Appendix

The appendix provides additional examples, results and methodological details. For remaining questions, please refer to the code at `github.com/google-research/scenic/tree/main/scenic/projects/owl_vit`.

A1.1 Qualitative Examples

Fig. A1. Text conditioning examples. Prompts: `"an image of a {}"`, where {} is replaced with one of `bookshelf, desk lamp, computer keyboard, binder, pc computer, computer mouse, computer monitor, chair, drawers, drinking glass, ipod, pink book, yellow book, curtains, red apple, banana, green apple, orange, grapefruit, potato, for sale sign, car wheel, car door, car mirror, gas tank, frog, head lights, license plate, door handle, tail lights`.

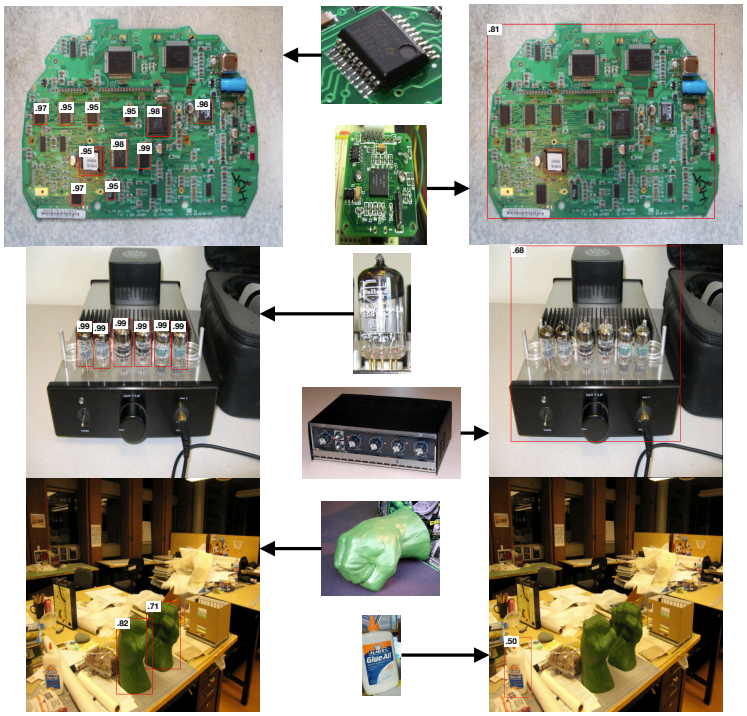

Fig. A2. Image conditioning examples. The center column shows the query patches and the outer columns show the detections along with the similarity score.

A1.2 Detection Datasets

Five datasets with object detection annotations were used for fine-tuning and evaluation in this work. Table A1 shows relevant statistics for each of these datasets:

MS-COCO (COCO) [27]: The Microsoft Common Objects in Context dataset is a medium-scale object detection dataset. It has about 900k bounding box annotations for 80 object categories, with about 7.3 annotations per image. It is one of the most used object detection datasets, and its images are often used within other datasets (including VG and LVIS). This work uses the 2017 train, validation and test splits.

Visual Genome (VG) [23] contains dense annotations for objects, regions, object attributes, and their relationships within each image. VG is based on COCO images, which are re-annotated with free-text annotations for an average of 35 objects per image. All entities are canonicalized to WordNet synsets. We only use object annotations from this dataset, and do not train models using the attribute, relationship or region annotations.

Objects 365 (O365) [35] is a large-scale object detection dataset with 365 object categories. The version we use has over 10M bounding boxes with about 15.8 object annotations per image.

LVIS [13]: The Large Vocabulary Instance Segmentation dataset has over a thousand object categories, following a long-tail distribution with some categories having only a few examples. Similarly to VG, LVIS uses the same images as in COCO, re-annotated with a larger number of object categories. In contrast to COCO and O365, LVIS is a federated dataset, which means that only a subset of categories is annotated in each image. Annotations therefore include positive and negative object labels for objects that are present and categories that are not present, respectively. In addition, LVIS categories are not pairwise disjoint, such that the same object can belong to several categories.

