Supplementary Material
UniT: Unified Knowledge Transfer for Any-shot Object Detection and Segmentation

Siddhesh Khandelwal[*,1,2] Raghav Goyal[*,1,2] Leonid Sigal[1,2,3]
[1]Department of Computer Science, University of British Columbia
[2]Vector Institute for AI [3]CIFAR AI Chair
skhandel@cs.ubc.ca rgoyal14@cs.ubc.ca lsigal@cs.ubc.ca

A. Semi-Supervised Any-shot Segmentation

In Equation 5 of the main paper we discuss the formulation of the *novel* segmentation head $f_{\mathbf{W}^{seg}_{novel}}(\mathbf{z})$, which is obtained via a structured transfer from the *base* segmentation head $f_{\mathbf{W}^{seg}_{base}}(\mathbf{z})$. We provide more details on how this is implemented in practice. Our implementation of segmentation module can be seen an extension to the Fast R-CNN [7] pipeline described in Section 4 of the main paper. In particular, the segmentation module consists of a transposed-convolution layer (`nn.ConvTranspose2D`), followed by `ReLU`, and a 1×1 convolution layer (`nn.Conv2D`). The feature vector $\mathbf{z}_{i,j}$ for a proposal j in image i is of dimension $(2048 \times 7 \times 7)$ where 2048 is the number of channels and 7 is the spatial resolution of the proposal's feature map. The segmentation module upsamples $\mathbf{z}$ (as in the main paper we drop i, j indexing) using the transposed convolution layer with a kernel size of 2, and then produces a class-specific mask using a 1×1 convolution layer. The resulting mask output is of size $(|\mathcal{C}| \times 14 \times 14)$, where $\mathcal{C}$ is the total number of object classes.

In Eq. 5 of the main paper, $f_{\mathbf{W}^{seg}_{*}}(\cdot)$ is the class-specific output of the segmentation module obtained after the 1×1 convolution. During training, we use the same loss formulation for $\mathcal{L}_{mask}$ as described in [8], where a per-pixel binary cross-entropy loss is used. During inference, the mask is interpolated to fit the regressed proposal (as obtained by Eq.(2) and Eq.(4) in the main paper) to produce the final output. For the semi-supervised zero-shot $(k = 0)$ scenario, the predictions for *novel* classes can be done as in Eq. (5) but omitting the "instance-level direct adaptation" term.

B. Implementation Details

For base-training, we train our model jointly with weak-supervision and base-detection/-segmentation losses with equal weighting (see Section 4.3). In particular, we use image-level data for all the classes to train the weakly-supervised OICR [22] branch, and use detection/segmentation data of base classes for training base detectors/segmentors. Unless pretrained proposals are used, the proposals used for training weakly-supervised branch come from the RPN trained using the base-detection branch. For the zero-shot experiments ($k = 0$) in Section 5.1, similar to the baselines, we replace the RPN with the precomputed MCG proposals [16]. We use 4 Nvidia Tesla T4 GPUs to train models. We build on top of Detectron2 [27] library written in PyTorch [15] framework, and unless mentioned, we keep their default configuration: SGD as the optimizer, RoI Align [8] as the pooling method, ResNet layer sizes/parameters. We use the standard loss for Faster R-CNN, *i.e.*, cross-entropy loss for classifier and smooth-L1 loss for regressor as described in [17].

Note that, in the following text, an iteration refers to a gradient step, and *not* the total number of examples in the training set.

Semi-Supervised Object Detection We train on the MSCOCO 2015 [12] data for semi-supervised zero-shot ($k = 0$) and MSCOCO 2017 [12] for semi-supervised few-shot ($k > 0$) experiments. We use 270K iterations (default in Detectron2 [27]) to account for more data. For fine-tuning, we use 1000 iterations for 12-shot, 3000 iterations for 33-shot, 6000 iterations for 55-shot, 8000 iterations for 76-shot, and 10000 iterations for 96-shot experiments.

Few-shot Object Detection: VOC. We train on VOC 07 + 12 dataset. We use a learning rate of 0.02 over 30K iterations. We decrease the learning rate by a factor of 10 at 12K and 24K iteration.

For fine-tuning, we are given k-shot data for novel classes where $k \in \{1, 2, 3, 5, 10\}$. We linearly scale the number of SGD steps for optimizing over the k-shot data. In particular, we choose 50 iterations for $k = 1$, 100 iterations for $k = 2$, and similarly linearly scale to 500 iterations for $k = 10$.

[*]Denotes equal contribution

#Shots		AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	AR_1	AR_{10}	AR_{100}	AR_S	AR_M	AR_L
$k=0$	UniT (Ours)	18.9	36.1	17.5	8.7	20.4	27.6	19.1	33.3	35.0	16.5	35.5	48.5
$k=10$	Transfer: FRCN [28]	6.5	13.4	5.9	1.8	5.3	11.3	12.6	17.7	17.8	6.5	14.4	28.6
	Kang *et al.* [10]	5.6	12.3	4.6	0.9	3.5	10.5	10.1	14.3	14.4	1.5	8.4	28.2
	Wang *et al.* [26]	7.1	14.6	6.1	1.0	4.1	12.2	11.9	15.1	15.5	1.7	9.7	30.1
	Yan *et al.* [28]	8.7	19.1	6.6	2.3	7.7	14.0	12.6	17.8	17.9	7.8	15.6	27.2
	Wang *et al.* [25]	10.0	-	9.3	-	-	-	-	-	-	-	-	-
	UniT (Ours)	**21.7**	**40.8**	**20.6**	**9.1**	**23.8**	**31.3**	**21.1**	**35.1**	**36.4**	**16.5**	**37.5**	**51.0**
$k=30$	Transfer: FRCN [28]	11.1	21.6	10.3	2.9	8.8	18.9	15.0	21.1	21.3	10.1	17.9	33.2
	Kang *et al.* [10]	9.1	19.0	7.6	0.8	4.9	16.8	13.2	17.7	17.8	1.5	10.4	33.5
	Wang *et al.* [26]	11.3	21.7	8.1	1.1	6.2	17.3	14.5	18.9	19.2	1.8	11.1	34.4
	Yan*et al.* [28]	12.4	25.3	10.8	2.8	11.6	19.0	15.0	21.4	21.7	8.6	20.0	32.1
	Wang *et al.* [25]	13.7	-	13.4	-	-	-	-	-	-	-	-	-
	UniT (Ours)	**23.1**	**43.0**	**21.6**	**9.8**	**25.3**	**33.8**	**22.4**	**36.7**	**37.9**	**16.5**	**38.7**	**53.3**

Table A1: **Complete few-shot object detection results on COCO**. FRCN=Faster R-CNN with ResNet-50 [9] backbone. Similar to [25, 28], we use ResNet-50 as the backbone.

