PACO: Parts and Attributes of Common Objects (Supplementary)

Vignesh Ramanathan[*1] Anmol Kalia[*1] Vladan Petrovic[*1] Yi Wen[1] Baixue Zheng[1]
Baishan Guo[1] Rui Wang[1] Aaron Marquez[1] Rama Kovvuri[1] Abhishek Kadian[1]
Amir Mousavi[2†] Yiwen Song[1] Abhimanyu Dubey[1] Dhruv Mahajan[1]
[1]Meta AI [2]Simon Fraser University

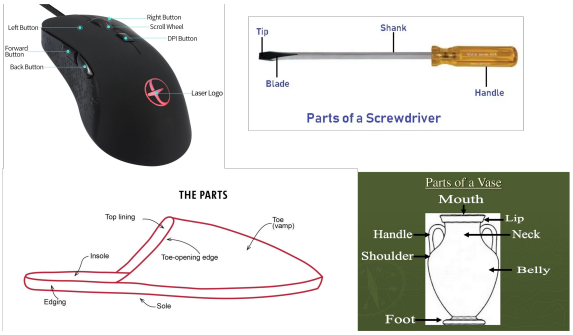

Figure 1. Sample web images used to mine part vocabulary. top-left: "Parts of a computer mouse", top-right: "Parts of a screwdriver", bottom-left: "Parts of a slipper" and bottom-right: "Parts of a vase".

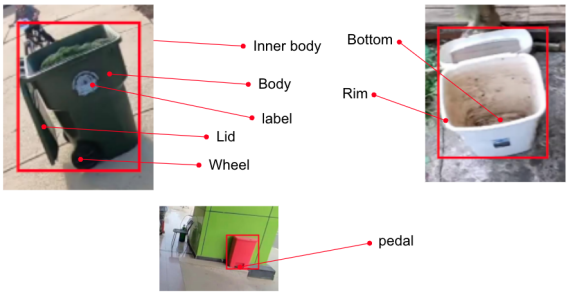

Figure 2. Example object category with manually defined parts. For the object "trash can", we manually defined all the parts with illustrative reference for the annotators as shown.

Appendix

A. Dataset construction

A.1. Parts vocabulary selection

We show sample images obtained by querying the web for "parts of an object category" for different object categories in Fig. 1. We see that the images provide a good vocabulary of parts for each of the objects. Alongside, they also provide clear pointers to the regions of the object the parts correspond to. We use these as reference images for annotators wherever such well defined part images are available from the web. Additionally, we also manually define parts for few objects when the web images aren't illustrative enough. In such cases, we came up with reasonable names for different regions of an object along with reference images to guide the annotators. Such manually defined parts with sample reference images are shown in Fig. 2 as well. Tab. 1 contains the final taxonomy of parts for the 75 object classes.

[*] Equal contribution [†] Work done during internship at Meta AI

A.2. Attribute vocabulary selection

For zero-shot instance recognition tasks, both object and part level attributes are important. In order to identify the set of attributes that we should annotate that are sufficient for the tasks, we conducted an in-depth user study.

We consider the following 5 attributes types: color, shape, reflectance, materials and patterns & marking with the aim of finding a small subset that is sufficient to discriminate between the instances. We show each user two different instances A and B of the same object (green mug and red mug for example), segmentation mask of common object parts between this pair. We use PACO-Ego4D data for this purpose. For object level attributes, we ask annotators to provide at most one difference (if any) for each attribute type. For part level attributes, annotators are asked to compare only between the common parts of the instance pair and they are allowed to annotate up to 3 part level attribute differences for one pair. For an attribute difference, if the discriminative attributes for A and B is nameable (e.g. A is red and B is blue), annotators will need to write down the attribute names. Otherwise, a freeform explanation is required to articulate this difference, particularly for unnameable attributes (e.g. a unique pattern, an irregular shape, etc.); For each object category we sampled 106 pairs each which are annotated by 3 annotators.

Objects	Parts Taxonomy
basket	bottom, handle, inner_side, cover, side, rim, base
belt	buckle, end_tip, strap, frame, bar, prong, loop, hole
bench	stretcher, seat, back, table_top, leg, arm
bicycle	stem, fork, top_tube, wheel, basket, seat_stay, saddle, handlebar, pedal, gear, head_tube, down_tube, seat_tube
blender	cable, handle, cover, spout, vapour_cover, base, inner_body, seal_ring, cup, switch, food_cup
book	page, cover
bottle	neck, label, shoulder, body, cap, bottom, inner_body, closure, heel, top, handle, ring, sipper, capsule, spout, base, punt
bowl	inner_body, bottom, body, rim, base
box	bottom, lid, inner_side, side
broom	lower_bristles, handle, brush_cap, ring, shaft, brush
bucket	handle, cover, body, base, inner_body, bottom, loop, rim
calculator	key, body
can	pull_tab, body, base, inner_body, bottom, lid, text, rim
car_(automobile)	headlight, turnsignal, tank, windshield, mirror, sign, wiper, fender, trunk, windowpane, seat, logo, grille, antenna, hood, splashboard, bumper, rim, handle, runningboard, window, roof, wheel, taillight, steeringwheel
carton	inner_side, tapering_top, cap, bottom, lid, text, side, top
cellular_telephone	button, screen, bezel, back_cover
chair	stretcher, swivel, apron, wheel, leg, base, spindle, seat, back, rail, stile, skirt, arm
clock	cable, decoration, hand, pediment, finial, case, base
crate	bottom, handle, inner_side, lid, side
cup	inner_body, handle, rim, base
dog	teeth, neck, foot, head, body, nose, leg, tail, ear, eye
drill	handle, body
drum_(musical_instrument)	head, rim, cover, body, loop, lug, base
earphone	headband, cable, ear_pads, housing, slider
fan	rod, canopy, motor, blade, base, string, light, bracket, fan_box, pedestal_column
glass_(drink_container)	inner_body, bottom, body, rim, base
guitar	key, headstock, bridge, body, fingerboard, back, string, side, pickguard, hole
hammer	handle, face, head, grip
handbag	zip, inner_body, handle, bottom, body, rim, base
hat	logo, pom_pom, inner_side, strap, visor, rim
helmet	face_shield, logo, inner_side, strap, visor, rim
jar	handle, body, base, inner_body, bottom, lid, sticker, text, rim
kettle	cable, handle, lid, body, spout, base
knife	handle, blade
ladder	rail, step, top_cap, foot
lamp	shade_inner_side, cable, pipe, shade, bulb, shade_cap, base, switch, finial
laptop_computer	cable, camera, base_panel, keyboard, logo, back, screen, touchpad
microwave_oven	inner_side, door_handle, time_display, control_panel, turntable, dial, side, top
mirror	frame
mouse_(computer_equipment)	logo, scroll_wheel, body, right_button, wire, side_button, left_button
mug	handle, body, base, inner_body, bottom, text, drawing, rim
newspaper	text
pan_(for_cooking)	bottom, handle, inner_side, lid, side, rim, base
pen	cap, grip, barrel, clip, tip
pencil	body, lead, eraser, ferrule
pillow	embroidery
pipe	nozzle, colied_tube, nozzle_stem
plastic_bag	inner_body, handle, text, hem, body
plate	top, bottom, inner_wall, body, rim, base
pliers	jaw, handle, joint, blade
remote_control	logo, back, button
scarf	fringes, body
scissors	handle, screw, finger_hole, blade
screwdriver	blade, handle, tip, shank
shoe	toe_box, tongue, vamp, outsole, insole, backstay, lining, quarter, heel, throat, eyelet, lace, welt
slipper_(footwear)	toe_box, vamp, outsole, strap, insole, lining
soap	neck, label, shoulder, body, sipper, capsule, spout, push_pull_cap, cap, base, bottom, closure, punt, top
sponge	rough_surface
spoon	neck, handle, bowl, tip
stool	seat, leg, step, footrest
sweater	shoulder, sleeve, neckband, hem, body, yoke, cuff
table	stretcher, drawer, inner_wall, shelf, apron, wheel, leg, top, rim
tape_(sticky_cloth_or_paper)	roll
telephone	button, screen, bezel, back_cover
television_set	bottom, button, side, top, base
tissue_paper	roll
towel	body, terry_bar, hem, border
trash_can	label, body, wheel, inner_body, bottom, lid, pedal, rim, hole
tray	bottom, inner_side, outer_side, rim, base
vase	neck, handle, foot, body, mouth
wallet	inner_body, flap
watch	buckle, case, dial, hand, strap, window, lug
wrench	handle, head

