PartDistillation: Learning Parts from Instance Segmentation - *Supplementary*

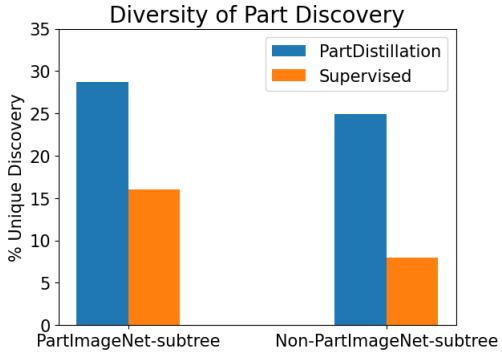


Figure 1. We show the diversity analysis of the discovered plots from both PartDistillation and supervised model trained on PartImageNet dataset. Similar to Fig. 2 of the main paper, we draw the same conclusion that there are larger number of salient discovery, that are unique.

1. Diversity Study

Here we show *diversity study* of the unsupervised part discovery. The purity evaluation in the main paper (Fig. 2) does not consider redundant part discovery. A model can discover 8 different "wheel" of vehicle class from different images, while failing to discover other parts like "window" or "bonnet". Therefore, we measured the total number of *unique parts* from the discovered parts. Specifically, from the 600 balanced set of object classes (inside/outside PartImageNet subtree), manual annotators count the number of unique parts from the parts that has 3 or more consistent parts in a 3×3 collage. In Fig. 1, we show the percentage of *unique* discovery of the supervised model trained on PartImageNet and PartDistillation. From the plot, we can draw two conclusions: (1) PartDistillation discover 2x more unique parts compared to the supervised model, and (2) PartDistillation does not suffer distribution gap like the supervised model. In Fig. 5, 6, 7, 8, 9, 10, 11, we visualize non-cherry-picked images of discovered parts. Note that this type of manual annotation is significantly cheaper than the usual segmentation annotation.

2. Method Details

The main architecture of PartDistillation is based on Mask2Former [4] with SwinL [13] backbone pretrained on ImageNet-21K [7] classification dataset. We explain details of implementation for each stage of training pipeline.

Initial instance segmentation model. PartDistillation begins with a strong instance segmentation model, as noted in Sec. 4.1 of the main paper. The Mask2Former is trained on COCO-Instance Segmentation dataset [12] for 100 epochs, closely following the original paper [4]. This includes the number of queries $N_q = 200$, the hyperparameters for losses ($\lambda_{\text{cls}} = 2, \lambda_{\text{mask}} = 5, \lambda_{\text{dice}} = 5$), etc. Unless otherwise mentioned, we follow the same original setting for other training, both PartDistillation and baselines.

Combining with Detic [18] framework. PartDistillation segments parts based on instance segmentation predictions, as described in Sec. 4.1, Eqn. 1 of the main paper. PartDistillation extend self-supervised part segmentation to over 21K object classes, and getting high quality object instance segmentation is a critical component. Therefore, we adopt Detic [18], a strong open-vocabulary instance segmentation model that segments and classifies object instances directly from the text embedding of the class name. Therefore, we train PartDistillation with the Detic framework in a naive fashion; we keep two separate segmentation models (Detic and Mask2Former) and use the Detic's prediction to provide instance segmentation mask to train PartDistillation. As noted in the experiment section Tab. 5 of the main paper and Tab. 4, the Mask2Former feature resulted the strongest initial part segments to train part proposals. This simple solution worked well for the purpose of our project, and a future direction is to build a unified framework from the base.

Part segments. To learn class-agnostic part proposals, we start by grouping the pixel-level features of our pretrained Mask2Former architecture and get a set of *part segments*. Specifically, we use the backbone feature of the last two levels of features (namely, *res3* and *res4*) interpolated and concatenated, as we experimentally found optimal, in Tab. 1. For every object instance segmentation mask M_i^o, we group the pixel-level features on the mask by K-Means cluster-

ing. As seen in Fig. 6 (left) of the main paper, we choose the optimal number of per-instance cluster $k = 4$. We follow scikit-learn [16]'s K-Means clustering protocol to find a good clustering by repeating 10 times. After clustering, we apply dense-CRF [10] to smooth-out the boundaries. Specifically, the clustering is done on $\times\frac{1}{8}$ image size on the feature map (*res3*-level). Then, the resulting segments are up-sampled to match the image size, and smoothed out with dense-CRF. We set hyperparameters of the dense-CRF: for the Gaussian kernel, $\theta_\gamma = 3, w_2 = 3$, and for the Bilateral kernel, $\theta_\alpha = 20, \theta_\beta = 13, w_1 = 10$. Since we apply on discrete label maps, we set the ground truth probability to be 0.7 (other than θ_α and $p_{\text{gt}} = 0.7$, we follow the default setting). All the process can be done efficiently offline in parallel, as the encoder part of PartDistillation is frozen during training.

Part-proposal learning. With the part segments, we train our class-agnostic query-based part decoder D^p. The training follows the same Hungarian matching and learning process as Mask2Former, where each part segment gets positive binary label ($y = 1$) and the part segmentation mask, $\hat{M}_i^p$. One notable difference is that for point-based losses in Mask2Former, we sample training points *at random* instead of the importance sampling of the original Mask2Former, as the part segments from clustering have noisy decision boundary by nature. Specifically, we sample the same amount of 12544 training points but entirely at random. We keep the same number of queries ($N_q = 200$) as instance queries for the whole image. Similar to the encoder, the decoder layers are also initialized from the COCO instance segmentation model. The only difference here is the number of classes during training ($N_c^o = N_c^p = 1$ for class-agnostic part segmentation) and the training point sampling strategy. We train the model (decoder) for 50k iterations with the batch size of 128 (4 images per GPU), with the learning rate of 0.0001 on AdamW [14] optimizer. We train with LargeScaleJittering [8], random cropping and random flipping augmentation. We resize images such that the longer side is 640. This is our default training setting and kept same for all model training unless otherwise mentioned.