Few-shot Object Detection: COCO. In the case of COCO dataset, we use 270K iterations (default in Detectron2 [27]) to account for more data as compared to VOC. For fine-tuning, we use 500 iterations for 10-shot and 1500 iterations for 30-shot experiment.

Weakly-supervised Object Detection The results are shown in Section E. Here we use a pre-trained VGG-16 [19] as the backbone to be consistent with the prior state-of-the-art works [3, 22, 1, 18]. We use a learning rate of 0.001 over 40K iterations for optimization, dropping it down to 0.0001 for the next 30K iterations.

We will also make all our code **publicly available.**

C. Few-shot Object Detection on MSCOCO

As described in Section 5.2 of the main paper, we compare the performance of UniT against state-of-the-art approaches on the task of Few-Shot Object Detection on the MSCOCO dataset [12]. Please refer to Section 5.2 for task and dataset specifications. Similar to [25, 28], we choose ResNet-50 [9] as the backbone. The k-shot tasks are sampled following [28] with $k \in \{0, 10, 30\}$. Due to lack of space, results in Table 4 of the main paper only compared our model against the baselines on 10-shots. Here we present the complete comparison in Table A1.

As can be seen from Table A1 we get a significant boost in performance even on large number of shots $k = 30$. As UniT uses additional weak image-level data for *novel* classes, this is not an equivalent comparison (see Sec. 5.3 of the main paper for comparisons under similar annotation budget). But we want to highlight that such data is readily available, much cheaper to obtain [2], and leads to a significant improvement in performance.

D. Comparison to Few-shot Instance Segmentation

As described in Section 5.2 of the main paper, we analyse the performance of our proposed approach on the task of Few-shot Instance Segmentation. Please refer to Section 5.2 for task and dataset specifications. Similar to [28], we choose ResNet-50 [9] as the backbone and use an additional segmentation head (as described in Section A of the supplementary). The k-shot tasks are sampled following [28] with $k \in \{0, 5, 10, 20\}$, and we follow the standard evaluation metrics on COCO [8]. The complete results are shown in Table A2. Our approach consistently improves over [28], demonstrating that our approach is not limited to bounding boxes and is able to generalize over the type of downstream structured label, including segmentation mask.

E. Weakly-Supervised Object Detection

Dataset. For completeness, we compare our approach against existing weakly-supervised object detection approaches. We evaluate our approach on VOC 2007 [6] which consists of a trainval set of 5011 images for training and 4951 images for test, keeping in line with the prior related works [3, 22, 21, 24, 1]. As we assume instance-level supervision for *base* classes in the dataset, we report performance on the *novel* classes for three different splits specified in Section 5.2 and [25, 28]. Note that, similar to the baselines, we assume *zero* instance-level supervision for *novel* classes (*i.e.* $k = 0$).

Results. Table A3 provides a summary of the results. Similar to [3, 22, 21, 1, 18], we use a pre-trained proposal network (Selective Search [23]) and an ImageNet pretrained VGG-16 [19] backbone for fair comparison. As we assume additional supervision for *base* classes, this is not an equivalent comparison. However, for *novel* classes, all methods have access to the *same* data. Our results beat the strong baseline of [1] on 2 out of 3 novel splits, and provides comparable performance

		Box						Mask					
#Shots	Method	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
$k=0$	Ours	20.2	36.8	19.5	8.5	20.9	28.9	17.6	32.7	17.0	5.6	17.6	27.7
$k=5$	Yan *et al.* [28]	3.5	9.9	1.2	1.2	3.9	5.8	2.8	6.9	1.7	0.3	2.3	4.7
	Ours	**22.1**	**39.9**	**21.7**	**9.2**	**23.0**	**31.7**	**20.0**	**37.5**	**19.0**	**6.1**	**19.4**	**31.1**
$k=10$	Yan *et al.* [28]	5.6	14.2	3.0	2.0	6.6	8.8	4.4	10.6	3.3	0.5	3.6	7.2
	Ours	**22.8**	**41.6**	**21.9**	**9.4**	**24.4**	**32.3**	**20.5**	**38.6**	**19.7**	**6.0**	**20.5**	**31.8**
$k=20$	Yan *et al.* [28]	6.2	16.6	2.5	1.7	6.7	9.6	6.4	14.8	4.4	0.7	4.9	9.3
	Ours	**23.8**	**42.7**	**23.8**	**9.9**	**24.9**	**34.6**	**21.4**	**39.2**	**21.2**	**6.2**	**20.5**	**33.2**

Table A2: **Complete Table for Few-shot Instance Segmentation on COCO**. All models use ImageNet [4] pretrained ResNet-50 [9] as the backbone.

	Novel split 1						Novel split 2						Novel split 3					
Method	bird	bus	cow	mbike	sofa	mean	aero	bottle	cow	horse	sofa	mean	boat	cat	mbike	sheep	sofa	mean
WSDDN [3]	31.5	64.5	35.7	55.6	40.7	45.6	39.4	12.6	35.7	34.4	40.7	32.6	16.3	42.6	55.6	30.2	40.7	37.1
OICR [22]	31.1	65.1	44.7	65.5	46.9	50.7	58.0	13.0	44.7	37.8	46.9	40.1	19.4	28.4	65.5	41.7	46.9	40.4
PCL [21]	39.3	62.9	52.5	67.7	57.5	56.0	54.4	15.7	52.5	39.3	57.5	43.9	19.2	30.0	67.7	46.6	57.5	44.2
PredNet [1]	52.8	**74.5**	53.0	70.8	**60.7**	62.4	66.7	24.7	53.0	**69.9**	**60.7**	55.0	**31.4**	67.3	70.8	54.6	**60.7**	57.0
Wetectron [18]	**57.0**	69.1	73.2	**77.7**	53.8	**66.2**	**68.8**	**28.9**	73.2	54.4	53.8	**55.8**	27.7	67.0	**77.7**	**64.1**	53.8	58.1
UniT + OICR (Ours)	45.5	71.8	**75.1**	74.0	52.7	63.8	64.0	17.6	**73.8**	59.9	54.4	53.9	30.8	**71.7**	74.9	63.4	55.1	**59.2**

Table A3: **Comparison to Weakly-supervised methods.** Comparison on the three *novel* splits described in Section 5.2 and [28, 25]. All methods use VGG-16 [19] as the backbone.

to the state-of-the-art [18]. Our significant improvement over OICR [22], which we build upon, on *novel* classes ($\sim$35% on average across three splits) highlights the effectiveness of our proposed transfer. We note that our approach is agnostic to the model architecture used for weak-supervision, and the model performance can be improved further if built on top of better weak detectors (*e.g.* [1, 18]).