OpenImages V4 (OI) [24] is currently the largest public object detection dataset with about 14.6 bounding box annotations (about 8 annotations per image). Like LVIS, it is a federated dataset.

Table A1. Statistics of object detection datasets used in this work.

Name	Train	Val	Test	Categories
MS-COCO 2017 [27]	118k	5k	40.1k	80
Visual Genome [23]	84.5k	21.6k	-	-
Objects 365 [35]	608.5k	30k	-	365
LVIS [13]	100k	19.8k	19.8k	1203
OpenImages V4 [24]	1.7M	41.6k	125k	601

De-duplication Our detection models are typically fine-tuned on a combination of OpenImages V4 (OI) and Visual Genome (VG) datasets and evaluated on MS-COCO 2017 (COCO) and LVIS. In several experiments our models are additionally trained on Objects 365 (O365). We never train on COCO and LVIS datasets, but the public versions of our training datasets contain some of the same images as the COCO and LVIS validation sets. To ensure that our models see no validation images during training, we filter out images from OI, VG and O365 train splits that also appear in LVIS and COCO validation and tests splits following a procedure identical to [21]. De-duplication statistics are given in Table A2.

Table A2. Train dataset de-duplication statistics. 'Examples' refers to images and 'instances' refers to bounding boxes.

	Original		**Duplicates**		**Remaining**	
Name	Examples	Instances	Examples	Instances	Examples	Instances
OpenImages V4	1.7M	14.6M	948	6.4k	1.7M	14.6M
Visual Genome	86.5k	2M	6.7k	156k	79.8K	1.9M
Objects 365	608.6k	9.2M	147	2.4k	608.5k	9.2M

A1.3 Hyper-parameters

Table A3 provides an exhaustive overview of the hyper-parameter settings used for our main experiments. Beyond this, we

- used cosine learning rate decay;
- used focal loss with $\alpha = 0.3$ and $\gamma = 2.0$;
- set equal weights for the bounding box, gIoU and classification losses [6];
- used the Adam optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.999$;
- used per-example global norm gradient clipping (see Section A1.9);
- limited the text encoder input length to 16 tokens for both LIT and CLIP-based models.

CLIP-based models. The visual encoder of the publicly available CLIP models provides, in addition to the image embedding features, a class token. In order to evaluate whether the information in the class token is useful for detection fine-tuning, we explored to either drop this token, or to merge it into other feature map tokens by multiplying it with them. We found that multiplying the class token with the feature map tokens, followed by layer norm, worked best for the majority of architectures, so we use this approach throughout. Other hyper-parameters used in the fine-tuning of CLIP models are shown in Table A3.

Table A3. List of hyperparameters used for all models shown in the paper. Asterisks (∗) indicate parameters varied in sweeps. MAP and GAP indicate the use of multihead attention pooling and global average pooling for image-level representation aggregation. Where two numbers are given for the droplayer rate, the first is for the image encoder and the second for the text encoder.