Class-specific part ranking. Once class-agnostic part decoder is trained, we estimate the density of the part segmentation for each object class c^o. Specifically, we accumulate part-level query features f_i^p from the decoder D^p (i.e., the query output features of the part decoder). As mentioned in Sec. 4.3 of the main paper, we have two postprocessing steps to choose more salient part predictions. First, we convert the overlapping part proposals into per-pixel predictions. That is, for each pixel location $x \in M_i^o$, we assign the pixel to $\arg\max_j M_j^p(x) s_j^p$ part query, where $j \leq N^p$.

We define the score map of mask for each query j as

$$M_j^p(x) = P(y(x) = 1 | q_j^p = 1) \tag{1}$$

where $y(x) = 1$ if there is a part at location x and 0 otherwise, and $s_j^p = P(q_j^p = 1)$. This process assigns a (possibly empty) set of pixels to each part query. Let's define such set of pixels as $\tilde{M}_j^p$ for all part queries q_j^p. We can measure the ratio of the assigned part area for each query q_j^p that is matched to an object M_i^o:

$$r_j = \frac{|\tilde{M}_j^p|}{|M_i^o|} \tag{2}$$

Then, we apply (a) an area filter $\tau_r = 0.05$ and (b) a score filter $\tau_s = 0.3$:

$$\tilde{q}_j^p = \begin{cases} q_j^p & \text{if } r_j > \tau_r \wedge s_j^p > \tau_s \\ \emptyset & \text{otherwise} \end{cases} \tag{3}$$

where $\emptyset$ queries are ignored. This cleans out significant amount of noisy part proposals. Next, we accumulate the resulting part queries across dataset for each class c^o, and use the density estimation with unit Gaussian prior, similar to Snell *et al*. [17]. This is equivalent to applying K-Means clustering and using the relative distance from a point to each cluster centroids to model the probability. We use this score as our final ranking function for part classification:

$$r_k(f_j^p | f_i^o) = \frac{\exp(-\|D^p(f_j^p, f_i^o) - \mu_k^o\|_2)}{\sum_{l=1}^{N_k} \exp(-\underbrace{\|D^p(f_j^p, f_i^o) - \mu_l^o\|_2}_{\triangleq D^l(f_j^p, f_i^o)})} \tag{4}$$

Here we set the number of modes (clusters) $N_c^p = 8$ for all object classes.

Final self-training. With the part ranking function, we now have a model that has (a) per-object-class part classifier and (b) corresponding part segmentation. We use this model as an initial model for self-training by treating each part mask $\tilde{M}_j^p$ and the associated label $y_j = \arg\max_k r_k(f_j^p | f_i^o)$. Then, for all object mask M_i^o in the dataset, we have the pseudo-labels

$$Y_{\text{pseudo}} = \{(\tilde{M}_j^p, y_j) : \tilde{q}_j^p \neq \emptyset\} \tag{5}$$

We train $N_c^o = 21000$ classifiers for each of the object classes in ImageNet-21K, and for each classifier we have $N_c^p = 8$ part classes. During training, we only apply gradient to the classifier with the particular object-class, and the scoring (softmax) is only among the N_c^p part classes.

$$r_k(f_j^p | f_i^o) = \frac{\exp((W_k^{\hat{o}})^\top D^p(f_j^p, f_i^o) + b_k^{\hat{o}})}{\sum_{l=1}^{N_c^p} \exp((W_l^{\hat{o}})^\top D^p(f_j^p, f_i^o) + b_l^{\hat{o}})} \tag{6}$$

where $\hat{o}$ is the corresponding object class that f_i^o is from. During inference, we use the prediction's object class prediction to get part scores. Final self-training completes PartDistillation pipeline. For final self-training, we train the model for 120k iterations, with the learning rate of 0.0001 on AdamW optimizer.

Few-shot Training. For few-shot training, we trained all models (supervised and PartDistillation) on AdamW optimizer with the learning rate of 0.00001. We adaptively chose the number of iterations to train to avoid overfitting. For every 1% of data, we trained for 1k iterations. We process images the same way as other training (image size and augmentation).

3. Baseline Details

Supervised models. For all our supervised baselines, we initialized with the same architectures and pretrained weights, which is Mask2Former with SwinL backbone, pretrained on COCO Instance Segmentation for 100 epochs. For each dataset, we train two versions of supervised models where first we only train class-agnostic part segmentation model and second train a class-specific part model. The former model gives the most apples-to-apples comparison to our first-stage model after proposal learning, and we observe training for class-agnostic part segmentation performed better on AR@200 metric compared to class-specific counterpart, as seen in Tab. 2. For training, we follow the default training recipe of the original Mask2Former. We use AdamW optimizer with learning rate 0.0001 and train for 20k iterations on 128 images per batch (4 images per GPU). Similar to PartDistillation, we apply random crop, random scale and random flip augmentations. The images are processed the same way as PartDistillation training.