F. Explanation of Annotation Budget Conversion Factor

In Section 5.3 of the main paper, we perform a constraint annotation budget analysis to facilitate an equivalent comparison to existing few-shot detection works [25, 10], while simultaneously analysing the relative importance of image-level annotations. To this end, for each value of k instance-level annotations in the few-shot setup, we trained a variant of UniT that assumes only $7 \times k$ image-level annotations for *novel* classes, which is referred to as $\text{UniT}_{budget=k}$. Here we describe the rationale behind using this conversion factor of 7 for the VOC dataset [5], derived from annotation times reported in the literature.

Image-Level Annotation (20.0 sec/image). As per [2] and [13], collecting image-level labels takes 1 second per class. As VOC [5] has 20 object classes, generating an image-level annotation is expected to take 20 seconds per image [2].

Instance-Level Bounding Box Annotation (137.2 sec/image). [2] reports an expected annotation time of 239.7 seconds per image for VOC [5]. However, this estimate additionally assumes time required to obtain instance segmentations. As the methods in the few-shot detection domain [25, 10] that we compare against in Section 5.3 only use bounding box annotations to train their models, we modify this estimate of 239.7 seconds to get a more accurate conversion factor.

There are 1.5 object classes per image on average in the VOC 2012 dataset [5]. As mentioned in [2], it takes 1 second to annotate every object class that is not present (*i.e.* obtain an image-level "no object" label). This amounts to $20 - 1.5 = 18.5$ seconds of labeling time. In addition, there are 2.8 object instances on average in the VOC dataset [5]. As mentioned in [20], if we use their efficient system to crowdsource annotations, the median time to get a bounding box annotation is 42.4 seconds (note that the average time is much higher at 88.0 seconds). Therefore, it is expected to take $2.8 \times 42.4 = 118.7$ seconds to obtain bounding box annotations per image in VOC. Thus, the total annotation time is: $18.5 + 118.7 = 137.2$ seconds per image.

It can be seen that obtaining instance-level annotations is approximately $137.2/20 \approx 7$ times more time consuming. Hence, we use a conversion factor of 7 for our experiments in Section 5.3. Also, according to [14], this estimate of 42.4 seconds per bounding box is *conservative*. It doesn't take into account the errors encountered during crowdsourcing

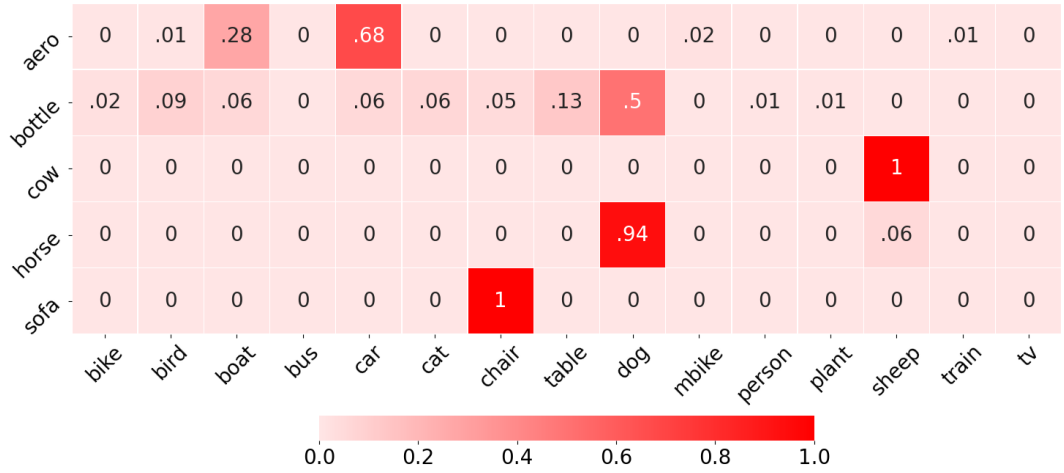


Figure A1: ***Normalized* lingual similarity matrix for the second *novel* split in PASCAL VOC.** Note that $\mathbf{S}^{lin}$ is proposal-agnostic. Most of the similarities are intuitive and semantic – `sofa` is most similar to a `chair`; `horse` to a `dog` and a `sheep`; `cow` is similar to a `sheep`; `aeroplane` is related to other transportation vehicles like `car` and `boat`. A notable departure is a `bottle` which has no closely related categories among *base* classes, resulting in less interpretable similarity and transfer.

Method	$k=0$						$k=5$					
	bird	bus	cow	mbike	sofa	mean	bird	bus	cow	mbike	sofa	mean
weak	58.1	73.5	70.4	68.5	49.1	63.9	59.6	75.9	72.7	71.8	55.1	67.0
weak + avg(Δ)	57.8	73.5	71.2	68.3	47.8	63.7	60.0	76.0	73.6	71.2	54.2	67.0
weak + $\mathbf{S}^{lin}_{cls}$	55.0	74.2	73.6	70.8	46.6	64.1	58.3	76.8	76.1	73.5	52.1	67.4
weak + $\mathbf{S}^{lin}_{cls}$ + $\mathbf{S}^{vis}_{cls}$	56.2	74.6	73.9	71.2	48.8	64.9	59.3	77.1	77.2	73.9	52.0	67.9
weak + $\mathbf{S}^{lin}_{cls,reg}$ + $\mathbf{S}^{vis}_{cls,reg}$	**69.9**	**83.4**	**86.1**	**81.1**	**57.8**	**75.7**	**69.8**	**84.0**	**86.2**	**81.3**	**59.0**	**76.1**

Table A4: **Ablation study on the VOC 07 + 12 dataset.** Please refer to Section G for model definitions. We report AP_{50} on the *first* split in [10], and show results on zero-shot ($k=0$) and few-shot ($k=5$).

(*e.g.* some boxes being rejected and needing to be redrawn). Therefore, we expect this conversion factor of 7 to be higher in practice.

