A Further Method Details and Overview

Recall in Section 3.2, to leverage information across a semantic spectrum, we extract features from throughout ResNet-50, instead of using only features from the final convolutional map as in [22]. Following the notation in Section 3.1 and Section 3.2, we provide a formal description of this procedure here.

Concretely, given a layer l, we define $\mathbf{y}^{[l]} = f^{[l]}(\mathbf{v}) \in \mathbb{R}^{D_F \times H_F \times W_F}$ as the feature map of height H_F, width W_F and feature resolution D_F obtained at the l^{th} layer of encoding via f. Then given a predefined set of layers $\mathcal{L}$, we obtain a final feature map $\mathbf{F} \in \mathbb{R}^{D'_F \times H'_F \times W'_F}$ via simple layer fusion:

$$\mathbf{F} = [S_1(f^{[1]}(\mathbf{v})); S_2(f^{[2]}(\mathbf{v})); \cdots; S_l(f^{[l]}(\mathbf{v}))], \forall l \in \mathcal{L} \tag{5}$$

where S_l is a bilinear upsampling or downsampling layer which projects the feature map at layer l to a desired, common spatial dimensionality $H'_F \times W'_F$ for the concatenation operator $[\cdot\,;\cdot]$. In our work, we let $\mathcal{L} = \{\texttt{res2}, \texttt{res3}, \texttt{res4}\}$.

We also provide pseudo-code to outline our framework. In Algorithm 1, we detail our KMeans bootstrapping process (c.f. Section 3.3). In Algorithm 2, we describe the von Mises-Fisher (vMF) clustering consistency loss (c.f. Section 3.4). Finally, in Algorithm 3, we show how these two components fit into the larger framework. Input parameters and hyperparameters are described in the corresponding sections of the main text.

Algorithm 1 Bootstrap segmentation masks (for a single batch of images $\{\mathbf{I}\}$).

function BootstrapMasks($f_\theta, \{\mathbf{I}\}, \boldsymbol{\ell}, K_{min}, K_{max}$)
  Extract feature map $\mathbf{y} = f_\theta^{[\boldsymbol{\ell}]}(\{\mathbf{I}\})$
  Sample $K \sim Unif(K_{min}, K_{max})$ ▷ Scale-dynamic sampling
  KMeans clustering on $\mathbf{y}$ with K clusters to obtain feature prototypes $\mathbf{P}$
  $\{\mathbf{M}\} \leftarrow$ Label features in $\mathbf{y}$ via Euclidean-closest prototype in $\mathbf{P}$
  Interpolate $\{\mathbf{M}\}$ to spatial dimensions of $\{\mathbf{I}\}$
  return Masks $\{\mathbf{M}\}$
end function

Algorithm 2 Clustering consistency loss (for a single batch of images $\{\mathbf{I}\}$).

function ClustConsistency($f_\theta, f_\xi, \boldsymbol{\ell}, \{\mathbf{I}\}, K, \lambda_{loss}$)
  Augmentations $t \sim \mathcal{T}, t' \sim \mathcal{T}'$
  $\mathbf{v} = t(\{\mathbf{I}\}), \mathbf{v}' = t'(\{\mathbf{I}\})$
  Extract feature maps $\mathbf{y} = f_\theta^{[\boldsymbol{\ell}]}(\mathbf{v}), \mathbf{y}' = f_\xi^{[\boldsymbol{\ell}]}(\mathbf{v}')$
  KMeans clustering on $\mathbf{y}, \mathbf{y}'$ with K clusters to obtain feature prototypes $\mathbf{P}, \mathbf{P}'$
  $\mathcal{L} = \lambda_{loss}\mathcal{L}_{clus}$ ▷ (3)
  Update f_θ to minimize $\mathcal{L}$, f_ξ as EMA of f_θ
end function

Algorithm 3 CYBORGS Training Flow (for a single batch of images $\{\mathbf{I}\}$).

for $i = 1 : W$ **do**
  ClustConsistency$(f_\theta, f_\xi, \boldsymbol{\ell}, \{\mathbf{I}\}, K, 1)$ ▷ vMF warmup period
end for

Masks $\{\mathbf{M}\}$ = BootstrapMasks$(f_\theta, \{\mathbf{I}\}, \boldsymbol{\ell}, K_{min}, K_{max})$ ▷ Section 3.3

for $t = W+1, W+2, \ldots,$ **do**:
  if $t \,\%\, N == 0$ **then**
    Masks $\{\mathbf{M}\}$ = BootstrapMasks$(f_\theta, \{\mathbf{I}\}, \boldsymbol{\ell}, K_{min}, K_{max})$
  end if
  Augmentations $t \sim \mathcal{T}, t' \sim \mathcal{T}'$
  $\mathbf{v} = t(\{\mathbf{I}\}), \mathbf{v}' = t'(\{\mathbf{I}\})$
  Extract object-level representations $\mathbf{h_m}, \mathbf{h'_{m'}}$ using $\{\mathbf{M}\}$ ▷ (1)
  $\mathcal{L} = \mathcal{L}_{mask}$ ▷ (2)
  Update f_θ to minimize $\mathcal{L}$, f_ξ as EMA of f_θ
  if $t \,\%\, M == 0$ **then**
    ClustConsistency$(f_\theta, f_\xi, \boldsymbol{\ell}, \{\mathbf{I}\}, K, \lambda_{vMF})$
  end if
end for