	Training duration	Batch size	Learning rate	Weight decay	Image size	Pool type	Training steps	Batch size	Learning rate	Weight decay	Droplayer rate	Image size	Training datasets	Dataset proportions	Mosaic proportions	Random negatives
Model	**Image-level pre-training**						**Detection fine-tuning**									
CLIP-based OWL-ViT models from Table 1:																
B/32							140k	256	5×10^{-5}	0	.2/.1	768	O365, VG	.8/.2	.4/.3/.3	yes
B/16							140k	256	5×10^{-5}	0	.2/.1	768	O365, VG	.8/.2	.4/.3/.3	yes
L/14							70k	256	2×10^{-5}	0	.2/.1	840	O365, VG	.8/.2	.4/.3/.3	yes
LiT-based OWL-ViT models from Table 1:																
B/32	16B	16k	3×10^{-4}	1×10^{-5}	224	MAP	140k	256	2×10^{-4}	0	0.0	768	O365, VG	.8/.2	.4/.3/.3	yes
B/16	8B	16k	3×10^{-4}	1×10^{-5}	224	MAP	140k	256	2×10^{-4}	0	0.0	768	O365, VG	.8/.2	.4/.3/.3	yes
R26+B/32	16B	16k	3×10^{-4}	1×10^{-5}	288	MAP	140k	256	2×10^{-4}	0	0.0	768	O365, VG	.8/.2	.4/.3/.3	yes
L/16	16B	16k	3×10^{-4}	1×10^{-5}	224	MAP	70k	256	5×10^{-5}	0	0.0	768	O365, VG	.8/.2	.4/.3/.3	yes
H/14	12B	16k	3×10^{-4}	1×10^{-5}	224	MAP	70k	256	5×10^{-5}	0	.1/.0	840	O365, VG	.8/.2	.4/.3/.3	yes
Model used for one-shot detection (Table 2):																
R50+H/32	24B	12k	7×10^{-4}	1×10^{-5}	224	GAP	28k	256	2×10^{-4}	0	0.1	960	OI, O365, VG	.4/.4/.2	.5/.33/.17	yes
Baseline models for the ablation study (Tables 3 and A5):																
B/32	2B	16k	3×10^{-4}	1×10^{-5}	224	MAP	70k	256	2×10^{-4}	0	0.0	768	OI, VG	.7/.3	.5/.33/.17	yes
R26+B/32	8B	16k	3×10^{-4}	1×10^{-5}	288	MAP	70k	256	2×10^{-4}	0	0.0	768	OI, VG	.7/.3	.5/.33/.17	yes
Models used in the scaling study (Figures 3 and 4):																
∗	∗	16k	∗	∗	∗	MAP	140k	256	∗	0	0.0	768	OI, VG	.7/.3	.5/.33/.17	no
R50+H/32	∗	12k	7×10^{-4}	1×10^{-5}	224	GAP	28k	256	2×10^{-4}	0	0.0	960	OI, VG	.7/.3	.5/.33/.17	yes

A1.4 Pre-Training Image Resolution

We investigated the effect of the image size used during image-text pre-training, on zero-shot classification and detection performance (Figure A3). To reduce clutter the results are shown for the ViT-B/32 architecture only, but the observed trends extend to other architectures, including Hybrid Transformers. The use of larger images during pre-training consistently benefits zero-shot classification, but makes no significant difference for the detection performance. We thus default to the commonly used 224×224 resolution for pre-training. We used 288×288 for some of our experiments with Hybrid Transformer models.

A1.5 Random Negatives

Our models are trained on federated datasets. In such datasets, not all categories are exhaustively annotated in every image. Instead, each image comes with a

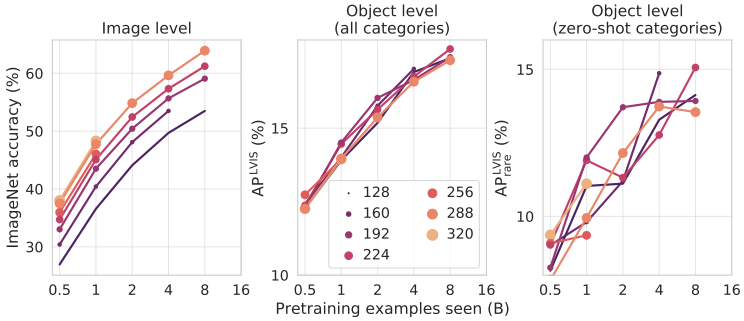

Fig. A3. Effect of image size used during image-level pre-training on zero-shot classification and detection performance shown for the ViT-B/32 architecture.

number of labeled bounding boxes (making up the set of *positive* categories), and a list of categories that are known to be absent from the image (i.e., *negative* categories). For all other categories, their presence in the image unknown. Since the number of negative labels can be small, prior work has found it beneficial to randomly sample "pseudo-negative" labels for each image and add them to the annotations [47]. We follow the same approach and add randomly sampled pseudo-negatives to the real negatives of each image until there are at least 50 negative categories. In contrast to [47], we sample categories in proportion to their frequency in the full dataset (i.e. a weighted combination of OI, VG, and potentially O365). We exclude categories from the sample that are among the positives for the given image.