Table 1. Parts taxonomy

Un-nameable shape attributes. Annotators noted that $> 50\%$ shape differences contain unnameable attributes.

Attribute Type	Attribute Classes
Color	black, light_blue, blue, dark_blue, light_brown, brown, dark_brown, light_green, green, dark_green, light_grey, grey, dark_grey, light_orange, orange, dark_orange, light_pink, pink, dark_pink, light_purple, purple, dark_purple, light_red, red, dark_red, white, light_yellow, yellow, dark_yellow
Pattern-Markings	plain, striped, dotted, checkered, woven, studded, perforated, floral, logo, text
Material	stone, wood, rattan, fabric, crochet, wool, leather, velvet, metal, paper, plastic, glass, ceramic
Reflectance	opaque, translucent, transparent

Table 2. Attributes taxonomy

Annotators reported these differences as very difficult to describe with words. Hence, we removed "shape" from the final list of attribute. Nevertheless, even in the absence of shape we note that the combination of the remaining attributes are seen to be sufficiently discriminative to differentiate the object instances.

Attributes Coverage. We try to identify the discriminative power of different subsets of attributes and identify the best subset to construct our attributes taxonomy. We adopted a greedy algorithm to study attribute sets. We start with one attribute and gradually add one best attribute at a time to incrementally construct an attribute set at each step. More specifically at a give step, for each attribute, we check how many new pairs can be distinguished if we introduce that attribute to the existing set of attributes. The attribute that distinguishes highest number of pairs is selected first, followed by the next best attribute in a greedy fashion. We define coverage of a set of attributes as the total number of object pairs that can be distinguished by the attributes (both with object-level attributes and/or part-level attributes).

We observed that coverage plateaus at 40 attributes. 98% of object instance pairs could be distinguished only using the 55 attributes included in our final version of PACO. Both object and part attributes were marked as important for differentiating instance pairs. 18% instance pairs could only be distinguished by object level attributes, while 10% could only be distinguished by part level attributes. Color is the biggest discriminative attribute type for instance recognition, differentiating at least 75% instance pairs with both object and part level color differences.

The final taxonomy of attributes is shown in Tab. 2.

A.3. Annotation pipeline

A.3.1 Instance annotation

To enable appearance based k-shot instance detection experiments we have annotated instances with unique instance IDs. For LVIS (image) dataset, we assume image of each object to be a separate instance. We inspected several images manually and found this assumption to be true. In Ego4D videos from which we sourced the frames, however, the same instance can occur multiple times at different timestamps and we had to set up an annotation task to properly group occurrences (frames) into instances. There are two challenges that we faced: (a) the same video can contain different instances of the same object class and those have to be split into separate instance IDs, and (b) Ego4D videos are fragmented and multiple videos can contain the same instances so occurrences from different videos had to be merged. To this end we performed a three-step splitting/merging annotation pipeline as follows.

Split: Using (video, category) pair as a good first guess for instance ID, we crop all the bounding boxes (occurrences) of an object category from frames that belong to the same video and show them to annotators. We then ask the annotators to split those crops into subgroups that belong to the same real instance. In case number of boxes is more than 16, we split them in to groups of at most 16 and then send them for annotation. This is then repeated for all object categories and all videos. All annotation jobs are reviewed by 3 annotators and a subset majority voting is performed to aggregate annotations. The majority voting is done by finding the maximum overlap between subgroups for each pair of annotators using Hungarian algorithm (bipartite matching).

Merge: After the splitting phase the annotated groups are very coherent, i.e., the majority of occurrences in the same group belong to the same instance. However due to video fragmentation and additional limitation on the number of boxes that can be shown to annotators (16) many occurrence groups belong to the same instance and need to be merged. To address this we use similarity in DINO model [1] embedding space. Each group from the splitting phase is represented by a bounding box crop with embedding closest to the group median. For each group representative g_i we find 16 nearest neighbors and ask annotators to validate which of the neighbors belong to the same instance as g_i. Similar to the splitting phase, responses from 3 annotators are aggregated by finding the maximum overlap between any two annotators. We repeat this for every group. We then build a graph by considering each group as a node with an edge between two nodes if they belong to the same instance. Nodes i and j are connected if g_j was marked as belonging to the same instance as g_i ***and*** g_i was marked as belonging to the same instance as g_j. Finally, we find connected components and assign a unique instance ID to each component.

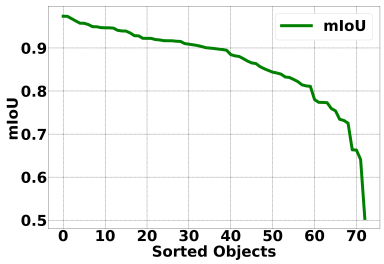

Figure 3. Distribution of mIoU with gold-standard part masks for different object classes. 90% of the object classes have mIoU $\geq$ 0.75 with the gold-standard masks.