B Experimental Details

Linear transfer. We perform linear classification on PASCAL VOC07 and semi-supervised linear transfer on ImageNet-1k, akin to [52]. For VOC07, we extract a *frozen* feature map from the penultimate layer of our ResNet-50 pretrained backbone, before downsampling to a 2×2 spatial grid via adaptive average pooling. We flatten and $\ell 2$-normalize the features, yielding $\mathbb{R}^{8192}$ feature vectors. We then train per-class SVMs on the trainval split, performing a 3-fold cross validation on costs $C \in \{0.01, 0.1, 1.0, 10.0\}$, before evaluating mAP on the test split.

For ImageNet-1k, we follow settings from [52] and finetune our backbone on the 1% and 10% training subset splits as released in [4]. We report the top-1 and top-5 accuracy on the official val split.

PASCAL VOC object detection. For object detection, we initialize a ResNet-50-C4 backbone with pretrained weights from CYBORGS, before inserting into a Faster RCNN architecture. We then finetune with the trainval07+12 split from PASCAL VOC, before reporting all metrics on the test2007 split via the Detectron2 API [49]. We use all default training settings from [42] for standardized comparison.

COCO instance segmentation. For segmentation, we initialize a ResNet-50-FPN backbone with pretrained weights from CYBORGS, before inserting into a Mask RCNN architecture. We then finetune with the train2017 split from COCO, before reporting all metrics on the val2017 split via the Detectron2 API [49]. Again, we use all default training settings from [42] for standardized comparison, except we follow a 1x training schedule with learning rate decay scheduled accordingly.

	Method	Pretraining Data	mIoU
1)	RANDOM INIT.	–	63.5
2)	SUPERVISED	Supervised, ImageNet	73.7
3)	SIMCLR [9]	ImageNet	73.1
4)	BYOL [17]	ImageNet	71.6
5)	MOCO-V2 [11]	ImageNet	74.5
6)	DENSECL [46]	ImageNet	75.7
7)	DETCO [51]	ImageNet	**76.5**
8)	MOCO-V2 [11]	COCO	73.8
9)	DENSECL [46]	COCO	75.6
10)	BYOL [17]	COCO	72.2
11)	ORL [52]	COCO	72.7
12)	CYBORGS(ours)	COCO	**75.9**

Table C.1: Transfer Learning on CityScapes, ResNet-50 backbones.

Method	Pretraining Data	LVIS, 1x schedule					
		AP^{bb}	AP^{bb}_{50}	AP^{bb}_{75}	AP^{mk}	AP^{mk}_{50}	AP^{mk}_{75}
1) SUPERVISED	–	20.4	32.9	21.7	19.4	30.6	20.5
2) SOCO [47]	ImageNet	**26.3**	**41.2**	**27.8**	**25.0**	**38.5**	**26.8**
3) DENSECL [46]	COCO	20.0	32.3	20.9	19.7	31.0	20.8
4) BYOL [17]	COCO	19.8	32.6	20.8	19.4	30.5	20.5
5) ORL [52]	COCO	20.5	33.5	21.5	20.1	31.5	21.4
6) CYBORGS(ours)	COCO	**23.9**	**38.3**	**25.2**	**23.4**	**36.2**	**24.7**

Table C.2: Transfer Learning on LVIS, ResNet-50 backbones.

C Further Semantic Segmentation Results

CityScapes semantic segmentation We further evaluate the segmentation prowess of CYBORGS representations by transferring to out-of-distribution data. We choose the real-world urban driving scenes CityScapes dataset, and follow the finetuning protocol of [46], training a FCN model on the `train_fine` split (2975 images) for 40k iterations before testing on the `val` split. We demonstrate strong results even when compared to state-of-the-art ImageNet-pretrained SSL methods, suggesting that our bootstrapping of segmentation masks can lead to representations which sensibly adapt to similar segmentation tasks under different data distributions. In particular, we outperform (+0.3 mIoU) DENSECL [46], which is a contrastive learning method specifically designed for strong downstream performance on dense prediction tasks, including object detection and semantic segmentation.

LVIS long-tail instance segmentation. An emerging property of self-supervised representations is their ability to generalize to the semantics of unseen, diverse object distributions [15, 43, 48]. By assessing transfer performance of CYBORGS on the LVIS long-tail instance segmentation benchmark, with 1203 object categories across ~164k images [18], we hope to verify a similar property in our representations.

We follow the standard finetuning protocol in [42], adjusting parameters for a 1x schedule. In comparison to other SSL methods pretrained on COCO, we offer a significant improvement of +3.4 AP^{bb} and +3.3 AP^{mk}.