A1.6 Image Scale Augmentation

To improve invariance of detection models to object size, prior work found it beneficial to use strong random jittering of the image scale during training [11]. We use a similar approach, but follow a two-stage strategy that minimizes image padding.

First, we randomly crop each training image. The sampling procedure is constrained to produce crops with an aspect ratio between 0.75 and 1.33, and an area between 33% and 100% of the original image. Bounding box annotations are retained if at least 60% of the box area is within the post-crop image area. After cropping, images are padded to a square aspect ratio by appending gray pixels at the bottom or right edge.

Second, we assemble multiple images into grids ("mosaics") of varying sizes, to further increase the range of image scales seen by the model. We randomly sample single images, 2×2 mosaics, and a 3×3 mosaics, with probabilities 0.5, 0.33, and 0.17, respectively, unless otherwise noted (Figure A4). This procedure allows us to use widely varying images scales while avoiding excessive padding and/or the need for variable model input size during training.

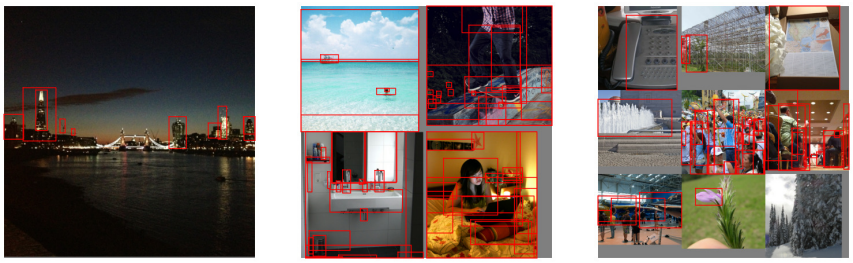

Fig. A4. Example training images. Ground-truth boxes are indicated in red. From left to right, a single image, a 2×2 mosaic, and a 3×3 mosaic are shown. Non-square images are padded at the bottom and right (gray color).

A1.7 One-shot (Image-Conditioned) Detection Details

Extracting Image Embeddings to Use as Queries. We are given a query image patch Q for which we would like to detect similar patches in a new target image, I. We first run inference on the image from which patch Q was selected, and extract an *image embedding* from our model's class head in the region of Q. In general, our model predicts many overlapping bounding boxes, some of which will have high overlap with Q. Each predicted bounding box b_i has a corresponding class head feature z_i. Due to our DETR-style bipartite matching loss, our model will generally predict a single *foreground* embedding for the object in Q and many *background* embeddings adjacent to it which should be ignored. Since all the background embeddings are similar to each other and different from the single foreground embedding, to find the foreground embedding, we search for the most *dissimilar* class embedding within the group of class embeddings whose corresponding box has IoU > 0.65 with Q. We score a class embedding z_i's similarity to other class embeddings as $f(z_i) = \sum_{j=0}^{N-1} z_i \cdot z_j^T$. Therefore, we use the most dissimilar class embedding $\text{argmin}_{z_i} f(z_i)$ as our query feature when running inference on I. In about 10% of the cases, there are no predicted boxes with IoU > 0.65 with Q. In these cases we fall back to using the embedding for the text query `"an image of an object"`.

Image-Conditioned Evaluation Protocol. We follow the evaluation protocol of [16]. During evaluation, we present the model with a target image containing at least one instance of a held-out MS-COCO category and a query image patch containing the same held-out category. Both the target image and the query patch are drawn from the validation set. We report the AP50 of the detections in the target image. Note that unlike typical object detection, it is assumed that there is at least one instance of the query image category within the target image. Like prior work, we use Mask-RCNN [14] to filter out query patches which are too small or do not show the query object clearly. During detection training, we took care to hold out all categories related to any category in the held-out split. We removed annotations for any label which matched a

Table A4. Open-vocabulary detection performance on COCO and O365 datasets. The results show the open-vocabulary generalization ability of our models to datasets that were not used for training. Results for models trained on the target dataset are shown in gray. Most of our models shown here were not trained directly on COCO or O365 (they are different from the models in Table 1). However, we did not remove COCO or O365 object categories from the training data, so these numbers are not "zero-shot". For our models, we report the mean performance over three fine-tuning runs.