Final split: We noticed some over-merging of instances, especially for instances with large number of occurrences. We therefore performed a third step where we showed instances with more than 10 occurrences to expert annotators and asked them to split them into subgroups. Each subgroup at the output of this step is then marked as a separate instance. There was no limit of 16 occurrences in this step, complete instances were shown in each annotation job.

A.3.2 Managing annotation quality

Fig. 3 shows the mIoU of annotated masks with gold set masks for each object category.

B. Dataset annotation examples

Object, part, and attribute annotations are shown in Figs. 4 and 9. Object and part segmentation masks are used to crop out segments for annotations with a specific attribute and shown in Fig. 4 for a subset of attributes. Fig. 9 shows various examples for PACO annotations. Full images are shown with object annotations (bounding boxes only so attributes are visible) in the left copy of the image and part annotations (segmentation masks) in the right copy of the image. Object and part attribute annotations are listed below each image pair.

C. Object statistics

Fig. 5 shows the distribution of instances across the 75 object categories in PACO-LVIS and PACO-EGO4D. All 75 object classes in PACO-LVIS and 71 classes in PACO-EGO4D have $\geq$ 10 instances. We observe the usual non-uniformity in the frequency for each category. For object category 'drill' with the lowest frequency in PACO-LVIS, we have 23 instances, and for 'scarf' with the lowest frequency in PACO-EGO4D data, we have 7 instances.

Model	mask AP		box AP	
	AP^{obj}	AP^{opart}	AP^{obj}	AP^{opart}
R50 FPN	31.2 $\pm$ 0.1	12.1 $\pm$ 0.1	34.3 $\pm$ 0.2	15.7 $\pm$ 0.2
R101 FPN	32.0 $\pm$ 0.3	12.5 $\pm$ 0.1	35.2 $\pm$ 0.3	16.2 $\pm$ 0.2
ViT-B FPN	35.5 $\pm$ 0.5	14.1 $\pm$ 0.3	39.2 $\pm$ 0.5	18.1 $\pm$ 0.5
ViT-L FPN	44.7 $\pm$ 0.4	18.1 $\pm$ 0.3	49.6 $\pm$ 0.4	22.9 $\pm$ 0.4

Table 3. Object and object-part segmentation results for mask-RCNN and ViT-det models trained jointly on PACO-LVIS and PACO-EGO4D and evaluated on PACO-LVIS

Model	mask AP		box AP	
	AP^{obj}	AP^{opart}	AP^{obj}	AP^{opart}
R50 FPN	16.6 $\pm$ 0.3	5.6 $\pm$ 0.0	18.9 $\pm$ 0.3	8.2 $\pm$ 0.1
R101 FPN	17.9 $\pm$ 0.2	6.0 $\pm$ 0.1	20.3 $\pm$ 0.2	8.7 $\pm$ 0.1
ViT-B FPN	18.6 $\pm$ 0.2	7.0 $\pm$ 0.2	20.7 $\pm$ 0.3	10.1 $\pm$ 0.1
ViT-L FPN	27.9 $\pm$ 0.3	10.5 $\pm$ 0.3	30.6 $\pm$ 0.2	14.8 $\pm$ 0.4

Table 4. Object and object-part segmentation results for mask-RCNN and ViT-det models trained jointly on PACO-LVIS and PACO-EGO4D and evaluated on PACO-EGO4D

D. Additional part segmentation and attribute prediction results

In Fig. 6, we show the architecture of the models used to train the joint segmentation and attribute prediction models. For our experiments, we vary the backbones across R-50, R-101 and two ViT-det [6] model backbones.

Examples of predictions from the ViT-L model are shown in Fig. 10.

D.1. Joint training on PACO-LVIS and PACO-EGO4D

In addition to models trained on PACO-LVIS, we also train models for part segmentation and attribute prediction jointly trained on both PACO-LVIS and PACO-EGO4D. We evaluate the jointly trained models on the test splits for both the datasets and present the results for part segmentation in Tab. 3 and Tab. 4. Tab. 5 and Tab. 6 show the results on attribute prediction. We notice that the results on PACO-EGO4D overall are lower compared to those for PACO-LVIS. This is indicative of the challenges in video domain particularly for ego-centric videos. Also, we note that the jointly trained model offers a small improvement compared to model trained only on PACO-LVIS, when evaluate on PACO-LVIS in Tab. 5. We observed 0.2% gain for R50-FPN and R101-FPN and 0.8% improvement for ViT-B FPN, compared to model trained only with PACO-LVIS.

D.2. `val` to `test` results transfer

Here, we wish to study if observations made from the `val` split are similar to the `test` split. This would help us verify if `val` split can be used for model tuning. In Tab. 7 and Tab. 8, we observe that the ranking of results is consis-

Figure 4. Randomly sampled object (top row) and part (bottom row) masks for a subset of attributes (one attribute per column).

Model	AP^{obj}_{att}	AP^{obj}_{col}	AP^{obj}_{pat}	AP^{obj}_{mat}	AP^{obj}_{ref}	AP^{opart}_{att}	AP^{opart}_{col}	AP^{opart}_{pat}	AP^{opart}_{mat}	AP^{opart}_{ref}
R50 FPN	13.8 ± 0.1	10.6 ± 0.4	14.9 ± 0.7	9.7 ± 0.2	19.8 ± 0.9	9.7 ± 0.1	10.3 ± 0.5	10.7 ± 0.5	7.2 ± 0.2	10.7 ± 0.2
R101 FPN	14.0 ± 0.4	11.2 ± 0.3	14.2 ± 0.9	9.8 ± 0.4	20.6 ± 1.6	10.1 ± 0.2	10.8 ± 0.4	11.0 ± 0.3	7.2 ± 0.0	11.3 ± 0.3
ViT-B FPN	16.2 ± 0.6	13.2 ± 0.4	16.7 ± 0.9	13.3 ± 0.3	21.4 ± 1.4	11.5 ± 0.1	12.0 ± 0.1	12.6 ± 0.2	9.4 ± 0.0	11.8 ± 0.4
ViT-L FPN	18.8 ± 0.7	15.6 ± 0.2	19.6 ± 1.1	15.7 ± 0.6	24.5 ± 1.2	14.1 ± 0.1	15.0 ± 0.3	15.2 ± 0.7	11.6 ± 0.1	14.3 ± 0.2

Table 5. Attribute prediction results for a mask R-CNN and ViT-det model trained jointly on PACO-LVIS and PACO-EGO4D and evaluated on PACO-LVIS. The results are shown for box AP for both object attributes and object-part attributes prediction.