One-stage self-training. One-stage self-training baseline is the baseline of our own. We test the effectiveness of the two-stage nature of self-training in PartDistillation by applying the most straight-forward self-training recipe similar to MDC [5]. More specifically, instead of getting "per-instance part segment" by clustering as described in Sec. 4.2 of the main paper, we accumulate pixel-level features f_i^o of all instances M_i^o (features within the object masks) with the same object class across dataset and apply clustering. The resulting cluster assignment directly becomes segmentation pseudo-labels with part cluster IDs. This is equivalent to *measuring the density of the pixel-level features as per-pixel part predictions*. We apply K-Means clustering on the pixel-level features with $K = 8$, and self-train with the obtained labels with the same setting as PartDistillation. We train this baseline for 120k iterations with the learning rate of 0.0001 on AdamW optimizer.

Porting Choudhury et al. [6] to larger architectures. Comparing PartDistillation with prior works such as Choudhury et al. [6] is impossible, since these methods work well when trained on a single object class. Regardless, we compare with existing works in Tab. 2 of the main paper. To show that the superiority of PartDistillation is not merely due to the advancement of model architecture and pretrain weights, we train Choudhury et al. closely following the initial codebase but with comparable model architecture. Training this for multi-object-class setting failed to learn any meaningful predictions. For single-object-class setting, training with Mask2Former framework, same setting as our model, struggled converging despite highly rigorous hyperparameter tuning. Following the original DeepLab-like framework [3] with losses on pixel-level features worked reasonably well, as reported in Tab. 2 of the main paper.

We have two versions of Choudhury et al., one for quantitative evaluation in Tab. 2 of the main paper and for manual evaluation in Fig. 4 of the main paper. For both versions, we have the following setting. Similar to the original paper, we apply color jittering and random TPS augmentation [2]. The DeepLab-like framework is composed of backbone layers and prediction layers. We initialized the same weight for SwinL backbone used in our model (COCO-pretrained weights). We trained all single-object-class models with learning rate $\eta = 0.00001$, AdamW optimizer for 30k iterations. We trained with batch size of 16 (2 images per GPU) as the training this model takes large memory. For Tab. 2, we use the original hyperparameters: $\lambda_{vc} = 30, \lambda_{eq} = 5.7 \times 10^3, \lambda_{sc} = 5, \lambda_{ct} = 2.3 \times 10^3$ for visual consistency loss, equivariance loss, semantic consistency loss, and contrastive loss, respectively. T number of parts $K = 4$, with the image size 256, and other settings closely follow the original paper. For the manual evaluation, we rigorously tuned the hyperparameters on PartImageNet [9] dataset, and we use $\lambda_{vc} = 1, \lambda_{eq} = 1.0 \times 10^3, \lambda_{sc} = 100, \lambda_{ct} = 1.0 \times 10^4$. Similar to PartDistillation training, images are resized so that the larger side is 640 with the number of parts $K = 8$.

4. Experiments

Here we report various results and experiment details that were omitted from the main paper due to space limit.

Evaluation metrics. We provide more details about the mean Intersection over Union (mIOU) metric in unsupervised setting. Measuring the quality of class-specific prediction is hard. A common practice in the literature is by mapping the cluster IDs to ground truth labels, either by Hungarian matching [11] or majority voting. Hungarian matching is useful when the number of labels is known in advance. Since we do not know the number of parts for each of the 21K object classes, we use majority vote to map

the cluster IDs to labels. Majority vote favors classes with large quantity, but all our metrics are averaged over classes (balanced over classes). After mapping to the ground truth labels, we evaluate the mapped predictions with the standard mIOU metric.

Zero-shot evaluation. Here we describe details of how zero-shot evaluation is done in Tab. 3 of the main paper. AR@200 is localization-only metric, and we use the standard evaluation protocol. For mIOU metric for all methods, evaluating on a new (target) dataset requires remapping the existing labels. This can be done in two ways: (1) clustering the predictions *explicitly* on the target images with labels and perform majority-voting-based mapping, or (2) using existing prediction labels and perform majority-voting-based mapping from confusion matrix. We name them mIOU-`clustered` and mIOU-`mapped`, respectively. The latter (2) is more straight-forward way to evaluate. However, when the source and target datasets share many common object classes but the granularity of parts differ a lot, mapping directly from confusion matrix can be misleading; models may be able to differentiate "arms" and "torso" but may produce a single "body" label. Therefore, we take the former (1) way and show that all methods (especially the supervised baselines) benefit from it, in Tab. 3.

Manual evaluation. Here we provide more details of manual evaluation. We first get 10K object classes from 21K original object classes by removing "non-object" or "object without parts" categories. Specifically, we remove all subtrees of the following classes: `food`, `foodstuff`, `room`, `herb`, `tree`, `fungus`, `event`, `act`, `area`, `grass`, `geological formation`, `sauce`, `woody plant`, `structure`, `abstraction`, `casting`, `plant organ`, `mixed drink`, `plant part`, `body part`, `part`, `region`, `mechanism`, `fabric`, `material`, `vine`, `substance`, `beverage`, `attribute`, `natural phenomenon`, `meal`, `cosmetic`, `weed`. Then, we randomly sample 10K classes from the remaining ImageNet classes. The 100, 500, 1K, 5K, 10K classes are sampled from this 10K classes at random. The 300/300 classes inside/outside subtrees of PartImageNet [9] are sampled at random from WordNet [15] hierarchy using NLTK library [1]. Since PartDistillation as well as all the baselines are initialized with a strong instance segmentation model, these models tend to produce instance segmentation when it sees unknown objects. We remove such predictions for all methods by filtering out part predictions that has IoU larger than 50% of the object mask.