Method	Backbone	Image-level	Object-level	Res.	AP^{COCO}	$AP50^{COCO}$	AP^{O365}	$AP50^{O365}$
ViLD [12]	ResNet50	CLIP	LVIS base	1024	36.6	55.6	11.8	18.2
Reg. CLIP [45]	R50-C4	CC3M	COCO base	?	-	50.4	-	-
Reg. CLIP [45]	R50x4-C4	CC3M	COCO base	?	-	55.7	-	-
GLIP [26]	Swin-T	Cap4M	O365, GoldG, ...	?	46.7	-	-	-
GLIP [26]	Swin-L	CC12M, SBU	OI, O365, VG, ...	?	49.8	-	-	-
Detic [46]	R50-C4	CLIP, COCO-Cap	COCO base	1333	-	45.0	-	-
Detic [46]	Swin-B	CLIP, I21K	LVIS base	869	-	-	21.5	-
OWL-ViT (ours)	ViT-B/32	CLIP	OI, VG	768	28.1	44.7	-	-
OWL-ViT (ours)	ViT-B/16	CLIP	OI, VG	768	31.7	49.2	-	-
OWL-ViT (ours)	ViT-L/14	CLIP	O365, VG	840	43.5	64.7	-	-
OWL-ViT (ours)	ViT-B/32	LiT	OI, VG	768	28.0	44.4	9.4	15.2
OWL-ViT (ours)	ViT-B/16	LiT	OI, VG	768	30.3	47.4	10.7	17.0
OWL-ViT (ours)	R26+B/32	LiT	OI, VG	768	30.7	47.2	11.1	17.4
OWL-ViT (ours)	ViT-L/16	LiT	OI, VG	672	34.7	53.9	13.7	21.6
OWL-ViT (ours)	ViT-H/14	LiT	OI, VG	840	36.0	55.3	15.5	24.0
OWL-ViT (ours)	ViT-H/14	LiT	O365, VG	840	42.2	64.5	-	-

held-out label or was a descendant of a held-out label (for example, the label "girl" is a descendant label of "person"). Beyond this we also manually removed any label which was similar to a held-out category. We will publish all held-out labels with the release of our code.

A1.8 Detection results on COCO and O365

We present additional evaluation results on the COCO and O365 datasets in Table A4. These results show the open-vocabulary generalization ability of our approach. Although we do not train these models directly on COCO or O365 (unless otherwise noted), our training datasets contain object categories overlapping with COCO and O365, so these results are not "zero-shot" according to our definition. The breadth of evaluation setups in the literature makes direct comparison to existing methods difficult. We strove to note the differences relevant for a fair comparison in Table A4.

A1.9 Extended Ablation Study

Table A5 extends the ablation results provided in Table 3 of the main text. It uses the same training and evaluation protocol as outlined in Table 3, but goes further in the range of settings and architectures (ViT-B/32 and ViT-R26+B/32) considered in the study. We discuss the additional ablations below.

Dataset ratios. In the majority of our experiments we use OI and VG datasets for training. In the ablation study presented in the main text (Table 3), we showed that having more training data (i.e. training on both VG and OI) improves zero-shot performance. Here, we further explored the optimal ratio in which these datasets should be mixed and found that a 7:3 = OI:VG ratio worked best. Note that this overweighs VG significantly compared to the relative size of these datasets. Overweighing VG might be beneficial because VG has a larger label space than OI, such that each VG example provides more valuable semantic supervision than each OI example.

We also tested the relative value of VG "object" and "region" annotations. In VG, "region" annotations provide free-text descriptions of whole image regions, as opposed to the standard single-object annotations. Interestingly, we found that training on the region annotations hurts the generalization ability of our models, so we do not use them for training.

