Object discovery and representation networks: supplementary material

A Appendix

A.1 Implementation: data pre-processing

Self-supervised pretraining. Each image is randomly augmented twice, resulting in two views $\boldsymbol{v}^1$ and $\boldsymbol{v}^2$. The augmentations are constructed as compositions of the following operations, each applied with a given probability:

1. random cropping: a random patch of the image is selected, whose area is uniformly sampled in $[0.08 \cdot \mathcal{A}, \mathcal{A}]$, where $\mathcal{A}$ is the area of the original image, and whose aspect ratio is logarithmically sampled in $[3/4, 4/3]$. The patch is then resized to 224 $\times$224 pixels using bicubic interpolation;
2. horizontal flipping;
3. color jittering: the brightness, contrast, saturation and hue are shifted by a uniformly distributed offset;
4. color dropping: the RGB image is replaced by its grey-scale values;
5. gaussian blurring with a 23$\times$23 square kernel and a standard deviation uniformly sampled from $[0.1, 2.0]$;
6. solarization: a point-wise color transformation $x \mapsto x \cdot \mathbb{1}_{x<0.5} + (1-x) \cdot \mathbb{1}_{x \geq 0.5}$ with pixels x in $[0, 1]$.

The augmented images $\boldsymbol{v}^1$ and $\boldsymbol{v}^2$ result from augmentations sampled from distributions $\mathcal{T}_1$ and $\mathcal{T}_2$ respectively. These distributions apply the primitives described above with different probabilities, and different magnitudes. The following table specifies these parameters for the BYOL framework [4], which we adopt without modification.

Parameter	$\mathcal{T}_1$	$\mathcal{T}_2$
Random crop probability	1.0	
Flip probability	0.5	
Color jittering probability	0.8	
Color dropping probability	0.2	
Brightness adjustment max	0.4	
Contrast adjustment max	0.4	
Saturation adjustment max	0.2	
Hue adjustment max	0.1	
Gaussian blurring probability	1.0	0.1
Solarization probability	0.0	0.2

The "spanning view" $\boldsymbol{v}^0$ is chosen as the smallest crop spanning the spatial extent of $\boldsymbol{v}^1$ and $\boldsymbol{v}^2$. When training ResNet or Swin, we resize $\boldsymbol{v}^0$ to 448×448 resolution. When training vision transformers, we resize it to 224×224 resolution.

Transfer to COCO. Resolutions used for Mask-RCNN and FCOS* are 1024×1024 and 800×1024, respectively. During testing, an image is resized by a factor s while preserving the aspect ratio, such that it is tightly contained inside the target resolution, and then padded. When fine-tuning, the image is rescaled by a factor of $u \cdot s$ where u is uniformly sampled in $[0.8, 1.25]$, and is then cropped or padded to the target resolution.

Transfer to PASCAL. During training, images are randomly flipped and scaled by a factor in $[0.5, 2.0]$. Training and testing are performed with 513×513-resolution images.

Transfer to Cityscapes. During training, images are randomly flipped horizontally and scaled by a factor in $[0.5, 2.0]$. Training is performed on 769×769-resolution images and testing on 1025×2049-resolution images.

A.2 Implementation: details of FCOS*

Here we describe FCOS*, a fully convolutional single-stage object detector based on FCOS [11] and its improvements [3, 12, 15].

At inference time, as in FCOS [11], an image is ingested by a backbone network such as ResNet-50 or Swin transformer, followed by a feature pyramid network [6] which produces dense feature maps at various scales (feature pyramid levels). Each feature map is processed independently by prediction heads with shared weights, producing three sets of dense outputs for each pyramid level: classification logits (cls), quality logits ($qual$) and bounding box parameters. Each location in each output map corresponds to a detection with the regressed bounding box parameters (parametrized as distances to the 4 edges divided by the level's stride), where the score for the detection of class c is computed as $\sqrt{\sigma(cls_c) \times \sigma(qual)}$, and σ is the sigmoid function. Non-maximum suppression is performed to produce the final set of detections.

The network is trained to 1) predict the correct class by minimizing the focal loss [7] as in FCOS [11], 2) regress the bounding box parameters for positive samples by minimizing the GIoU loss [10] as in FCOS [11], and 3) predict correct detections using the quality logits as in [12] by minimizing the binary cross-entropy loss. The positive samples are defined through the assignment of the dense predictions to ground truth boxes via ATSS [15].

For all components and parameters of the method we use the default settings from the respective papers. Slight departures consist of using 1) a lower resolution (800×1024 instead of more standard 800×1333), 2) the cross-replica batch-norm [9] in backbones where applicable, and 3) the T-Head for all three prediction heads while [3] does not use it for the quality branch.