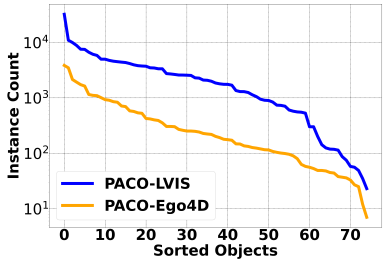

Figure 5. Distribution of instances across the 75 object categories.

tent across `val` and `test`. Across different architectures the trends are similar. This study is similar to what is reported in LVIS [3] for object detection.

D.3. Object segmentation only models

In this section, we explore the effect of joint training on multiple tasks (segmentation and attribute prediction) together on object segmentation results. As an ablation, we train models on only the task of object segmentation for two backbones: R-50 and ViT-L. We report our observations in Tab. 9. For the smaller R-50 backbone, the object segmentation performance deteriorates slightly when joint training with multiple tasks. However, surprisingly for the higher capacity ViT-L backbone, object segmentation improves considerably when training on the joint task.

D.4. Attribute prediction bounds

In the main paper, we report the bounds on AP^{obj}_{att}. The lower bound is calculated by assuming that the score for the object-attribute prediction is the same as the score for the object prediction, i.e., the lower bound performance is the same as if only the detector was used for attribute prediction. The upper bound performance assumes perfect attribute prediction by setting the score for gt attribute to 1.0 and any false positive attribute predictions to 0.0 for a given object prediction. Here, object refers to both object and object-parts.

E. Additional zero-shot instance detection results

We show results for FPN and cascade models trained and evaluated on PACO-LVIS in Tab. 10. Cascade models improve the performance for all but the largest model. In Tab. 11 we also show the results from models trained on the joint PACO dataset and evaluated on PACO-LVIS and PACO-EGO4D test sets. PACO-EGO4D is a more challenging dataset, zero-shot results are in line with attributes prediction results shown in Tab. 6.

F. Ablation studies for zero-shot instance detection

We also measure the importance of different aspects such as object category, object-part category, object colors, part colors and non-color attributes for this end to end task by incrementally including them over a vanilla detection model in Tab. 12. As expected, the object-only performance is poor and each additional component improves the instance detection performance.

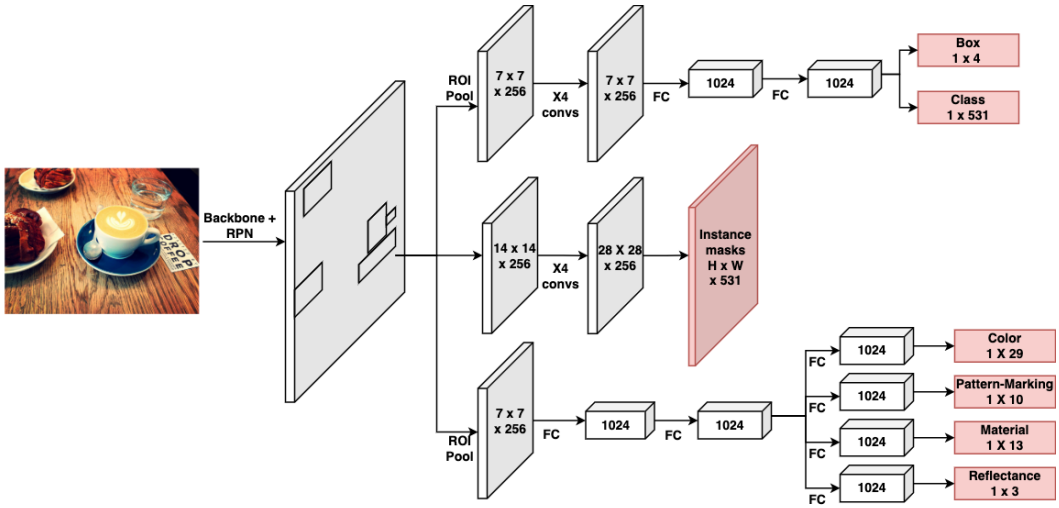

Figure 6. Our model adds an attribute prediction head to Mask R-CNN for joint instance segmentation with attribute prediction

Model	AP^{obj}_{att}	AP^{obj}_{col}	AP^{obj}_{pat}	AP^{obj}_{mat}	AP^{obj}_{ref}	AP^{opart}_{att}	AP^{opart}_{col}	AP^{opart}_{pat}	AP^{opart}_{mat}	AP^{opart}_{ref}
R50 FPN	6.6 $\pm$ 0.4	5.2 $\pm$ 0.2	7.0 $\pm$ 0.3	6.6 $\pm$ 0.8	7.7 $\pm$ 0.4	5.6 $\pm$ 0.1	5.6 $\pm$ 0.5	6.6 $\pm$ 0.6	5.7 $\pm$ 0.3	4.5 $\pm$ 0.3
R101 FPN	7.3 $\pm$ 0.2	5.4 $\pm$ 0.2	7.6 $\pm$ 0.3	8.1 $\pm$ 0.5	8.2 $\pm$ 0.4	5.9 $\pm$ 0.1	5.7 $\pm$ 0.6	7.0 $\pm$ 1.1	6.1 $\pm$ 0.3	4.6 $\pm$ 0.3
ViT-B FPN	8.6 $\pm$ 0.1	6.6 $\pm$ 0.5	10.8 $\pm$ 0.7	8.7 $\pm$ 0.3	8.2 $\pm$ 0.7	7.3 $\pm$ 0.1	6.2 $\pm$ 0.8	10.7 $\pm$ 0.4	6.8 $\pm$ 0.6	5.7 $\pm$ 0.0
ViT-L FPN	11.7 $\pm$ 0.3	9.0 $\pm$ 0.1	13.1 $\pm$ 1.5	12.4 $\pm$ 0.2	12.4 $\pm$ 0.4	10.0 $\pm$ 0.5	7.8 $\pm$ 0.6	12.6 $\pm$ 2.0	9.9 $\pm$ 0.2	9.7 $\pm$ 0.3

Table 6. Attribute prediction results for a mask R-CNN and ViT-det model trained jointly on PACO-LVIS and PACO-EGO4D and evaluated on PACO-EGO4D. The results are shown for box AP for both object attributes and object-part attributes prediction.