Loss normalization and gradient clipping. In its official implementation, DETR [6] uses *local* (i.e. per-device) loss normalization and is thus sensitive to the (local) batch size. We found this to be an important detail in practice, which can significantly affect performance. We explored whether normalizing the box, gIoU and classification losses by the number of instances in the image or the number of instances in the entire batch performed better. Our experiments show that per-example normalization performs best, but only *when combined with per-example gradient clipping*, i.e. when clipping the gradient norm to 1.0 for each example individually, before accumulating gradients across the batch. We found that per-example clipping improves training stability, leads to overall lower losses and allows for training models with larger batch sizes.

Instance merging. Federated datasets such as OI have non-disjoint label spaces, which means that several labels can apply to the same object, either due to (near-)synonymous labels (e.g. "Jug" and "Mug"), or due to non-disjoint concepts (e.g. "Toy" and "Elephant" labels both apply to a toy elephant). Due to the annotation procedure, in which a single label is considered at a time, one object can therefore be annotated with several similar (but not identical) bounding boxes. We found it helpful to merge such instances into a single multi-label instance. Multi-label annotations are consistent with the non-disjoint nature of federated annotations and we speculate that this provides more efficient supervision to the models, since it trains each token to predict a single box for all appropriate labels. Without this instance merging, the model would be required to predict individual boxes for each label applying to an object, which clearly cannot generalize to the countless possible object labels.

To merge overlapping instances we use a randomized iterative procedure with the following steps for each image:

1. Pick the two instances with the largest bounding box overlap.
2. If their intersection over union (IoU) is above a given threshold:

2.1. Merge their labels.
2.2. Randomly pick one of the original bounding boxes as the merged instance bounding box.

The picked instances are then removed and the procedure is repeated until no instances with a high enough IoU are left. Having explored multiple IoU thresholds, we note that not merging instances with highly similar bounding boxes is clearly worse than merging them; and that a moderately high threshold of 0.7-0.9 works best in practice.

Learning rates. In Table 3 we show that using the same learning rate for the image and text encoders is clearly sub-optimal, and that it is necessary to training the text encoder with a lower learning rate. This may help to prevent catastrophic forgetting of the wide knowledge the model acquired during the contrastive pre-training stage. Here we explore a range of text encoder learning rates and demonstrate that the learning rate for the text encoder needs to be much lower (e.g. 100×) than that of the image encoder to get good zero-shot transfer (AP_{rare}^{LVIS}). However, freezing the text encoder completely (learning rate 0) does not work well either. AP^{OI}, which measure in-distribution performance, behaves in the opposite way. While using the same learning rate for the image and text encoders results in a big drop in AP_{rare}^{LVIS}, it increases AP^{OI}. This demonstrates that the optimal recipe for zero-shot transfer (AP_{rare}^{LVIS}) does not necessarily maximize in-distribution performance (AP^{OI}).

Cropped bounding box filtering. We use random image crop augmentation when training our models. Upon manual inspection of the resulting images and bounding boxes we noticed a frequent occurrence of instances with degenerate bounding boxes that no longer matched their original instance label (e.g. a bounding box around a hand with label "Person" resulting from cropping most of the person out of the image). To reduce the chance of our models overfitting due to having to memorize such instances, we remove object annotations if a large fraction of their box area falls outside of the random crop area. The optimal area threshold lies between 40% and 60%, and that neither keeping all boxes, nor keeping only uncropped boxes, performs as well (Tables 3 and A1.9).

Mosaics. As described in Appendix A1.6, we perform image scale augmentation by tiling multiple small images into one large "mosaic". We explored mosaic sizes up to 4×4, and found that while using only 2×2 mosaics in addition to single images is clearly worse than also including larger mosaics, for the considered resolutions and patch sizes the benefits of using larger mosaics (i.e. smaller mosaic tiles) saturates with the inclusion of 3×3 or 4×4 mosaics. We have not performed extensive sweeps of the mosaic ratios, and for mosaics with grid sizes from 1×1 (i.e. a single image) to $M \times M$ we use a heuristic of sampling $k \times k$ girds with probability $\frac{2 \cdot (M-k+1)}{M \cdot (1+M)}$, such that smaller mosaics are sampled more frequently than the larger mosaics proportionally to the mosaic size.