A.3 Implementation: optimization

Self-supervised pretraining. We pretrain ResNet-50 and Swin-Transformers on ImageNet for 1000 epochs using the LARS optimizer [14] with a batch size of 4096 split across 128 Cloud TPU v3 workers. We adopt the optimization details of BYOL, scaling the learning rate linearly with the batch size and decaying it according to a cosine schedule. The base learning rate is 0.2 and the weight decay is $1.5 \cdot 10^{-6}$. When pretraining Swin transformers we additionally use gradient clipping with a maximum norm of 1. The contrastive loss temperature α is 0.1.

We pretrain vision transformers (ViT) on ImageNet for 300 epochs using the Adam optimizer [5] with a batch size of 2048 split across 256 Cloud TPU v3 workers. We use a learning rate of $3 \cdot 10^{-4}$, a weight decay of 0.1, and momentum parameters $\beta_1 = 0.9$ and $\beta_2 = 0.95$.

When training ResNet-50 and Swin-Transformers, we fuse intermediate feature maps into a single high-resolution latent array using Feature Pyramid Networks, and segment these features using their projections $\boldsymbol{z}^0$. When training vision transformers, we use the hidden vectors $\boldsymbol{h}^0$ for segmentation.

Transfer to COCO with Mask-RCNN. We fine-tune with stochastic gradient descent, increasing the learning rate linearly for the first 500 iterations and dropping twice by a factor of 10, after $\frac{2}{3}$ and $\frac{8}{9}$ of the total training time, following [13] . We use a base learning rate of 0.3, a momentum of 0.9, a weight decay of $4 \cdot 10^{-5}$, and a batch size of 64 images split across 16 workers.

Transfer to COCO with FCOS*. The network is trained for 30 epochs with batch size 128 split across 16 workers, with AdamW [8], weight decay 10^{-4} and base learning rate of 10^{-3}. The learning rate rises linearly for $\frac{1}{4}$ of an epoch, and is dropped twice by a factor of 10, after $\frac{2}{3}$ and $\frac{8}{9}$ of the total training time.

Transfer to PASCAL. We fine-tune for 45 epochs with stochastic gradient descent, with a batch size of 16 and weight decay of 10^{-4}. The learning rate is 0.02 and dropped by a factor of 10 at the 70th and 90th percentiles.

Transfer to Cityscapes. We fine-tune for 160 epochs with stochastic gradient descent and a Nesterov momentum of 0.9, using a batch size of 2 and weight decay of 10^{-4}. The initial learning rate is 0.005 and dropped by a factor of 10 at the 70th and 90th percentiles.

A.4 Implementation: evaluating object discovery

We extract the central crop of COCO images and resize them to a target resolution of 1024×1024 for ResNet models, and 448×448 for vision transformers. These images are encoded by the feature extractor, resulting in a 32×32 feature grid for ResNet, 256×256 for ResNet equipped with FPN, and 56×56 for the vision transformer. For ease of comparison with other forms of pretraining, when evaluating ViT and ResNet we use the final hidden layer $\boldsymbol{h}$ (with 768 and 2048 channels, respectively) for unsupervised segmentation. When using the ResNet

equipped with FPN, we use the projections $\boldsymbol{z}$. In all cases, vectors are L^2 normalized before applying k-means clustering.

For each ground-truth segment $\boldsymbol{g}_t$ we compute the overlap with all proposals $\boldsymbol{m}_k$ using their intersection-over-union (IoU), and record the best overlap (BO) by taking the maximum across proposals:

$$\text{IoU}(\boldsymbol{g}_t, \boldsymbol{m}_k) = \frac{\sum_{i,j} \min(\boldsymbol{g}_t, \boldsymbol{m}_k)[i,j]}{\sum_{i,j} \max(\boldsymbol{g}_t, \boldsymbol{m}_k)[i,j]} \tag{1}$$

$$\text{BO}(\boldsymbol{g}_t) = \max_k \text{IoU}(\boldsymbol{g}_t, \boldsymbol{m}_k). \tag{2}$$

For a given image with T ground-truth segments, we obtain the "average best overlap" (ABO) metric [1] by averaging BO across ground-truth segments, and the "object recovery" metric [2] by computing the fraction of "best overlaps" greater than 50%:

$$\text{ABO} = \frac{1}{T} \sum_{t=1}^{T} \text{BO}(\boldsymbol{g}_t) \tag{3}$$

$$\text{OR} = \frac{1}{T} \sum_{t=1}^{T} \mathbb{1}_{\text{BO}(\boldsymbol{g}_t) > 0.5} \tag{4}$$

The ABO^i and OR metrics use instance-based masks as ground-truth segments $\boldsymbol{g}_t$. The ABO^c metric merges instances of the same class into the same ground-truth segment, before applying the same analysis. We average each of these metrics across images.