G. Ablation study

Please refer to Section 5.2 of the main paper for a detailed explanation of task setup. We perform ablation over the terms used in Equations (3), (4), and (5) of the main paper on the *novel* classes for the VOC [12] and MSCOCO [12] datasets. We show the benefit of using our proposed transfer both in the zero-shot ($k=0$) and the few-shot ($k>0$) setting. We first describe the ablated variants of our model, and then discuss the results. The results are summarized in Table A4 for VOC and Table A5 for MSCOCO.

We start by forming detectors using only the weakly-supervised branch $f_{\mathbf{W}^{weak}}$ (denoted as "weak" in Tables A4 and A5), and progressively add refinement terms to observe their impact on detection/segmentation performance. We then incorporate the transfer from the *base* classes $f_{\Delta\mathbf{W}^{cls}_{base}}$ into the weak detector (see Equation 3 in the main paper). For each *novel* class, we first compare to a simple baseline approach: averaging over all the *base* classes (denoted by weak + avg(Δ)). We note that a naïve averaging doesn't provide any performance improvements, suggesting the need for a more informed transfer between *base* and *novel* classes. We then explore the role of proposed similarity matrices, detailed in Section 4.2 of the main paper. The similarity matrix between *base* and *novel* classes can be decomposed into two components: lingual similarity $\mathbf{S}^{lin}$ and visual similarity $\mathbf{S}^{vis}(\mathbf{z})$. We analyse the impact of using the aforementioned similarities in obtaining category-aware classifiers, regressors, and segmentors. Following the terms in Eq. (3), (4), and (5) of the main paper, we define ablated variants of our final model.

- "weak + $\mathbf{S}^{lin}_{cls}$" is the model where-in the category-aware classifier for the *novel* classes is obtained by using only the lingual similarity, and the category-aware regressor is fixed to predict zeros (*i.e.* the model uses

Method	Box						Mask					
	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
weak	10.7	28.3	5.1	5.3	12.9	14.3	4.7	17.6	1.1	2.1	6.0	6.0
weak + avg(Δ)	10.7	28.3	5.1	5.3	12.9	14.2	4.7	17.5	1.1	2.0	6.0	6.0
weak + $\mathbf{S}^{lin}_{cls}$	11.4	29.7	5.7	6.3	13.2	14.6	5.0	18.5	1.0	2.3	6.0	6.1
weak + $\mathbf{S}^{lin}_{cls}$ + $\mathbf{S}^{vis}_{cls}$	11.7	29.9	6.0	6.4	13.3	14.7	5.2	18.7	1.2	2.4	6.1	6.1
weak + $\mathbf{S}^{lin}_{cls,reg}$ + $\mathbf{S}^{vis}_{cls,reg}$	20.2	36.8	19.5	8.5	20.9	28.9	8.5	25.0	4.5	2.8	8.7	12.0
weak + $\mathbf{S}^{lin}_{cls,reg,seg}$ + $\mathbf{S}^{vis}_{cls,reg,seg}$	**20.2**	**36.8**	**19.5**	**8.5**	**20.9**	**28.9**	**17.6**	**32.7**	**17.0**	**5.6**	**17.7**	**27.7**

Table A5: **Ablation study on the MSCOCO dataset.** Please refer to Section G for model definitions. The results show the *zero* shot ($k = 0$) performance of the models.

Shot	Method	Novel classes						Base classes																mAP
		bird	bus	cow	mbike	sofa	mean	aero	bike	boat	bottle	car	cat	chair	table	dog	horse	person	plant	sheep	train	tv	mean	
0	Wang *et al*. [25]	-	-	-	-	-	9.0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	**80.8**	62.9
	UniT (Ours) *et al*. [25]	69.9	83.4	86.1	81.1	57.8	**75.6**	82.0	84.7	70.0	69.2	87.9	88.4	60.5	71.3	84.8	86.1	84.5	54.0	82.2	85.1	78.6	78.0	**77.3**
3	Joint: FRCN [28]	13.7	0.4	6.4	0.8	0.2	4.3	75.9	80.0	65.9	61.3	85.5	86.1	54.1	68.4	83.3	79.1	78.8	43.7	72.8	80.8	74.7	72.7	55.6
	Transfer: FRCN [28]	29.1	34.1	55.9	28.6	16.1	32.8	67.4	62.0	54.3	48.5	74.0	85.8	42.2	58.1	72.0	77.8	75.8	32.3	61.0	73.7	68.6	63.6	55.9
	Kang *et al*. [10]	26.1	19.1	40.7	20.4	27.1	26.7	73.6	73.1	56.7	41.6	76.1	78.7	42.6	66.8	72.0	77.7	68.5	42.0	57.1	74.7	70.7	64.8	55.2
	Yan *et al*. [28]	30.1	44.6	50.8	38.8	10.7	35.0	67.6	70.5	59.8	50.0	75.7	81.4	44.9	57.7	76.3	74.9	76.9	34.7	58.7	74.7	67.8	64.8	57.3
	Wang *et al*. [25]	-	-	-	-	-	44.7	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	**79.1**	70.5
	UniT (Ours)	**70.0**	**83.9**	**86.2**	**81.5**	**58.0**	**75.9**	**81.9**	**84.7**	**69.0**	**68.9**	**87.9**	**88.1**	**60.4**	**71.3**	**84.7**	**86.2**	**84.4**	**54.2**	**82.0**	**84.7**	**78.8**	77.8	**77.3**
10	Joint: FRCN [28]	14.6	20.3	19.2	24.3	2.2	16.1	78.1	80.0	65.9	64.1	86.0	87.1	56.9	69.7	84.1	80.0	78.4	44.8	74.6	82.7	74.1	73.8	59.4
	Transfer: FRCN [28]	40.1	47.8	45.5	47.5	47.0	45.6	65.7	69.2	52.6	46.5	74.6	73.6	40.7	55.0	69.3	73.5	73.2	33.8	56.5	69.8	65.1	61.3	57.4
	Kang *et al*. [10]	30.0	62.7	43.2	60.6	39.6	47.2	65.3	73.5	54.7	39.5	75.7	81.1	35.3	62.5	72.8	78.8	68.6	41.5	59.2	76.2	69.2	63.6	59.5
	Yan *et al*. [28]	52.5	55.9	52.7	54.6	41.6	51.5	68.1	73.9	59.8	54.2	80.1	82.9	48.8	62.8	80.1	81.4	77.2	37.2	65.7	75.8	70.6	67.9	63.8
	Wang *et al*. [25]	-	-	-	-	-	56.0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	**78.4**	72.8
	UniT (Ours)	**71.4**	**84.4**	**86.3**	**82.2**	**59.2**	**76.7**	**82.0**	**84.6**	**68.9**	**69.3**	**87.7**	**88.0**	**59.7**	**71.4**	**84.9**	**86.5**	**84.6**	**54.0**	**81.2**	**84.0**	**78.3**	77.7	**77.4**