Model	split	AP^{obj}	AP^{opart}	AP^{obj}_{att}	AP^{opart}_{att}
R50 FPN	val	38.3 $\pm$ 0.4	18.4 $\pm$ 0.3	20.0 $\pm$ 0.5	17.1 $\pm$ 0.4
	test	34.3 $\pm$ 0.2	15.7 $\pm$ 0.2	13.8 $\pm$ 0.1	9.7 $\pm$ 0.1
R101 FPN	val	39.4 $\pm$ 0.2	18.7 $\pm$ 0.5	21.0 $\pm$ 0.9	17.3 $\pm$ 0.4
	test	35.2 $\pm$ 0.3	16.2 $\pm$ 0.2	14.0 $\pm$ 0.4	10.1 $\pm$ 0.2
ViT-B FPN	val	42.6 $\pm$ 0.7	20.8 $\pm$ 0.7	25.2 $\pm$ 0.5	21.0 $\pm$ 0.3
	test	39.2 $\pm$ 0.5	18.1 $\pm$ 0.5	16.2 $\pm$ 0.6	11.5 $\pm$ 0.1
ViT-L FPN	val	52.6 $\pm$ 0.5	25.9 $\pm$ 0.7	29.2 $\pm$ 0.6	25.9 $\pm$ 0.1
	test	49.6 $\pm$ 0.4	22.9 $\pm$ 0.4	18.8 $\pm$ 0.7	14.1 $\pm$ 0.1

Table 7. We compare how object detection, object-part detection and attribute prediction results transfer from `val` set to `test` set. The models are trained jointly on PACO-LVIS and PACO-EGO4D and evaluated on PACO-LVIS. The ranking is consistent across both splits

Model	split	AP^{obj}	AP^{opart}	AP^{obj}_{att}	AP^{opart}_{att}
R50 FPN	val	37.3 $\pm$ 0.7	18.6 $\pm$ 0.4	28.8 $\pm$ 3.3	21.2 $\pm$ 3.7
	test	18.9 $\pm$ 0.3	8.2 $\pm$ 0.1	6.6 $\pm$ 0.4	5.6 $\pm$ 0.1
R101 FPN	val	38.9 $\pm$ 0.3	19.4 $\pm$ 0.2	30.5 $\pm$ 3.4	22.7 $\pm$ 3.0
	test	20.3 $\pm$ 0.2	8.7 $\pm$ 0.1	7.3 $\pm$ 0.2	5.9 $\pm$ 0.1
ViT-B FPN	val	48.1 $\pm$ 0.3	24.9 $\pm$ 0.1	44.3 $\pm$ 2.2	35.5 $\pm$ 1.0
	test	20.7 $\pm$ 0.3	10.1 $\pm$ 0.1	8.6 $\pm$ 0.1	7.3 $\pm$ 0.1
ViT-L FPN	val	56.1 $\pm$ 0.1	30.8 $\pm$ 0.1	48.8 $\pm$ 3.1	39.8 $\pm$ 0.7
	test	30.6 $\pm$ 0.2	14.8 $\pm$ 0.4	11.7 $\pm$ 0.3	10.0 $\pm$ 0.5

Table 8. We compare how object detection, object-part detection and attribute prediction results transfer from `val` set to `test` set. The models are trained jointly on PACO-LVIS and PACO-EGO4D and evaluated on PACO-EGO4D. The ranking is consistent across both splits

G. From model outputs to query scores

For prediction ranking in the zero-shot instance detection task we need query scores for each detected box. However models trained in Sec. Sec. 5.2 produce object, part, and attribute scores instead. In this section we provide details of how these scores are used to obtain query scores for each box.

Let Q be a query for an object o, with object-level attributes A, parts P and part-level attributes $A_p \forall p \in P$. For example, the query "Black dog with white ear and brown foot" corresponds to o (dog), object-level attributes A ({"black"}), parts P ({"ear", "foot"}), part-level attributes A_p ({"white"} for "ear", {"brown"} for "foot").

Given such a query, we assign a query score to all object boxes in an image. This is a two-step process. In the first step, we associate object-parts detected by our model to the corresponding object boxes in the image. In the second step, we calculate the query score for each object box based on the associated parts.

Part association. Since object-part and object boxes are detected independently by our model, we need to associate

part boxes to objects first. For a given object box, consider all part boxes where the part class corresponds to the object class of the object box, e.g., for a "car" object box, we will only consider predictions for "car-wheel" and not "bicycle-wheel". From these, select part boxes where more than 50% of the part mask area is contained within the object mask. Call these part boxes matched parts. The matched parts may contain multiple occurrences of the same object-part class, keep only the one with the highest score. This results in set of matched parts for each object box. For some objects, we may have no matched parts for a specific object-part (eg: we may find no "car-wheel" matched with a "car" box).

Query score. For a given object box b, let the predicted score for the query object category o be given by o_o. Similarly, let the predicted object attribute scores be a_k for $k \in A$. Similarly, the part scores of the matched object-parts are given by p_p for $p \in P$. These are the predicted category scores for the matched part box corresponding to each object-part category mentioned in the query. For an object-part category if no part box is matched to b, this score is set to 0. We also have attribute scores for each matched object-part $a_{p,k}$ for $p \in P, k \in A_p$. These scores are again set to 0 if no part box of the corresponding object-part category is matched with b. For the query Q, the score is then computed as follows:

$$s(Q,b) = \begin{cases} \text{if } |A| > 0: & \sqrt{o_o \times \sqrt[|A|]{\prod_{k \in A} a_k}} \\ \text{otherwise}: & o_o \end{cases} \times \begin{cases} \text{if } |P| > 0: & \frac{\sum_{p \in P} \sqrt{p_p \times \sqrt[|A_p|]{\prod_{k \in A_p} a_{p,k}}}}{|P|} \\ \text{otherwise}: & 1 \end{cases}$$

This is repeated for all queries and all detected boxes.

The above scoring function combines the scores of the object, object-attribute, parts and part-attributes mentioned in the query. Note that the first part of the scoring function only combines object and object-attribute scores, while the second part combines part and part-attribute scores. While combining part scores we use a combination of arithmetic and geometric means. We found this combination to provide the best results empirically.

H. Evaluation of open world detectors on zero-shot instance detection task

In this section we give details on how we evaluated Detic [10] and MDETR [5] on zero-shot instance detection task. For both projects we used code open sourced on GitHub.

Detic supports a custom vocabulary and encodes natural language class descriptions using pre-trained CLIP text encoder. We used all $5k$ queries as custom vocabulary so that we have prediction scores for all queries for each detected box. Due to large vocabulary we had to increase the number of detections per image. We experimented with this parameter and found that $2,000$ boxes gives the best results. We used plain query strings (e.g., "A dog with brown ear and black neck") from PACO dataset as class descriptions along with 3 more prompt variants with prefixes "A photo of", "A close up picture of", and "A close up photo of" in front of the plain query strings. The "close up" variants were an attempt to guide text embeddings closer to a detection setup but we didn't see much improvement in performance. We use `Detic_LCOCOI21k_CLIP_SwinB_896b32_4x_ft4x_max-size.pth` model and report mean and standard deviation $AR@k$ calculated over results from these 4 prompt variants.

