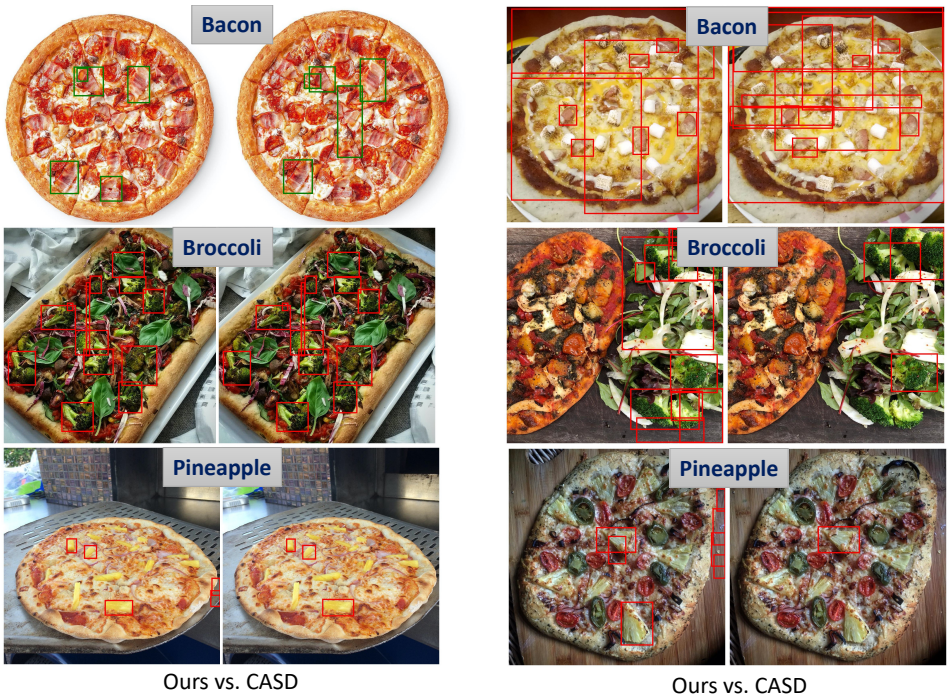


Figure 7: Challenging cases in the test set of Realpizza10 dataset where both our $\texttt{D}^2\texttt{F2WOD}_{\text{casd}}$ and the baseline CASD fail. We hypothesize this is because these three categories have significantly smaller number of training examples. We only show instances with scores over 0.3 to maintain visibility.

References

[1] Aditya Arun, CV Jawahar, and M Pawan Kumar. Dissimilarity coefficient based weakly supervised object detection. In *CVPR*, 2019. 6

[2] Hakan Bilen and Andrea Vedaldi. Weakly supervised deep detection networks. In *CVPR*, 2016. 6

[3] Daniel Bolya, Sean Foley, James Hays, and Judy Hoffman. Tide: A general toolbox for identifying object detection errors. In *ECCV*, 2020. 5

[4] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *IJCV*, 88(2):303–338, 2010. 2

[5] Yan Gao, Boxiao Liu, Nan Guo, Xiaochun Ye, Fang Wan, Haihang You, and Dongrui Fan. C-midn: Coupled multiple instance detection network with segmentation guidance for weakly supervised object detection. In *ICCV*, 2019. 6

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 3

[7] Han-Kai Hsu, Chun-Han Yao, Yi-Hsuan Tsai, Wei-Chih Hung, Hung-Yu Tseng, Maneesh Singh, and Ming-Hsuan Yang. Progressive domain adaptation for object detection. In *WACV*, 2020. 6

[8] Zitong Huang, Yiping Bao, Bowen Dong, Erjin Zhou, and Wangmeng Zuo. W2n: Switching from weak supervision to noisy supervision for object detection. In *ECCV*, 2022. 6

[9] Zeyi Huang, Yang Zou, BVK Kumar, and Dong Huang. Comprehensive attention self-distillation for weakly-supervised object detection. In *NeurIPS*, 2020. 1, 2, 3, 6

[10] Naoto Inoue, Ryosuke Furuta, Toshihiko Yamasaki, and Kiyoharu Aizawa. Cross-domain weakly-supervised object detection through progressive domain adaptation. In *CVPR*, 2018. 2, 3, 6

[11] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014. 2

[12] Dim P Papadopoulos, Youssef Tamaazousti, Ferda Ofli, Ingmar Weber, and Antonio Torralba. How to make a pizza: Learning a compositional layer-based gan model. In *CVPR*, 2019. 2

[13] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NeurIPS*, 2015. 3, 6

[14] Zhongzheng Ren, Zhiding Yu, Xiaodong Yang, Ming-Yu Liu, Yong Jae Lee, Alexander G Schwing, and Jan Kautz. Instance-aware, context-focused, and memory-efficient weakly supervised object detection. In *CVPR*, 2020. 1, 6

[15] Byungseok Roh, JaeWoong Shin, Wuhyun Shin, and Saehoon Kim. Sparse detr: Efficient end-to-end object detection with learnable sparsity. In *ICLR*, 2022. 3

[16] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *IJCV*, 115(3):211–252, 2015. 3

[17] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLP*, 2015. 3

[18] Peng Tang, Xinggang Wang, Song Bai, Wei Shen, Xiang Bai, Wenyu Liu, and Alan Yuille. Pcl: Proposal cluster learning for weakly supervised object detection. *TPAMI*, 42(1):176–191, 2018. 6

[19] Peng Tang, Xinggang Wang, Xiang Bai, and Wenyu Liu. Multiple instance detection network with online instance classifier refinement. In *CVPR*, 2017. 1, 2, 3, 6

[20] Peng Tang, Xinggang Wang, Angtian Wang, Yongluan Yan, Wenyu Liu, Junzhou Huang, and Alan Yuille. Weakly supervised region proposal network and object detection. In *ECCV*, 2018. 6

[21] Fang Wan, Chang Liu, Wei Ke, Xiangyang Ji, Jianbin Jiao, and Qixiang Ye. C-mil: Continuation multiple instance learning for weakly supervised object detection. In *CVPR*, 2019. 6

[22] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *ICCV*, 2019. 3

[23] Zhaoyang Zeng, Bei Liu, Jianlong Fu, Hongyang Chao, and Lei Zhang. Wsod2: Learning bottom-up and top-down objectness distillation for weakly-supervised object detection. In *ICCV*, 2019. 1, 6

[24] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *ICCV*, 2017. 3