In Table 4 we perform this analysis using the original COCO masks, without modification. We notice however that some COCO masks are very small, making them barely noticeable perceptually. We verified that these masks do not bias our results by repeating the analysis while excluding these masks. Specifically, we limit the number of masks to 16 per image, and remove masks whose area is smaller than 100 pixels (in a 224×224-resolution image). Table A.1 shows that, after this preprocessing of COCO masks, our results are very similar, and our conclusions are unchanged.

A.5 Implementation: computational complexity

When continuously updating the discovery network, Odin requires an additional forward pass relative to the usual 2-view contrastive setup. This results in a +16% computational overhead when using the same resolution as the other two views, or +67% when doubling the image resolution. The k-means clustering of object-discovery features requires 0.25B FLOPS for standard-resolution images, or 1B FLOPS when doubling the resolution. When using a ResNet-50 backbone, these costs represent a +1.2% and +5% overhead respectively.

When using a sparse set of discrete updates, e.g. every 100 epochs as in Table 6, the unsupervised segmentations can be cached in-between updates. This reduces the computational overhead by a factor of 100×, making it negligible.

Table A.1. Object discovery on COCO, with ground-truth mask filtering: we apply the same object discovery analysis as in Table 4, after filtering COCO masks to remove very small segments (see section A.4)

	ResNet-50			ViT-B		
Pretraining	ABO^i	ABO^c	OR	ABO^i	ABO^c	OR
Random init	29.6	33.6	16.7	29.5	34.2	18.0
Supervised	39.5	42.4	25.6	42.9	49.7	36.8
DINO	40.5	46.0	30.3	42.7	48.3	35.7
Odin	**44.7**	**49.0**	**38.4**	**49.7**	**54.7**	**48.2**
Odin†	**47.2**	**53.9**	**46.2**			

A.6 Results: using object discovery or target networks for transfer learning

Although the object discovery and target networks are designed to provide a learning signal for the online network, we asked whether they could also be used for transfer learning. We therefore fine-tuned these networks for object detection on COCO and semantic segmentation on PASCAL and Cityscapes as before. We found their performance to be similar, but slightly worse than that of the online representation network (Table A.2). This is reasonable: once the online network has converged, its exponential moving average will largely catch up with it. The fact that they slightly underperform justifies our usage of the representation network for transfer.

Table A.2. Transfer learning with different model parameters: fine-tuning on COCO object detection and instance segmentation, and fine-tuning on PASCAL VOC and Cityscapes semantic segmentation are the same as in Table 1 (left) and Table 2.

	COCO		VOC	Citysc.
Model parameters	AP^{bb}	AP^{mk}	mIoU	
target: ξ	42.7	38.3	78.5	76.6
teacher: τ	42.8	**38.4**	78.4	76.9
online: θ	**42.9**	**38.4**	**78.6**	**77.1**