Supervised baselines on source dataset. We show the performance of the supervised baselines on the dataset that they are trained. In Tab. 2, we report the class-agnostic models for AR@200 and class-specific models for mIOU. For AR@200, we additionally show the class-specific part models' performance in gray. For mIOU, we report both mIOU-`clustered` and mIOU-`mapped`.

res3	res4	res5	AR@10
✓			10.8
	✓		23.4
		✓	20.6
✓	✓		**27.6**
	✓	✓	26.1
✓	✓	✓	25.3

Table 1. Detailed ablation for multi-level feature fusion for pixel grouping. Our default setting (Mask2Former on COCO pre-trained SwinL backbone) is used to measure the localization on PartImageNet-`val,test` splits. Gray row is chosen to be our default.

Train	Test	AR@200	mIOU
Pascal Part-`train`	Pascal Part-`val`	47.3 (44.9)	29.0
PartImageNet-`train`	PartImageNet-`val,test`	73.7 (73.4)	48.7

Table 2. Supervised baseline models tested on their source dataset. Please note that for a fair comparison, we trained the baseline models that are compared on AR@200 metric on proposals only, as it consistently had better results across different datasets (with standard baseline numbers inside parenthesis). For class-specific part segmentation, we report mIOU-`clustered`.

Train	Test	mIOU-`clustered`	mIOU-`mapped`
Pascal Part	Pascal Part	29.0	**43.2**
Pascal Part	PartImageNet	**34.9**	16.3
PartImageNet	PartImageNet	48.7	**68.7**
PartImageNet	Pascal Part	**21.8**	10.8

Table 3. We show that directly mapping source labels to target labels with confusion matrix suffers a lot due to the discrepancy in part definition such as granularity, as discussed in Sec. 4. In the main paper, we report mIOU-`clustered` for zero-shot evaluation.

Direct mapping vs clustering for zero-shot. As discussed in Sec. 4, we reported the models performance from *part clustering* results. That is, similar to our *part ranking* process, we cluster the part-level query features with $K = 8$ and use majority voting to evaluate for each target object class. We report both results in Tab. 3, and show that for all datasets, clustering-based evaluation had better number for supervised baselines for zero-shot cases.

Feature	Dataset	Pretrain Task	Frakework	AR@1	AR@10
SwinB	LVIS+COCO	Open Vocab.	Detic	2.7	10.8
SwinB	COCO	Instance Seg.	Mask2Former	6.6	27.0
SwinL	COCO	Instance Seg.	Mask2Former	**6.6**	**27.6**
SwinL	COCO	Panoptic Seg.	Mask2Former	6.6	27.1
SwinL	Cityscapes	Instance Seg.	Mask2Former	5.1	21.7
SwinL	Cityscapes	Semantic Seg.	Mask2Former	5.5	22.5
SwinL	Cityscapes	Panoptic Seg.	Mask2Former	5.3	21.9
SwinL	ADE20K	Instance Seg.	Mask2Former	5.8	24.5
SwinL	ADE20K	Semantic Seg.	Mask2Former	6.0	24.8
SwinL	ADE20K	Panoptic Seg.	Mask2Former	5.9	24.8

Table 4. More ablations on part segmentation by pixel grouping from different architectures, datasets, tasks, and framework. Gray row is chosen to be our default.

Per-pixel Pred.	Area Thres. (τ_r)	Score Thres. (τ_s)	mIOU-`clustered`
			18.5
✓			40.2
	✓		21.3
		✓	40.8
	✓	✓	41.5
✓	✓	✓	**48.0**

Table 5. Ablation results for postprocessing steps in PartDistillation. Gray row is chosen to be our default.

5. Ablation Studies

Initial features. In Tab. 4, we show more ablation results for getting part segments by pixel grouping. Together with the results from Tab. 5 of the main paper, the choice of backbone (swinL), pretrain task (COCO), and the framework (Mask2Former) are the most critical factors.

Postprocesses. In our part ranking step, we estimate the density of the clusters of the part proposals for each object class. We apply several postprocessing steps where (1) convert the proposals unique per-pixel, (2) have area threshold, and (3) have a confidence threshold. In Tab. 5, we show that each of these steps are essential.

6. Visualizations

For all visualization, it is best viewed in **color** and **zoomed-in**.

Part segments. In Fig. 2, we visualize the pixel grouping quality for the initial part segmentation. We show that the model architecture, pretrain task, and framework matter a lot. The visualization was generated from `val,test` splits of PartImageNet dataset.

Final results. In Fig. 3, we visualize some predictions of the baselines (supervised, "one-stage self-training", and PartDistillation) on the `val` split of Pascal Parts dataset. Supervised models suffer the domain gap (incomplete / missing segmentation).

Discovered parts. In Fig. 4, we show some more example collages of the part discovery result that annotators see, generated by a fully-trained PartDistillation model. Each collage has 9 randomly sampled images with the same discovered parts highlighted in red. In Fig. 5, 6, 7, 8, 9, 10, 11, we visualize non-cherry-picked images of discovered parts. We randomly sample object classes from the 600 balanced ImageNet-21K classes that are inside and outside PartImageNet subtree. We visualize 3 random images at each row for each discovered part that is annotated as "consistent" by the annotators. For all object classes, PartDistillation make 8 part discovery.