Model	mask AP AP^{obj}	box AP AP^{obj}
R50 FPN	31.2 $\pm$ 0.1	34.3 $\pm$ 0.2
R50 FPN - object only	32.4 $\pm$ 0.6	35.5 $\pm$ 0.5
ViT-L FPN	44.7 $\pm$ 0.4	49.6 $\pm$ 0.4
ViT-L FPN - object only	39.8 $\pm$ 0.1	43.6 $\pm$ 0.1

Table 9. Comparison of model performance on object segmentation when trained only on the task of object segmentation vs joint training on object and part segmentation and attribute prediction.

MDETR is geared towards referring expressions and phrase grounding and treats each image-text pair independently. We follow inference similar to LVIS evaluation reported in the MDETR paper. Namely, for inference on a given image, we evaluate the model on each of the $5k$ queries separately, then merge the sets of boxes detected on each of the queries and keep the boxes corresponding to top K query scores. Unlike Detic, predicted boxes are not shared across queries since MDETR predicts bounding boxes independently for each query. As a result, we had to increase the number of detections per image even further to $10,000$ to obtain the best results. We also experimented with two MDETR models with R101 backbone [1], one trained for referring expressions task (`refcocog_resnet101_checkpoint.pth`) and the other for LVIS few-shot task (`lvis10_checkpoint.pth`) and observed that LVIS few-shot task model performs better. We report mean and standard deviation of results from that model over the same 4 query prompt variants as Detic.

I. Challenge test set splits

To facilitate zero-shot instance detection challenge competitions on PACO-EGO4D we perform an additional split of the test data. We split the full test dataset 50/50 to limit the variance increase due to smaller number of queries and call the two splits `test-dev` and `test-std`. We show

[1] A known issue (#86) prevented the use of ENB backbones

Model	L1 queries		L2 queries		L3 queries		all queries	
	$AR@1$	$AR@5$	$AR@1$	$AR@5$	$AR@1$	$AR@5$	$AR@1$	$AR@5$
R50 FPN	22.5 ± 0.7	39.2 ± 0.5	20.1 ± 0.4	38.5 ± 0.1	22.3 ± 0.9	44.5 ± 1.1	21.4 ± 0.6	40.9 ± 0.3
+ cascade	23.5 ± 1.4	41.1 ± 2.7	21.4 ± 2.4	40.9 ± 3.2	25.3 ± 2.7	48.1 ± 3.2	23.3 ± 2.3	43.7 ± 3.1
R101 FPN	23.1 ± 0.7	40.5 ± 1.4	20.0 ± 0.6	39.3 ± 1.0	23.1 ± 0.7	45.2 ± 0.6	21.7 ± 0.6	41.8 ± 0.8
+ cascade	26.3 ± 0.4	45.1 ± 0.5	24.0 ± 0.1	43.2 ± 0.1	26.6 ± 1.2	49.5 ± 0.8	25.4 ± 0.5	45.9 ± 0.4
ViT-B FPN	26.8 ± 0.2	45.8 ± 0.2	22.7 ± 0.5	40.0 ± 0.7	24.1 ± 0.5	42.5 ± 1.5	23.9 ± 0.4	42.0 ± 0.9
+ cascade	27.0 ± 0.4	46.1 ± 0.5	23.0 ± 0.9	40.3 ± 0.2	25.5 ± 0.8	43.1 ± 0.5	24.7 ± 0.7	42.4 ± 0.2
ViT-L FPN	35.3 ± 0.7	57.3 ± 0.6	29.7 ± 0.6	50.1 ± 0.2	31.1 ± 0.8	52.3 ± 0.9	31.2 ± 0.4	52.2 ± 0.5
+ cascade	33.8 ± 0.7	57.2 ± 0.2	29.0 ± 0.7	50.2 ± 0.2	30.1 ± 0.7	51.8 ± 1.8	30.2 ± 0.6	52.0 ± 0.6

Table 10. Zero-shot instance detection results for different query levels for FPN and cascade models from Sec. Sec. 5.2 trained and evaluated on PACO-LVIS.

Model	Eval set	L1 queries		L2 queries		L3 queries		all queries	
		$AR@1$	$AR@5$	$AR@1$	$AR@5$	$AR@1$	$AR@5$	$AR@1$	$AR@5$
R50 FPN	PACO-LVIS	22.0 ± 0.4	39.6 ± 0.6	20.6 ± 0.5	39.0 ± 0.7	24.7 ± 1.0	45.5 ± 1.4	22.4 ± 0.3	41.6 ± 0.7
R101 FPN	PACO-LVIS	23.5 ± 0.5	40.9 ± 0.4	21.2 ± 0.3	40.1 ± 0.7	24.3 ± 1.3	45.2 ± 0.9	22.8 ± 0.5	42.2 ± 0.6
ViT-B FPN	PACO-LVIS	29.5 ± 0.6	49.5 ± 1.1	25.8 ± 1.4	44.9 ± 2.3	26.2 ± 1.2	45.7 ± 2.9	26.6 ± 1.1	46.0 ± 2.2
ViT-L FPN	PACO-LVIS	38.0 ± 0.6	60.8 ± 1.2	33.3 ± 1.7	55.6 ± 1.9	33.1 ± 2.6	59.0 ± 2.8	34.0 ± 1.8	57.8 ± 2.1
R50 FPN	PACO-EGO4D	15.4 ± 0.1	29.1 ± 0.6	13.2 ± 0.2	28.0 ± 0.9	14.4 ± 1.8	29.1 ± 1.3	14.2 ± 0.9	28.7 ± 0.8
R101 FPN	PACO-EGO4D	16.3 ± 0.5	29.8 ± 0.9	15.0 ± 0.6	28.6 ± 0.7	14.2 ± 0.6	28.3 ± 0.9	14.9 ± 0.1	28.6 ± 0.5
ViT-B FPN	PACO-EGO4D	13.5 ± 1.2	24.4 ± 1.3	11.0 ± 0.4	19.5 ± 0.7	9.3 ± 0.5	18.1 ± 0.5	10.6 ± 0.1	19.7 ± 0.4
ViT-L FPN	PACO-EGO4D	20.8 ± 0.2	36.9 ± 0.7	19.8 ± 1.3	33.3 ± 1.3	21.4 ± 1.2	34.9 ± 0.7	20.7 ± 1.0	34.7 ± 0.9

Table 11. Zero-shot instance detection results for different query levels for FPN models from Sec. Sec. 5.2 trained on joint PACO dataset and evaluated on PACO-LVIS and PACO-EGO4D.