Prompting. For generating text queries, similar to prior work, we augment object category names with prompt templates such as `"a photo of a {}"` (where {} is replaced by the category name) to reduce the distribution shift between image-level pre-training and detection fine-tuning. We use the prompt templates proposed by CLIP [33]. During training, we randomly sample from the list of 80 CLIP prompt templates such that, within an image, every instance of a category has the same prompt, but prompt templates differ between categories and across images. During testing, we evaluate the model for each of the "7 best" CLIP prompts and ensemble the resulting predicted probabilities by averaging them. The results in Table A5 show that not using any prompting does not perform well, especially on the in-distribution AP^{OI} metric. Perhaps unsurprisingly, test-time prompt ensembling works better in cases when random prompting was also used during training. In some cases, prompting can have different effects on different model architectures. For example, applying random prompt augmentation to the VG dataset tends to improve performance of the B/32 model, but worsens that of the R26+B/32 model. We speculate that this variability is due to the relatively small number of prompt templates; expanding the list of prompt templates might provide more consistent benefits. We thus only use train-time random prompting for the OI dataset, where it yields consistent benefits.

Location bias. As discussed in the main text, biasing box predictions to the location of the corresponding image patch improves training speed and final performance. The gain is especially large for the pure Transformer architecture (ViT-B/32 in Table A1.9), where removing the bias reduces performance by almost 3 points on $AP^{LVIS}$and AP^{LVIS}_{rare}, whereas the hybrid R26+B/32 drops by only slightly more than 1 point. We therefore speculate that the spatial inductive bias of the convolutional component of the hybrid serves a similar function as the location bias.

Table A5. Additional ablations. VG(obj) and VG(reg) respectively refer to Visual Genome object and region annotations.