References

[1] Steven Bird and Edward Loper. NLTK: The natural language toolkit. In *Proceedings of the ACL Interactive Poster and Demonstration Sessions*, pages 214–217, Barcelona, Spain, July 2004. Association for Computational Linguistics. 4

[2] Fred L. Bookstein. Principal warps: Thin-plate splines and the decomposition of deformations. *IEEE Transactions on pattern analysis and machine intelligence*, 11(6):567–585, 1989. 3

[3] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(4):834–848, 2018. 3

[4] Bowen Cheng, Ishan Misra, Alexander G. Schwing, Alexander Kirillov, and Rohit Girdhar. Masked-attention mask transformer for universal image segmentation. 2022. 1

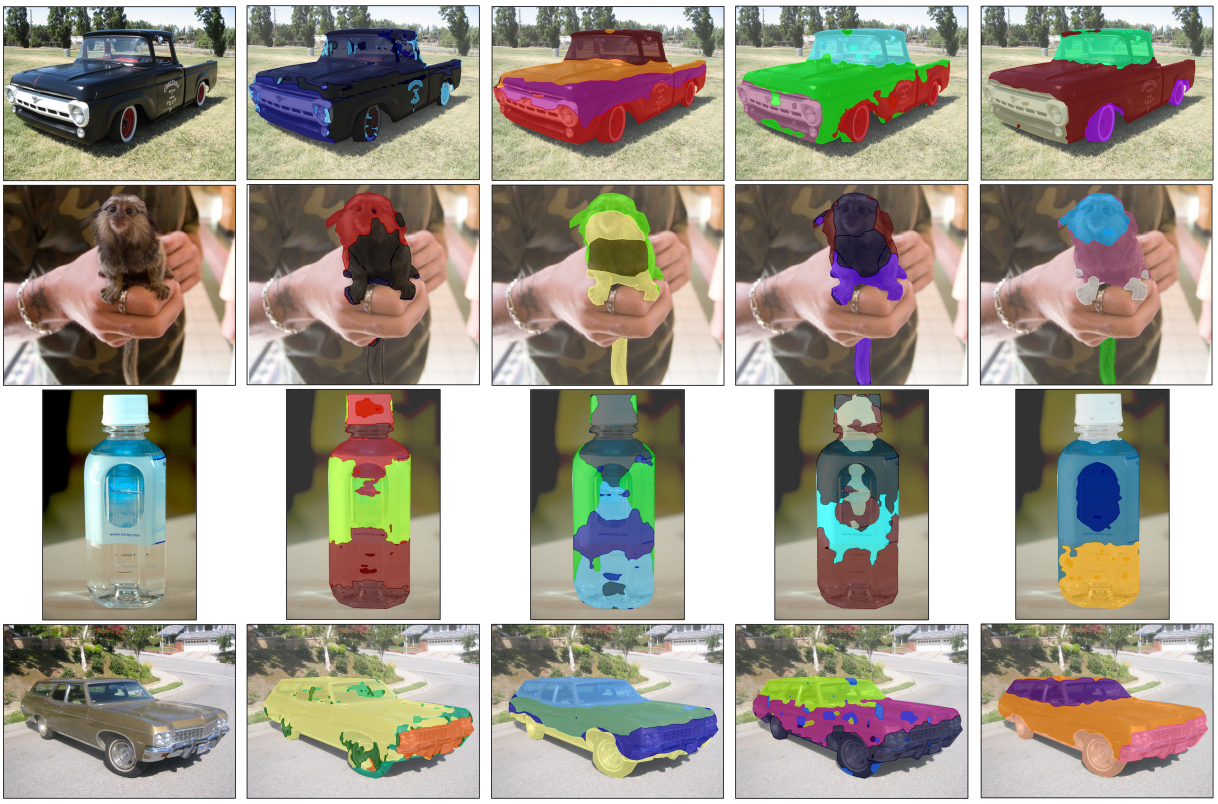
[5] Jang Hyun Cho, Utkarsh Mall, Kavita Bala, and Bharath Hariharan. Picie: Unsupervised semantic segmentation using invariance and equivariance in clustering. In *CVPR*, 2021. 3

[6] Subhabrata Choudhury, Iro Laina, Christian Rupprecht, and Andrea Vedaldi. Unsupervised part discovery from contrastive reconstruction. *Advances in Neural Information Processing Systems*, 34:28104–28118, 2021. 3

[7] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. 1

[8] Golnaz Ghiasi, Yin Cui, Aravind Srinivas, Rui Qian, Tsung-Yi Lin, Ekin D. Cubuk, Quoc V. Le, and Barret Zoph. Simple copy-paste is a strong data augmentation method for instance segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2918–2928, June 2021. 2

[9] Ju He, Shuo Yang, Shaokang Yang, Adam Kortylewski, Xiaoding Yuan, Jie-Neng Chen, Shuai Liu, Cheng Yang, and Alan Yuille. Partimagenet: A large, high-quality dataset of parts. *arXiv preprint arXiv:2112.00933*, 2021. 3, 4



(a) Image **(b)** SwinL-Detic **(c)** ResNet101-M2F-COCO **(d)** SwinL-ImageNet21K **(e)** SwinL-M2F-COCO

Figure 2. Visualization of the pixel grouping results from different backbone, framework, and pretrain tasks. We choose (e) as our default setting.

[10] Philipp Krähenbühl and Vladlen Koltun. Efficient inference in fully connected crfs with gaussian edge potentials. In *NIPS*, 2011. 2

[11] Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955. 3

[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. 1

[13] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021. 1

[14] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 2

[15] George A. Miller. WordNet: A lexical database for English. In *Human Language Technology: Proceedings of a Workshop held at Plainsboro, New Jersey, March 8-11, 1994*, 1994. 4

[16] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011. 2

[17] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. *Advances in neural information processing systems*, 30, 2017. 2

[18] 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. 1

(a) Supervised - Cityscapes Parts **(b)** Supervised - PartImageNet **(c)** One-stage Self-training **(d)** PartDistillation

Figure 3. Visualization of the baselines and PartDistillation predictions on `val`-split of Pascal Part dataset. Supervised models ((a) and (b)) critically suffer from domain gap.