Table A6: **Base class performance on VOC.** AP and mAP on VOC2007 test set for novel classes and base classes of the first base/novel split. We evaluate the performance for 0/3/10-shot novel-class examples with FRCN under ResNet-101. Note that as Wang *et al*. [25] do not report per-class performance numbers, we just show their reported aggregated performance. Additionally, all models use the *same* amount of annotation for the *base* classes.

the output of the category-agnostic Fast-RCNN regressor **rbox**). As there is no estimate for the *novel* mask head, we predict a uniform mask over the selected bounding box region.

- "weak + $\mathbf{S}^{lin}_{cls}$ + $\mathbf{S}^{vis}_{cls}$" is defined as the model where-in the category-aware classifier for the *novel* classes is obtained by using both lingual and visual similarities (Eq. (3)), and the category-aware regressor is fixed to predict zeros. As there is no estimate for the *novel* mask head, we predict a uniform mask over the selected bounding box region.

- "weak + $\mathbf{S}^{lin}_{cls,reg}$ + $\mathbf{S}^{vis}_{cls,reg}$" is defined as the model where-in both the category-aware classifier and the category-aware regressor for the *novel* classes is obtained by using lingual and visual similarities (Eq. (3)) and (4)). As there is no estimate for the *novel* mask head, we predict a uniform mask over the selected bounding box region. For experiments in Table A4, this is the complete UniT model.

- Finally, "weak + $\mathbf{S}^{lin}_{cls,reg,seg}$ + $\mathbf{S}^{vis}_{cls,reg,seg}$" is defined as the model where-in the category-aware classifier, the category-aware regressor, and the category-aware segmentor for the *novel* classes is obtained by using lingual and visual similarities (Eq. (3), (4), and (5)). For experiments in Table A5, this is the complete UniT model.

From Table A4 it can be seen that adding each of our terms improves model performance. Particularly, transferring information from *base* regressors to *novel* regressors ("weak + $\mathbf{S}^{lin}_{cls,reg}$ + $\mathbf{S}^{vis}_{cls,reg}$") provides a significant boost. We additionally show that this trend holds even when the models are fine-tuned on few-shot ($k = 5$) examples for *novel* classes. Table A5 further highlights the effectiveness of our proposed transfer on segmentation masks ("weak + $\mathbf{S}^{lin}_{cls,reg,seg}$ + $\mathbf{S}^{vis}_{cls,reg,seg}$"). Note that the final two lines in Table A5 only differ in mask AP performance as the regression transfer terms are identical in both the ablated models.

Figure A3 provides qualitative examples to further high-

light the impact of using our proposed transfer from *base* to *novel* classes. Column (a) in Figure A3 refers to "weak", column (b) refers to "weak + $\mathbf{S}^{lin}_{cls} + \mathbf{S}^{vis}_{cls}$", column (c) refers to the "weak + $\mathbf{S}^{lin}_{cls,reg} + \mathbf{S}^{vis}_{cls,reg}$" with $k = 0$, and column (d) refers to the "weak + $\mathbf{S}^{lin}_{cls,reg} + \mathbf{S}^{vis}_{cls,reg}$" after being trained with $k = 5$ shots. It can be seen that the "weak" model either fails to identify all objects or doesn't generate high-probability proposals for the desired objects (column (a)). "weak + $\mathbf{S}^{lin}_{cls} + \mathbf{S}^{vis}_{cls}$" improves performance by generating a bunch of reasonable proposals (column (b)). "weak + $\mathbf{S}^{lin}_{cls,reg} + \mathbf{S}^{vis}_{cls,reg}$" further refines the proposals to obtain accurate bounding boxes for the objects (column (c)). Finally, fine-tuning on $k = 5$ shots improves the bounding box confidence and slightly refines the predictions (column (d)).

H. Analysis of Similarity Matrices

As described in Section 4 of the main paper, the key contribution of our approach is the ability to semantically decompose the classifiers, detectors and segmentors of *novel* classes into their base classes' counterparts. To this end, we define a proposal-aware similarity $\mathbf{S}(\mathbf{z}) \in \mathbb{R}^{|C_{novel}| \times |C_{base}|}$, which is further decomposed into two components: lingual $\mathbf{S}^{lin}$ and visual $\mathbf{S}^{vis}(\mathbf{z})$ similarity. Please refer to Section 4.2 of the main paper on details pertaining to how these similarities are computed.

We qualitatively visualize these similarity matrices to highlight the intuitive semantics learned by our proposed model. Figure A1 shows the *normalized* lingual similarity matrix $\mathbf{S}^{lin} \in \mathbb{R}^{|C_{novel}| \times |C_{base}|}$ for the second *novel* split in PASCAL VOC. Figure A2 shows the *normalized* visual similarity $\mathbf{S}^{vis}(\mathbf{z}) \in \mathbb{R}^{|C_{base}|}$ for each proposal $\mathbf{z}$ (highlighted in blue).

I. Few-shot Performance on VOC's Base Classes

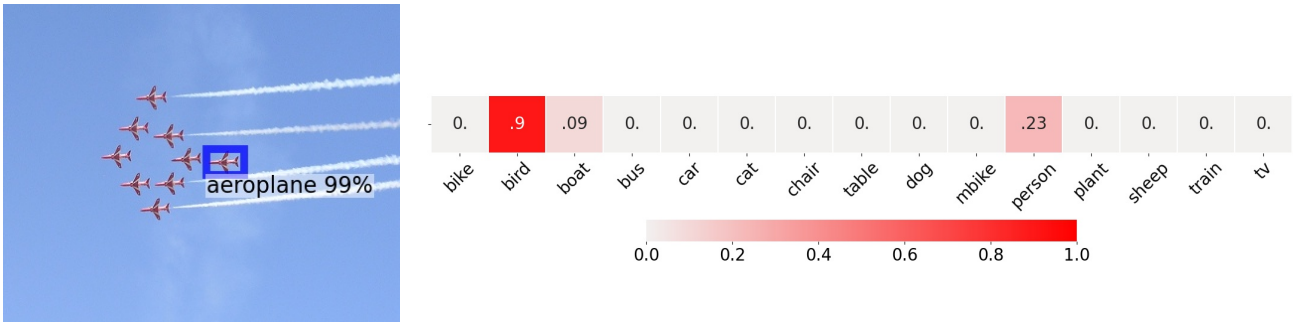
Due to lack of space, in the main paper, we focus on the detection/segmentation results obtained on the *novel* object classes; however, our model also learns to detect/segment *base* class objects as well. We now illustrate that our proposed method improves performance on *novel* classes as the value of k is increased, without any significant performance drop on *base* classes. The experimental setup and baselines are identical to the one described in Section 5.2 of the main paper. Table A6 summarizes results on VOC [6] for the 1-st novel split with k-shots, $k \in \{0, 3, 10\}$.