Score components	all queries	
	$AR@1$	$AR@5$
Object only	1.9 ± 0.5	8.2 ± 0.2
Object + part	2.4 ± 0.4	10.8 ± 0.9
Object + color	5.6 ± 0.5	15.5 ± 0.1
Object + attribute	8.5 ± 0.4	22.3 ± 0.2
Object + part + color	20.8 ± 0.6	40.2 ± 0.6
All	31.2 ± 0.4	52.2 ± 0.5

Table 12. Ablation study on importance of object, part, and attribute predictions on zero-shot instance detection performance. We compute metrics using only object, object + part, object + color, object + attribute, object + part + color, and all ViT-L FPN model scores.

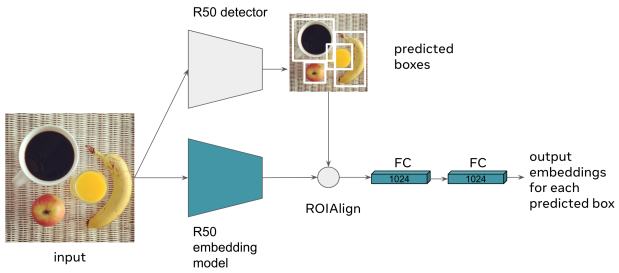

Figure 7. The few-shot instance detection model consists of a frozen detector and an embedding model. The detector outputs class-agnostic bounding boxes. The embedding model takes an image and a set of predicted bounding boxes on the image as inputs, and outputs embeddings for every box.

results for the two splits in Tab. 13 corresponding to full test set results in Tab. 11. As expected, the variance on the smaller splits increases compared to the full set, however the results on both splits and the full dataset are close and follow similar trends.

J. Few-shot instance detection experiments

The few-shot model is a two-tower model as shown in Fig. 7, where the (a) first tower is a detection model which predicts object boxes in the images and (b) the second tower is an embedding model that provides a feature embedding for each of the predicted boxes. The two towers are learned independently.

Query feature registration. In the few-shot setup, for each query Q we are provided a set of "query images" with one bounding box per image for the "query" object instance. We first extract a feature for each of query boxes only using the embedding model. Given k query images (with bounding box) for the query Q, we extract k query features. The features are then averaged to obtain an average query feature vector f_Q.

Instance detection with query features. We are also provided a set of target images for each query $\mathcal{I}_Q$ from which another bounding box corresponding to the query needs to be extracted. For each image $I \in \mathcal{I}_Q$, we first predict 100 bounding boxes $\mathcal{B}_I$ using the detection tower of our model. Each of these boxes $B \in \mathcal{B}_I$ are then represented by a fea-

Model	Eval set	L1 queries		L2 queries		L3 queries		all queries	
		AR@1	AR@5	AR@1	AR@5	AR@1	AR@5	AR@1	AR@5
R50 FPN	PACO-EGO4D-dev	16.1 ± 0.2	29.2 ± 0.8	11.7 ± 1.1	26.4 ± 1.8	13.2 ± 2.4	27.6 ± 2.8	13.2 ± 1.3	27.5 ± 1.8
R101 FPN	PACO-EGO4D-dev	15.6 ± 0.4	27.9 ± 0.6	13.3 ± 1.7	26.3 ± 1.7	13.2 ± 0.7	25.7 ± 1.3	13.7 ± 0.9	26.3 ± 1.3
ViT-B FPN	PACO-EGO4D-dev	13.4 ± 0.8	23.4 ± 1.3	10.0 ± 0.8	18.3 ± 1.4	8.0 ± 0.6	14.9 ± 1.2	9.6 ± 0.7	17.6 ± 1.1
ViT-L FPN	PACO-EGO4D-dev	21.7 ± 0.5	36.8 ± 1.3	20.8 ± 1.2	33.2 ± 1.2	25.0 ± 1.9	36.5 ± 1.0	23.0 ± 1.3	35.5 ± 0.9
R50 FPN	PACO-EGO4D-std	14.6 ± 0.3	28.9 ± 0.8	14.9 ± 1.4	29.8 ± 1.5	15.8 ± 1.3	30.8 ± 1.4	15.3 ± 1.0	30.1 ± 1.2
R101 FPN	PACO-EGO4D-std	17.1 ± 1.1	31.7 ± 1.7	16.8 ± 0.8	31.1 ± 1.2	15.3 ± 1.6	31.1 ± 1.5	16.2 ± 0.8	31.2 ± 1.0
ViT-B FPN	PACO-EGO4D-std	13.6 ± 1.8	25.4 ± 1.3	12.1 ± 0.4	20.7 ± 0.2	10.8 ± 1.4	21.7 ± 0.6	11.8 ± 0.5	22.1 ± 0.4
ViT-L FPN	PACO-EGO4D-std	20.0 ± 0.4	37.1 ± 0.2	18.6 ± 1.7	33.3 ± 1.5	17.2 ± 2.0	33.2 ± 1.4	18.2 ± 1.3	33.9 ± 1.1

Table 13. Zero-shot instance detection results for different query levels for FPN models from Sec. Sec. 5.2 trained on joint PACO dataset and evaluated on PACO-EGO4D `test-dev` and `test-std` splits.

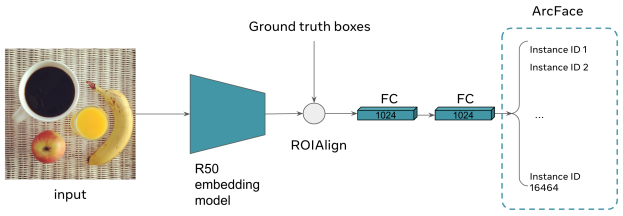

Figure 8. The embedding model is a mask R-CNN style model with a custom ROI head where the softmax loss is replaced with an ArcFace loss using instance IDs as supervision for richer representations for instance recognition. Once training is finished, we throw away the ArcFace layer and use the outputs from the last FC layer as per-box representations.

ture vector $f_{B,I}$ using the embedding model. All the boxes are then ranked based on the cosine similarity of their feature with the query feature f_Q. The top N returned boxes from $\mathcal{I}_Q$ are used to compute AR@N for $N = 1, 5$.

Detection model. We train a standard R50-FPN mask R-CNN model with 75 object categories on the `train` split of the PACO dataset. During the feature registration and instance detection stage, we ignore the category label and only use the predicted boxes.