Figure 4. Some example collages that the annotators see. Each collage has 9 randomly selected images and the corresponding part highlighted in red.

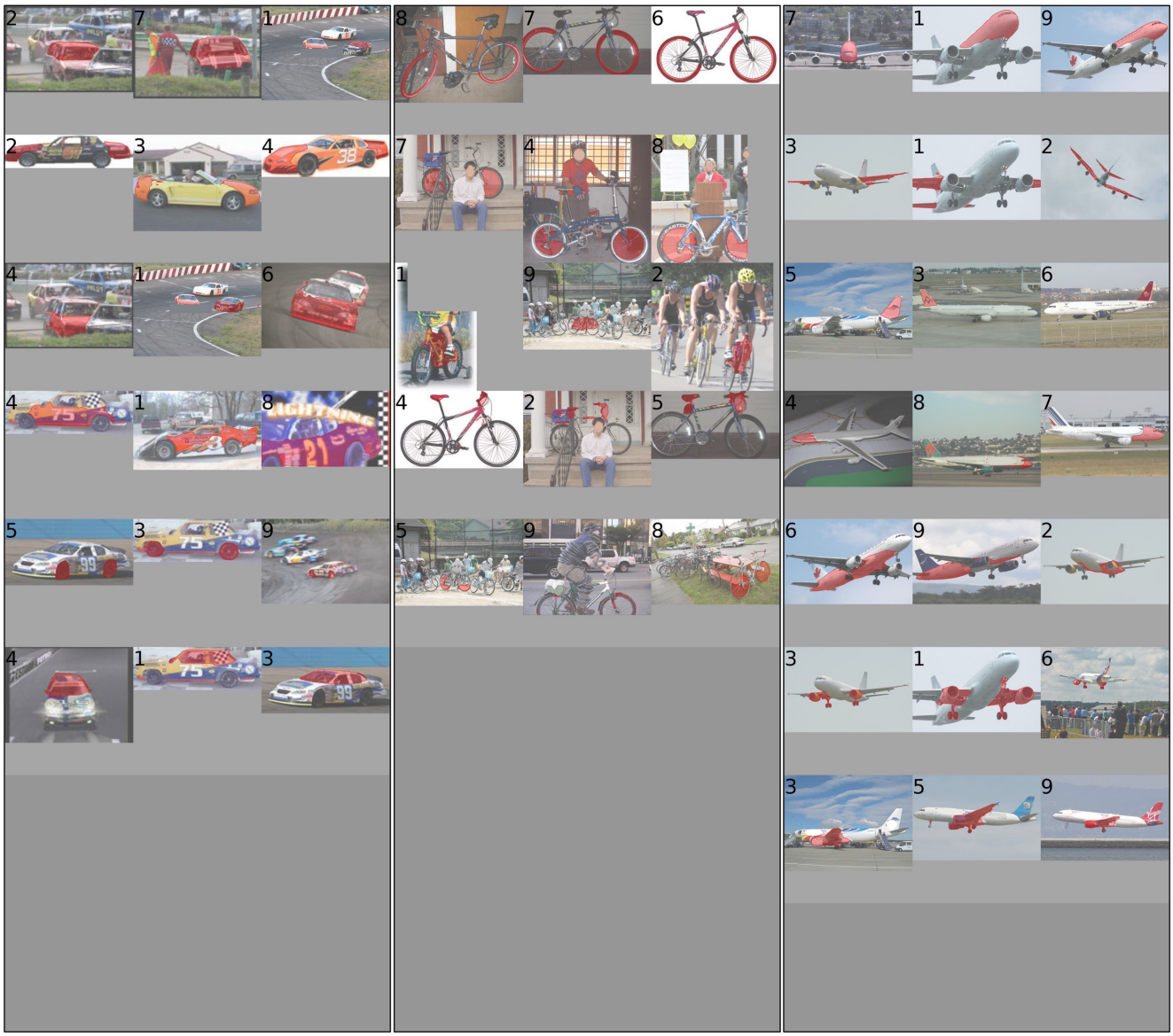


Figure 5. Randomly sampled ImageNet class and the discovered parts. Each row has 3 random images of the same discovered part.

Figure 6. Randomly sampled ImageNet class and the discovered parts. Each row has 3 random images of the same discovered part.

Figure 7. Randomly sampled ImageNet class and the discovered parts. Each row has 3 random images of the same discovered part.