It is important to note that we are not using any additional annotations for the *base* classes (when compared to [10, 28, 25]). It can be seen that our model's performance on *novel* classes steadily increases with increased k, while simultaneously retaining accuracy on *base* classes.

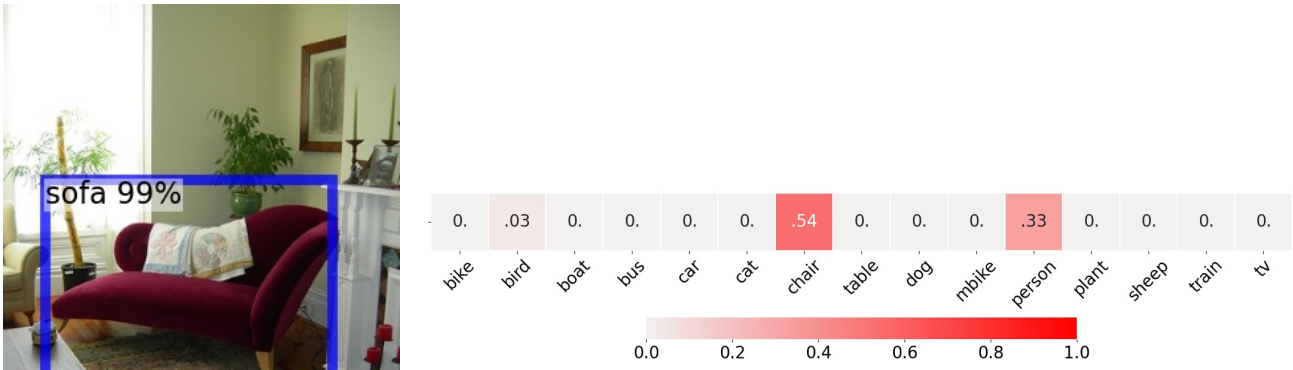
Our slightly poorer performance on *base* classes compared to [25] can be attributed to the fact that [25] use feature pyramid networks (FPN) [11] whereas we don't. According to [11], FPN provides a performance improvement of about 3.8 on AP_{50} (See Table 3 in [11], rows (a) and (c)). Our approach, despite not using FPNs, only performs 2.8 points poorer on $k = 0$. In addition, this gap reduces as the value of k is increased (1.3 AP_{50} on $k = 3$; 0.7 AP_{50} on $k = 10$). Also, compared to [25], UniT has a smaller drop in *base* class performance as k is increased. As an example, when k is increased from 0 to 5, UniT has a performance drop of 0.2 AP_{50} on *base* classes whereas [25] has a larger drop of 1.7 AP_{50}. This observation highlights the fact that our proposed approach is better at retaining *base* class information compared to the existing state-of-the-art in [25]. This improved retention can be mainly attributed to the structured decomposition of our detectors: weak detector + learned refinement. We believe that such a decomposition results in an inductive model bias that leads to convergence to an ultimately better solution.

J. Additional Visualizations on MSCOCO Detection and Segmentation

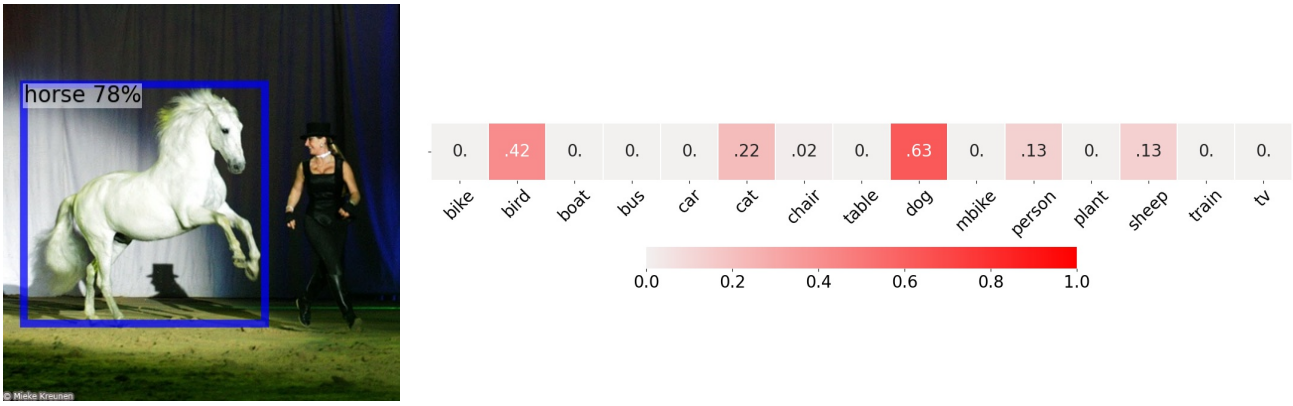
We show additional visualizations highlighting the performance of our approach on the MSCOCO [12] dataset. The experimental setup is identical to the ones described in Sections 5.2 of the main paper. Figure A4 shows additional examples for the task of object detection, and Figure A5 shows additional examples for the task of instance segmentation. Note that these visualizations are generated on *novel* classes under the $k = 0$ setup.



(a) Complementary to $\mathbf{S}^{lin}$ that assigns weights to classes `boat` and `car`, $\mathbf{S}^{vis}(\mathbf{z})$ is additionally able to capture that an `aeroplane` flying in the sky shares some visual characteristics with the class `bird`.