	ViT-B/32				ViT-R26+B/32			
Ablation	$\mathbf{AP^{LVIS}}$	$\mathbf{AP^{LVIS}_{rare}}$	$\mathbf{AP^{COCO}}$	$\mathbf{AP^{OI}}$	$\mathbf{AP^{LVIS}}$	$\mathbf{AP^{LVIS}_{rare}}$	$\mathbf{AP^{COCO}}$	$\mathbf{AP^{OI}}$
Baseline	15.7	14.1	24.1	48.5	21.0	18.9	30.9	54.1
Dataset ratio. Baseline uses OI:VG(obj) = 7:3								
OI:VG(obj) = 2:8	−1.9	−2.7	−2.4	−4.8	−4.2	−4.1	−4.7	−4.8
OI:VG(obj) = 3:7	−1.0	−1.9	−1.2	−3.1	−3.0	−3.0	−3.3	−2.9
OI:VG(obj) = 4:6	−0.6	−1.8	−0.4	−1.7	−2.2	−3.6	−2.2	−1.5
OI:VG(obj) = 5:5	0.0	−0.5	0.1	−0.6	−1.0	−1.1	−1.0	−1.1
OI:VG(obj) = 6:4	0.1	−0.6	0.1	−0.3	−0.3	−1.4	−0.4	−0.2
OI:VG(obj) = 8:2	−0.7	−0.9	−0.6	−0.1	−0.4	−0.3	0.2	0.4
OI:VG(obj) = 9:1	−1.8	−1.1	−1.6	0.1	−1.8	−1.8	−1.1	0.3
OI:VG(obj, reg) = 7:3	−0.6	0.0	−0.9	−3.3	−1.2	−0.5	−0.8	−3.6
OI:VG(reg) = 7:3	−2.1	−1.4	−2.3	−2.5	−2.9	−2.3	−2.2	−2.2
Only OI	−4.9	−3.2	−3.5	−0.5	−6.9	−5.7	−4.2	0.3
Only VG(obj)	−8.0	−8.4	−14.2	−28.5	−14.5	−14.0	−23.6	−38.3
Gradient clipping. Baseline uses per-example clipping and per-example normalization.								
Global clip, global norm	−1.0	−2.0	−1.4	−4.9	−2.3	−2.9	−2.8	−5.4
Global clip, per-ex. norm	−4.0	−2.6	−5.3	−4.7	−5.0	−5.0	−5.7	−5.7
Instance merging. Baseline merges instance that overlap with IoU ≥ 0.9								
No merging	−0.8	−1.2	−0.3	−1.2	−0.8	−1.3	−0.6	−0.7
IoU ≥ 0.7	0.2	0.3	−0.2	0.1	0.2	0.2	0.0	0.6
IoU ≥ 0.8	0.0	0.4	0.0	0.4	0.0	−1.3	0.1	0.4
IoU ≥ 0.95	−0.1	−0.1	0.0	−0.7	−0.5	−1.3	−0.2	−0.5
Text encoder learning rate. Baseline uses image LR 2×10^{-4} and text LR 2×10^{-6}.								
LR 2×10^{-3}	−5.1	−10.3	−0.8	−0.6	−7.1	−14.1	−1.4	−0.5
LR 2×10^{-4}	−2.3	−6.7	−0.7	0.2	−3.0	−8.5	−0.5	0.4
LR 2×10^{-5}	−1.1	−3.8	−0.5	0.6	−1.2	−3.2	−0.4	0.9
Do not fine-tune text enc.	−1.8	−1.2	−1.9	−0.7	−1.5	−2.3	−0.6	1.2
Cropped box filtering. Baseline retains boxes with $\geq 60\%$ of their original area.								
No box area filtering	−0.1	−0.3	−0.2	−0.2	−0.1	0.0	0.1	−0.1
$\geq 20\%$ area	−0.3	−1.7	0.0	−0.3	−0.2	−0.8	−0.2	−0.1
$\geq 40\%$ area	0.1	0.0	0.0	0.2	0.1	0.9	0.1	−0.2
Only full boxes	−0.2	−0.9	−0.3	−0.2	−0.1	−0.6	0.1	0.2
Mosaics. Baseline uses 1-to-3-size mosaics at ratio $0.5 : 0.33 : 0.17$								
1–2 @ 2:1	−0.4	−1.1	−0.1	0.4	−0.5	0.3	−0.5	0.0
1–4 @ 4:3:2:1	0.1	0.3	0.0	−0.3	0.0	−0.8	0.1	−0.3
No mosaics	−1.4	−1.6	−1.5	−0.4	−2.3	−1.5	−1.7	−0.7
No mosaics, 2x train sched.	−1.0	−1.8	−0.3	1.2	−2.9	−2.8	−1.8	−0.7
No mosaics, 3x train sched.	−1.2	−3.4	0.3	1.1	−3.4	−3.6	−1.8	−0.8
Prompting. Baseline uses train prompting for OI and test ensemble (ens.) prompting.								
Train: none; test: none	0.0	−0.1	0.8	−10.2	−1.2	−1.3	−0.6	−6.3
Train: none; test: ens.	−2.6	−2.2	−7.3	−11.1	−4.5	−5.0	−10.0	−6.6
Train: OI+VG; test: ens.	0.8	1.3	0.9	−0.1	−0.7	−0.7	−0.4	−0.2
Train: VG; test: ens.	−0.8	−1.1	−2.9	−7.8	−3.1	−4.0	−7.8	−5.6
Other. Baseline uses location bias, samples 50 random negatives and removes LVIS rare labels.								
No location bias	−2.8	−2.9	−3.7	−2.6	−1.2	−1.1	−1.3	−1.0
No random negatives	−1.2	−3.7	−0.8	−0.4	−1.0	−2.8	−0.4	1.0
Keep LVIS rare	0.1	0.9	0.0	0.7	0.1	0.2	−0.1	1.1