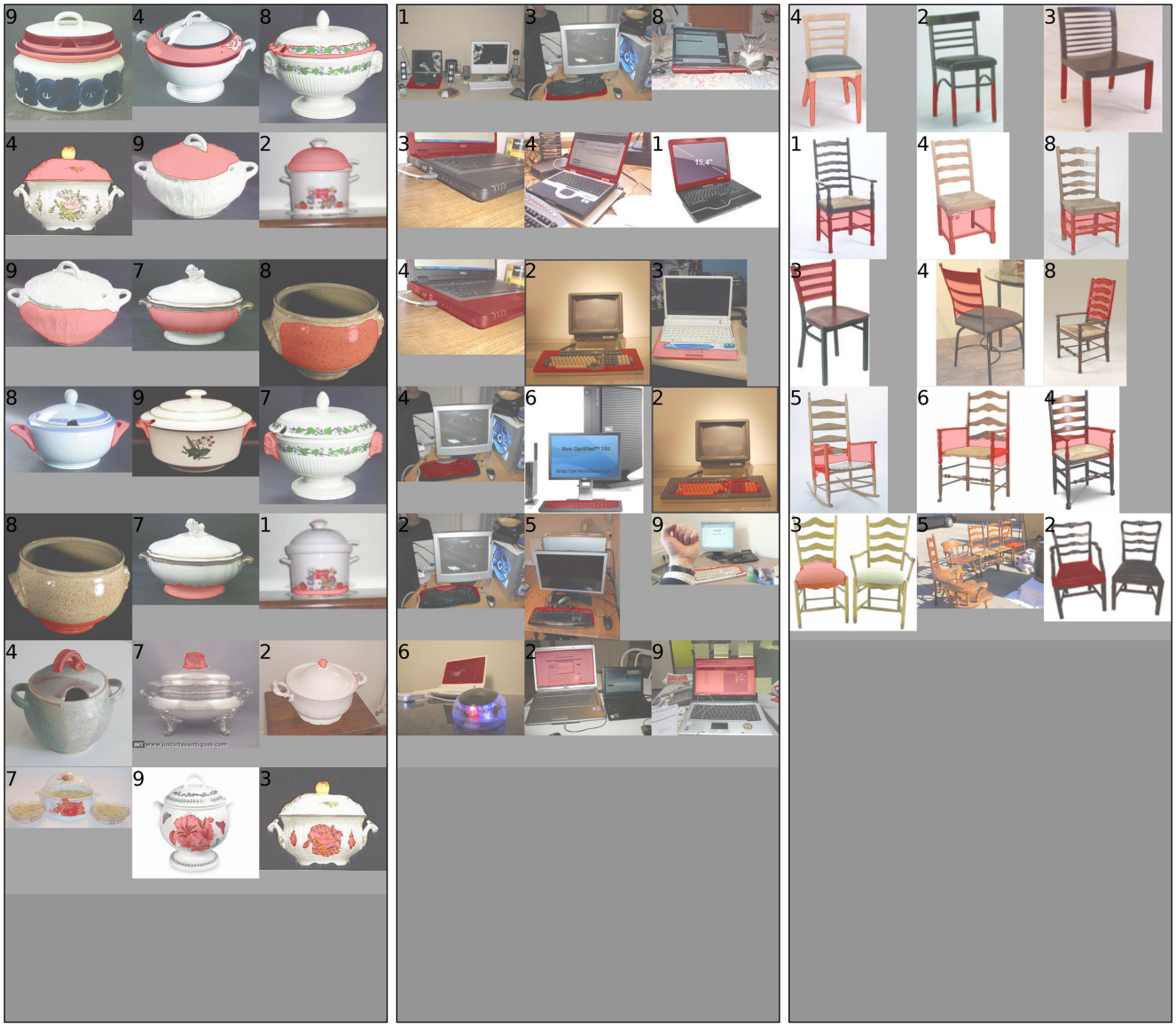


Figure 8. Randomly sampled ImageNet class and the discovered parts. Each row has 3 random images of the same discovered part.