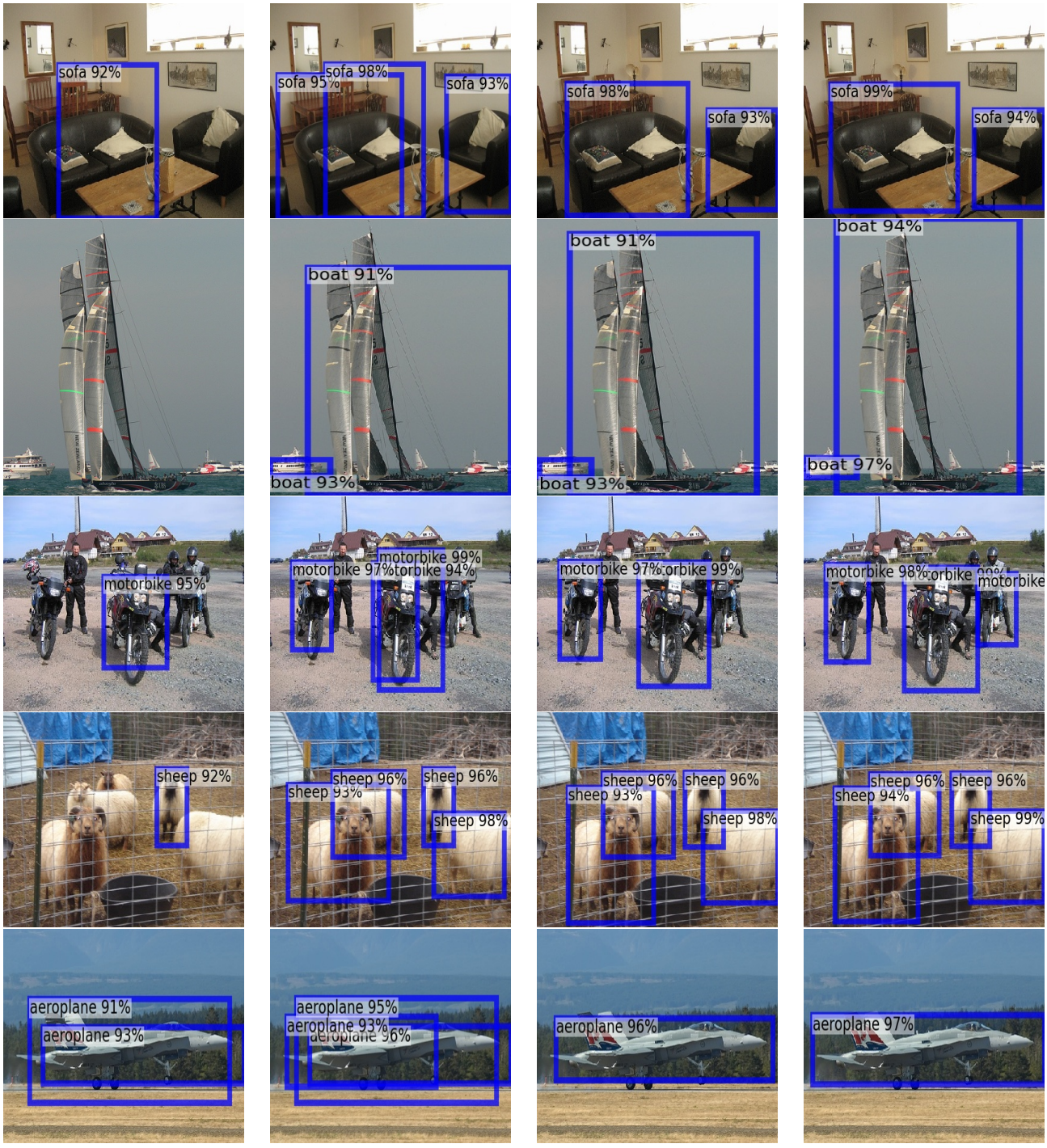


(b) Complementary to $\mathbf{S}^{lin}$ that gives a large weight to the class `chair`, $\mathbf{S}^{vis}(\mathbf{z})$ is additionally able to capture that there is a high correlation between the class `person` and the class `sofa`. This follows the common observation that people are likely to be sitting on a sofa.



(c) Complementary to $\mathbf{S}^{lin}$ that gives a large weight to the class `dog`, $\mathbf{S}^{vis}(\mathbf{z})$ is additionally able to capture that the class `horse` is visually similar to other animal classes `bird`, `cat`, and `sheep`. Additionally, it captures a correlation with the class `person`, which follows from the observation that humans usually ride horses.

Figure A2: **Visual similarity for the second *novel* split in PASCAL VOC.** The input proposal $\mathbf{z}$ is highlighted in **blue**. The visual similarity captures complementary information to the lingual similarity.



(a) (b) (c) (d)

Figure A3: **Qualitative Visualizations for the Ablation Study.** (a) refers to the "weak" model, (b) refers to "weak + $\mathbf{S}_{cls}^{lin}$ + $\mathbf{S}_{cls}^{vis}$", (c) refers to the "weak + $\mathbf{S}_{cls,reg}^{lin}$ + $\mathbf{S}_{cls,reg}^{vis}$" with $k = 0$, and (d) refers to the "weak + $\mathbf{S}_{cls,reg}^{lin}$ + $\mathbf{S}_{cls,reg}^{vis}$" after being trained on $k = 5$ shots. Section G provides a detailed description of these models.