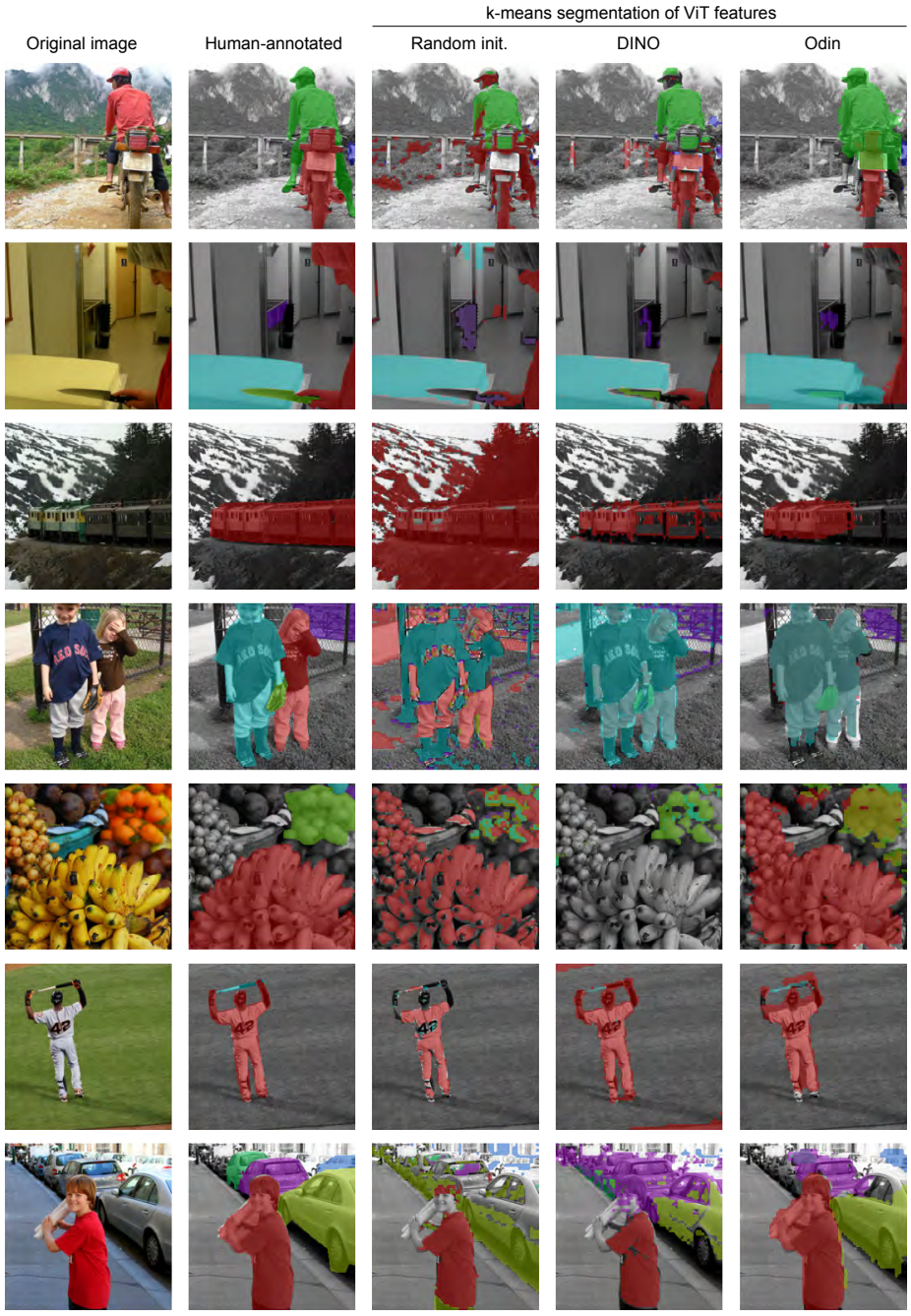

Fig. A.1. Object discovery with vision transformers. 1st column: original image, 2nd: human-annotated COCO segmentations, 3rd, 4th, 5th: segmentations obtained from k-means clustering on randomly intialized, DINO-, and Odin-trained vision transformer features, respectively.

Bibliography

[1] Arbeláez, P., Pont-Tuset, J., Barron, J.T., Marques, F., Malik, J.: Multiscale combinatorial grouping. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 328–335 (2014)

[2] Cho, M., Kwak, S., Schmid, C., Ponce, J.: Unsupervised object discovery and localization in the wild: Part-based matching with bottom-up region proposals. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1201–1210 (2015)

[3] Feng, C., Zhong, Y., Gao, Y., Scott, M.R., Huang, W.: TOOD: Task-aligned one-stage object detection. In: Int. Conf. Comput. Vis. (2021)

[4] Grill, J.B., Strub, F., Altché, F., Tallec, C., Richemond, P., Buchatskaya, E., Doersch, C., Avila Pires, B., Guo, Z., Gheshlaghi Azar, M., et al.: Bootstrap your own latent-a new approach to self-supervised learning. Advances in Neural Information Processing Systems **33** (2020)

[5] Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)

[6] Lin, T.Y., Dollár, P., Girshick, R., He, K., Hariharan, B., Belongie, S.: Feature pyramid networks for object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2117–2125 (2017)

[7] Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollár, P.: Focal loss for dense object detection. In: Int. Conf. Comput. Vis. (2017)

[8] Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. In: Int. Conf. Learn. Represent. (2019)

[9] Peng, C., Xiao, T., Li, Z., Jiang, Y., Zhang, X., Jia, K., Yu, G., Sun, J.: Megdet: A large mini-batch object detector. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 6181–6189 (2018)

[10] Rezatofighi, H., Tsoi, N., Gwak, J., Sadeghian, A., Reid, I., Savarese, S.: Generalized intersection over union. In: IEEE Conf. Comput. Vis. Pattern Recog. (2019)

[11] Tian, Z., Shen, C., Chen, H., He, T.: FCOS: Fully convolutional one-stage object detection. In: Int. Conf. Comput. Vis. (2019)

[12] Wu, S., Li, X., Wang, X.: IoU-aware single-stage object detector for accurate localization. Image and Vision Computing (2020)

[13] Wu, Y., Kirillov, A., Massa, F., Lo, W.Y., Girshick, R.: Detectron2. https://github.com/facebookresearch/detectron2 (2019)

[14] You, Y., Gitman, I., Ginsburg, B.: Large batch training of convolutional networks. arXiv preprint arXiv:1708.03888 (2017)

[15] Zhang, S., Chi, C., Yao, Y., Lei, Z., Li, S.Z.: Bridging the gap between anchor-based and anchor-free detection via adaptive training sample selection. In: IEEE Conf. Comput. Vis. Pattern Recog. (2020)