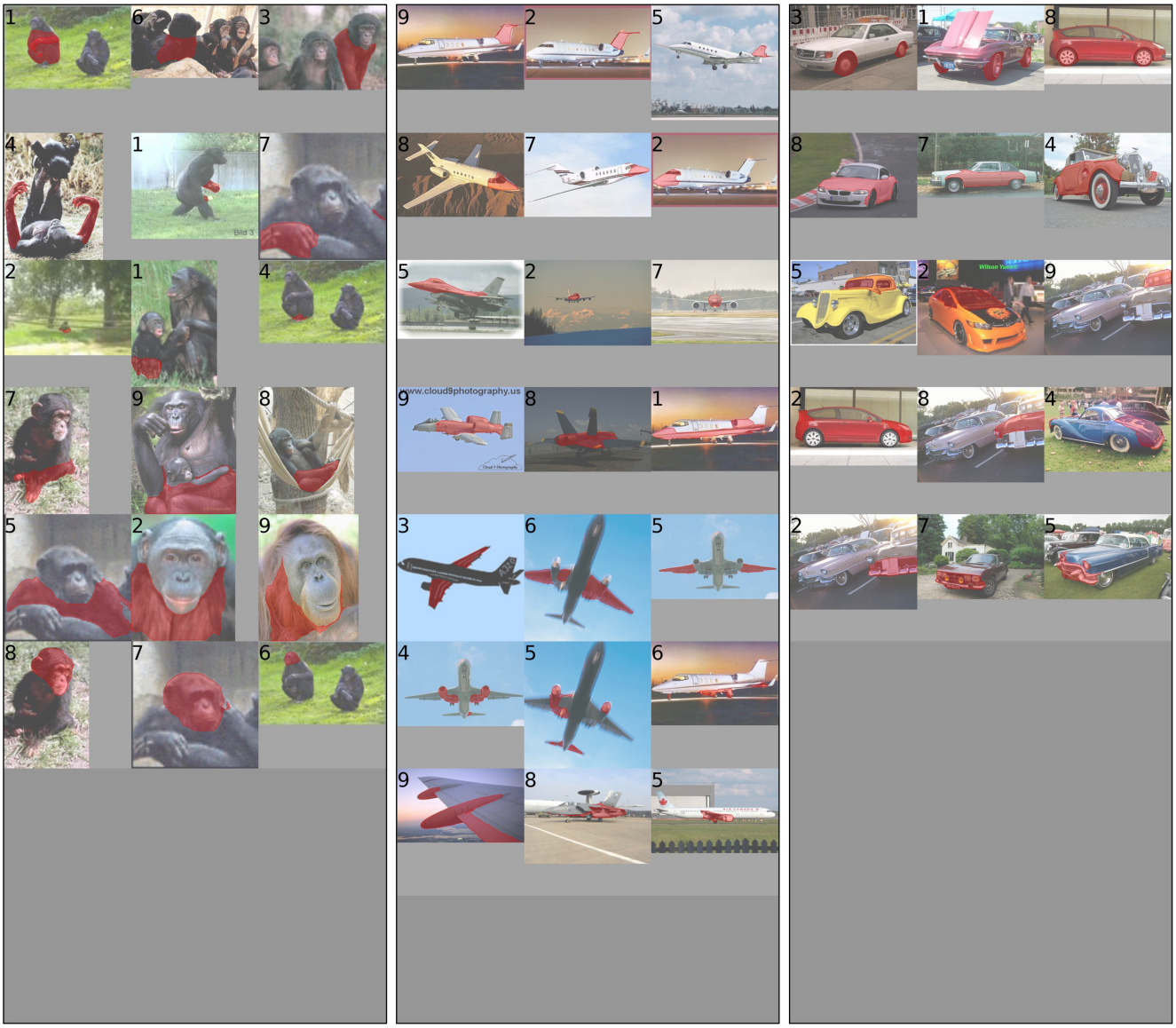


Figure 9. Randomly sampled ImageNet class and the discovered parts. Each row has 3 random images of the same discovered part.

Figure 10. Randomly sampled ImageNet class and the discovered parts. Each row has 3 random images of the same discovered part.

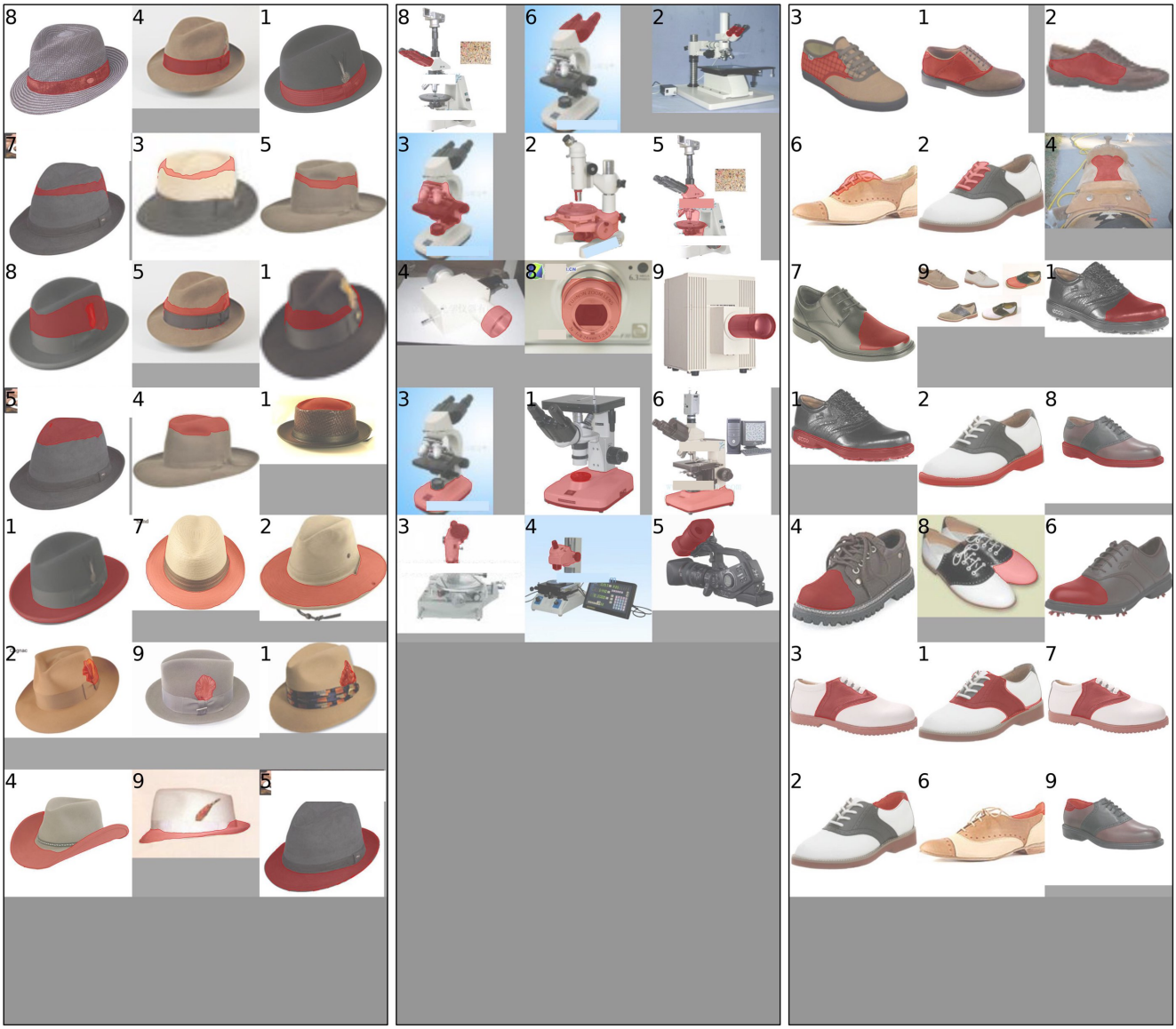


Figure 11. Randomly sampled ImageNet class and the discovered parts. Each row has 3 random images of the same discovered part.

Figure 12. Randomly sampled ImageNet class and the discovered parts. Each row has 3 random images of the same discovered part.