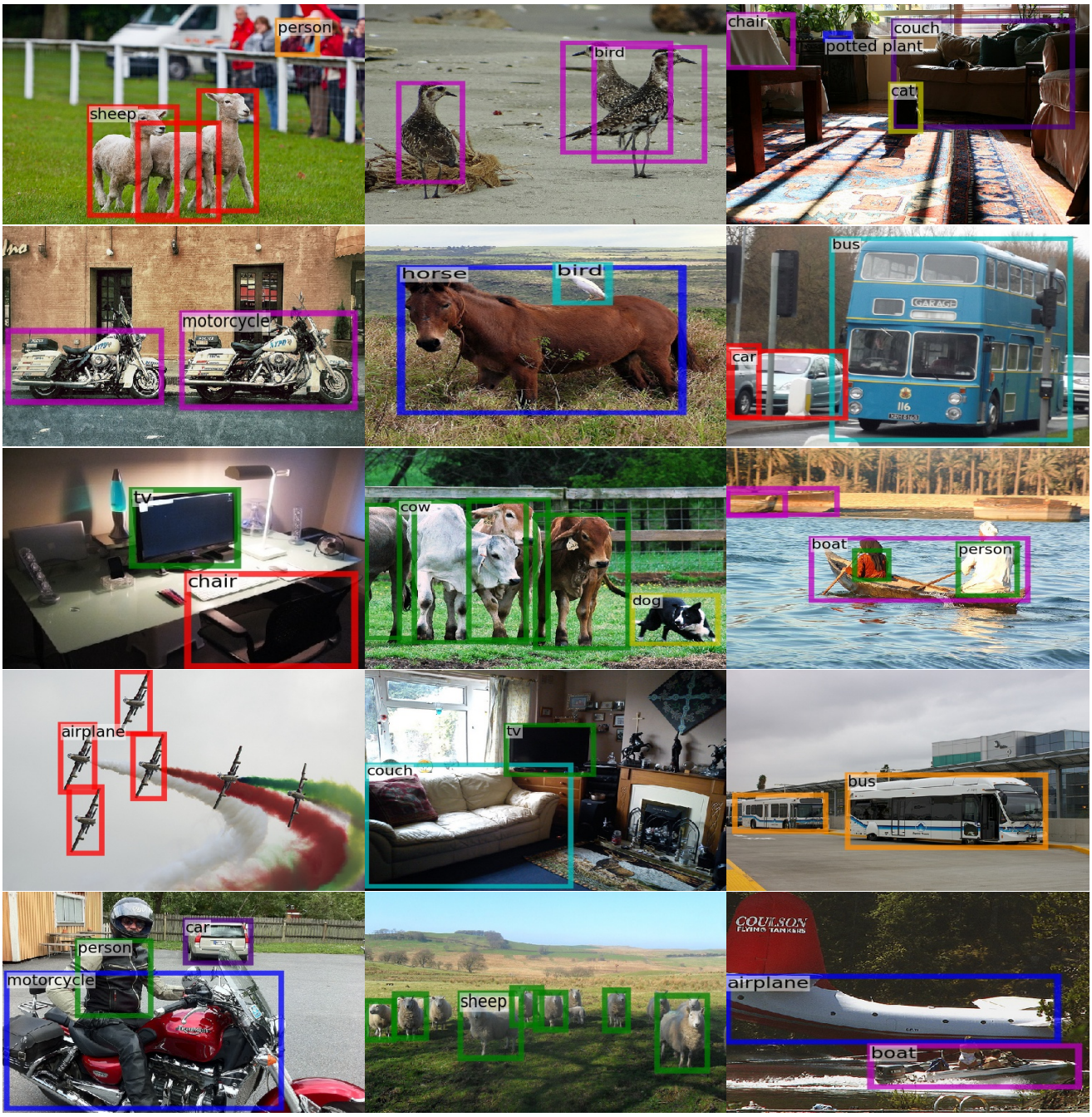


Figure A4: **Qualitative Visualizations.** Semi-supervised zero-shot ($k = 0$) detection performance on *novel* classes in MS-COCO (color = object category).

Figure A5: **Qualitative Visualizations.** Semi-supervised zero-shot ($k = 0$) instance segmentation performance on *novel* classes in MS-COCO (color = object category).

References

[1] Aditya Arun, CV Jawahar, and M Pawan Kumar. Dissimilarity coefficient based weakly supervised object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9432–9441, 2019.

[2] A. Bearman, O. Russakovsky, V. Ferrari, and L. Fei-Fei. Whats the point: Semantic segmentation with point supervision. In *ECCV*, 2016.

[3] Hakan Bilen and Andrea Vedaldi. Weakly supervised deep detection networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2846–2854, 2016.

[4] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[5] Mark Everingham, SM Ali Eslami, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes challenge: A retrospective. *International journal of computer vision*, 111(1):98–136, 2015.

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

[7] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[8] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[10] Bingyi Kang, Zhuang Liu, Xin Wang, Fisher Yu, Jiashi Feng, and Trevor Darrell. Few-shot object detection via feature reweighting. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.

[11] 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.

[12] 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 *European conference on computer vision*, pages 740–755. Springer, 2014.

[13] Dim P Papadopoulos, Alasdair DF Clarke, Frank Keller, and Vittorio Ferrari. Training object class detectors from eye tracking data. In *European conference on computer vision*, pages 361–376. Springer, 2014.

[14] Dim P Papadopoulos, Jasper RR Uijlings, Frank Keller, and Vittorio Ferrari. Training object class detectors with click supervision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6374–6383, 2017.

[15] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pages 8024–8035, 2019.

[16] Jordi Pont-Tuset, Pablo Arbelaez, Jonathan T Barron, Ferran Marques, and Jitendra Malik. Multiscale combinatorial grouping for image segmentation and object proposal generation. *IEEE transactions on pattern analysis and machine intelligence*, 39(1):128–140, 2016.

[17] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[18] 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10598–10607, 2020.

[19] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[20] Hao Su, Jia Deng, and Li Fei-Fei. Crowdsourcing annotations for visual object detection. In *Workshops at the Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012.

[21] Peng Tang, Xinggang Wang, Song Bai, Wei Shen, Xiang Bai, Wenyu Liu, and Alan Yuille. Pcl: Proposal cluster learning for weakly supervised object detection. *IEEE transactions on pattern analysis and machine intelligence*, 42(1):176–191, 2018.

[22] Peng Tang, Xinggang Wang, Xiang Bai, and Wenyu Liu. Multiple instance detection network with online instance classifier refinement. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2843–2851, 2017.

[23] Jasper RR Uijlings, Koen EA Van De Sande, Theo Gevers, and Arnold WM Smeulders. Selective search for object recognition. *International journal of computer vision*, 104(2):154–171, 2013.

[24] Jiajie Wang, Jiangchao Yao, Ya Zhang, and Rui Zhang. Collaborative learning for weakly supervised object detection. *arXiv preprint arXiv:1802.03531*, 2018.

[25] Xin Wang, Thomas E Huang, Trevor Darrell, Joseph E Gonzalez, and Fisher Yu. Frustratingly simple few-shot object detection. *ICML*, 2020.

[26] Yu-Xiong Wang, Deva Ramanan, and Martial Hebert. Meta-learning to detect rare objects. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.

[27] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. https://github.com/facebookresearch/detectron2, 2019.

[28] Xiaopeng Yan, Ziliang Chen, Anni Xu, Xiaoxi Wang, Xiaodan Liang, and Liang Lin. Meta r-cnn: Towards general solver for instance-level low-shot learning. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.