Embedding model. The embedding model is a mask R-CNN style model with a custom ROI head as shown in Fig. 8. During inference, it takes predicted bounding boxes as input and outputs embeddings for each box with ROIAlign [4]. The model is trained with ArcFace [2] loss to have richer representations for instance recognition. We trained the embedding model with an ArcFace loss to perform 16464-way instance ID classification at box-level. The model was trained to distinguish the 16464 different object instances in the PACO-Ego4d `train` split. In this dataset, each instance has multiple bounding boxes, making it possible to train such a model. We simply use ground truth boxes during training to avoid handling the additional complexity from distinguishing foreground and background boxes. Note that the sets of instances in `train` and `test` splits are completely disjoint.

Implementation details. We use R50-FPN [7] as the backbone. The custom ROI head is implemented as a ROIAlign operator followed by 2 FC layers with 1024 dimensions. The ArcFace layer is configured with *margin = 0.5* and *scale = 8.0*. We use the default data augmentation for Faster R-CNN [8] training in Detectron2 [9]. We train the embedding model on the PACO-Ego4D `train` split for 22.5K iterations. We set *lr = 0.04* and use Cosine *lr* decay. The batch size is 128 distributed across 32 GPUs (4 images per GPU).

Figure 9. Annotation examples. Each image contains object bounding boxes (object masks omitted so attributes are visible) on the left and part masks on the right. Object and part attributes are listed below each image.

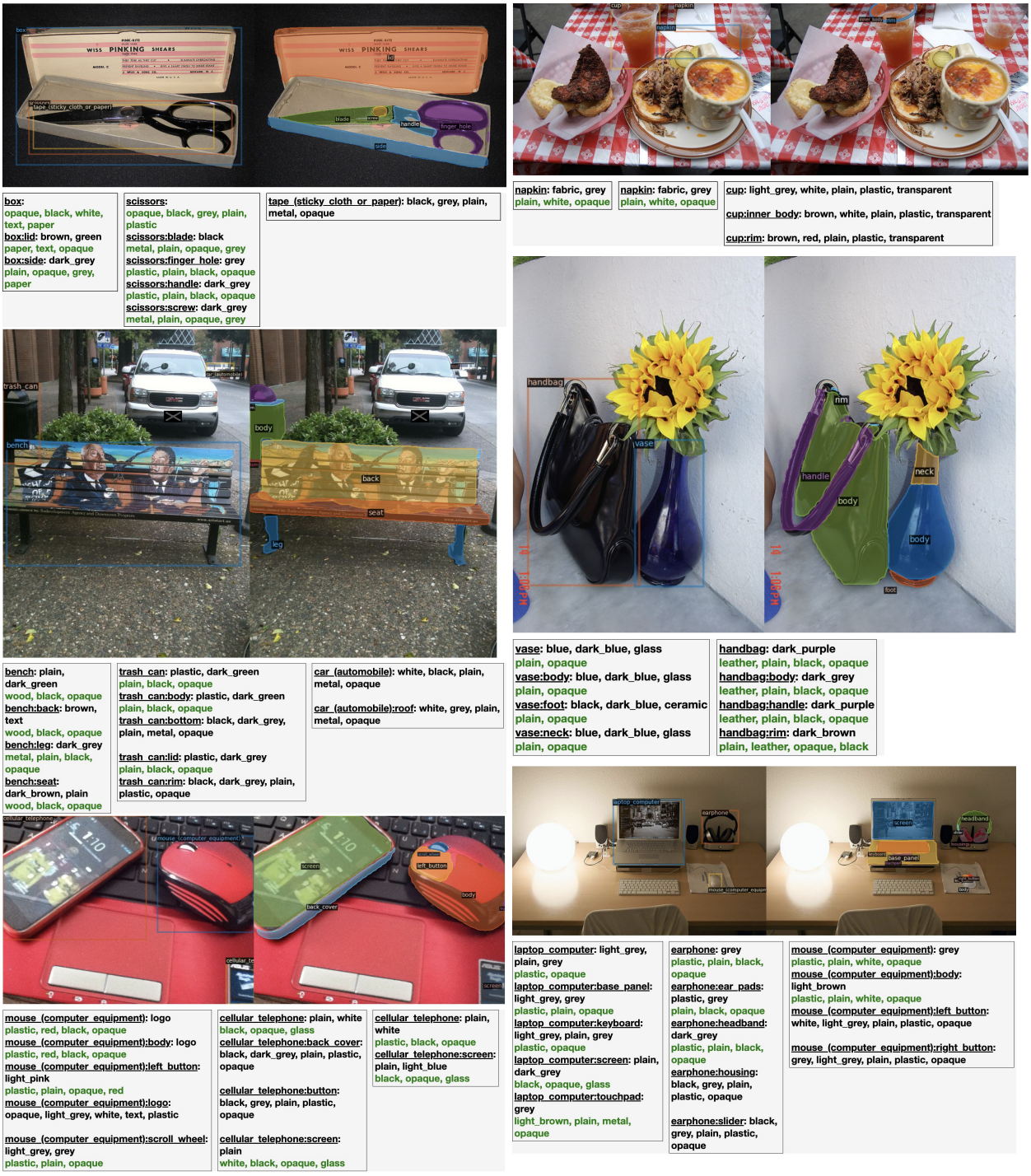

Figure 10. Part segmentation and attribute prediction examples from a Vit-L model trained on PACO-LVIS and PACO-EGO4D. Each image contains predicted object bounding boxes for the 3 highest scoring objects on the left and predicted part masks which overlap with these objects on the right. The corresponding object and part attribute predictions are listed below each image. Attribute predictions in green are contained in ground truth.

References

[1] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the International Conference on Computer Vi-*

sion (ICCV), 2021. 3

[2] Jiankang Deng, Jia Guo, Niannan Xue, and Stefanos Zafeiriou. Arcface: Additive angular margin loss for deep face recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4690–4699, 2019. 9

[3] Agrim Gupta, Piotr Dollar, and Ross Girshick. LVIS: A dataset for large vocabulary instance segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019. 5

[4] 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

[5] Aishwarya Kamath, Mannat Singh, Yann LeCun, Gabriel Synnaeve, Ishan Misra, and Nicolas Carion. Mdetr-modulated detection for end-to-end multi-modal understanding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1780–1790, 2021. 7

[6] Yanghao Li, Hanzi Mao, Ross B. Girshick, and Kaiming He. Exploring plain vision transformer backbones for object detection. *ArXiv*, abs/2203.16527, 2022. 4

[7] 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. 9

[8] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015. 9

[9] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. https://github.com/facebookresearch/detectron2, 2019. 9

[10] Xingyi Zhou, Rohit Girdhar, Armand Joulin, Phillip Krähenbühl, and Ishan Misra. Detecting twenty-thousand classes using image-level supervision. *arXiv preprint arXiv:2201.02605*, 2022. 7