Supplementary Material of "Beyond the Encoder: Joint Encoder-Decoder Contrastive Pre-Training Improves Dense Prediction"

1. Decoder architecures

Fully Convolutional Network: To match previous evaluation with mmseg framework, we implement a fully convolutional Network (FCN) as a two 3×3 convolutional block of 256 channels (dilation set to 6) with batch normalization and ReLU activations.

Feature Pyramid Network: Feature Pyramid Network (FPN) C5 implementation is customized from the Detectron2 library [22]. The architecture has four lateral convolutional layers. Each layer uses a 1×1 convolution to reduce the ResNet-50 outputs from 2048, 1024, 512, and 256 channels down to 256 channels. We then sum these lateral outputs in a bottom-up manner and pass the combined output through a 3×3 convolution at each decoder level to produce a 256-channel decoder output.

For pre-training with DeCon-ML under deep supervision, we apply a 3×3 convolution at each of the four decoder levels. The resulting outputs are then passed to separate auxiliary layers, yielding four distinct decoder losses. Note that, deep supervision loss is only calculated in FPN scenario where lateral connections are present.

2. Implementation details

DenseCL pre-training: For DenseCL, the two encoders, along with their respective auxiliary layers—i.e., the global and dense projection heads—are kept the same as well as the independent dictionaries for global and dense losses. Additionally, we introduce two decoders, which take their respective encoders' outputs as inputs. We also replicate the encoder's dense and global projection heads and dictionaries separately for the two decoders. As for SlotCon, the input dimension of the decoder projectors is 256 instead of 2048 for the encoder. The size of the encoders and decoders dictionaries size is set to 16384. Other hyper parameters follow the official implementation. We follow DenseCL by updating the first branch of the architecture through Backpropagation, while the second branch is updated using an EMA of the first one.

PixPro pre-training: For PixPro, the two encoders, along with their respective auxiliary layers—i.e., the projectors and pixel to propagation modules—are kept the same. Additionally, we introduce two decoders, which take their respective encoders' outputs as inputs. We also replicate the encoders' projection heads and propagation module for the decoders. As for SlotCon, the input dimension of the decoder projectors is 256 instead of 2048 for the encoder. We follow PixPro by updating the first branch of the architecture through backpropagation, while the second branch is updated using an EMA of the first one. Hyper parameters follow the official implementation.

REFUGE and ISIC pre-training: We pre-train SlotCon and a DeCon-SL adaptation of SlotCon with an FCN decoder on two small-scale datasets: ISIC 2017 [4] and REFUGE [13]. More details on these datasets are provided in the "Semantic segmentation on out-of-domain datasets" Section. For ISIC dataset, we pre-train for 800 epochs with a batch size of 256 and a base learning rate of 1.0 linearly scaled with the batch size. For REFUGE, we pre-train for 2400 epochs with a batch size of 192 and the same learning rate as for ISIC. These pre-trainings are performed with one NVIDIA A6000 GPU with 48GB of memory.

Object detection and instance segmentation: We performed object detection and instance segmentation on COCO 2017 dataset [12]. We fine-tune a Mask R-CNN with FPN backbone for 90000 iterations on the COCO *train2017* dataset and evaluate on COCO *val2017* split following SlotCon [21]. We initialize the network with a pre-trained ResNet-50 or a ConvNeXt-Small encoder along with a randomly initialized decoder, and fine-tune end-to-end in all cases with a batch size of 16. The base learning rate is set to 0.02 for ResNet-50 and to 0.0002 for the ConvNeXt-Small encoder. The ROI mask head uses four convolution layers and the ROI box head uses two fully connected layers. All the aforementioned experiments are performed using one NVIDIA A100 GPU with 80GB memory.

In-domain semantic segmentation: We performed semantic segmentation on Pascal VOC [6], Cityscapes [5], and ADE20K [28] datasets following SlotCon [21] and PixCon [14]. A ResNet-50 encoder with a two-layer FCN decoder, similar to the FCN decoder used during pre-training, is used. We also fine-tuned a ConvNeXt-Small backbone with a two-layer FCN decoder on Pascal VOC dataset. For ADE20K, two decoder heads are used: a one-layer FCN

auxiliary decoder along with a two-layer FCN decoder following SlotCon [21]. For Pascal VOC, we train on VOC *train_aug2012* set for 30000 iterations and evaluate on VOC *val2012* set. For Cityscapes, we fine tune on the *train_fine* set for 90000 iterations and evaluate on the *val_fine* set. For ADE20k with ResNet-50-FCN, we fine tune on the *training* set for 80000 iterations and evaluate on the *validation* set. In all the cases, batch size was 16. In Pascal VOC tasks with ResNet-50-FCN, the models were optimized using SGD with a learning rate of 0.003, momentum of 0.9, and weight decay of 0.0001 with a step learning rate policy to reduce the learning rate by a factor of 0.1 at 21,000 and 27,000 iterations. For Cityscapes semantic segmentation with ResNet-50-FCN, the models were optimized with an SGD optimizer, a learning rate of 0.01, momentum of 0.9, and weight decay of 0.0001, while the learning rate is reduced by a factor of 0.1 at 63,000 and 81,000 iterations following a step policy. In ADE20K task with ResNet-50 FCN, the models were optimized with an SGD optimizer, a learning rate of 0.01, momentum of 0.9, and weight decay of 0.0005, with a polynomial learning rate decay that reduced the learning rate to 1e-4 over iterations. All the aforementioned experiments are performed using one NVIDIA A100 GPU with 80GB (Cityscapes) or 40GB (Pascal VOC and ADE20K) memory.

To compare performance with ViT-based methods, we have fine-tuned a ConvNeXt-Small backbone (pre-trained on ImageNet-1K for 250 epochs using DeCon-SL framework with FCN decoder) using an UPerNet [23] decoder on ADE20K for 160000 iterations. An FCN auxiliary decoder was also used along with it. The model was optimized with AdamW optimizer, a learning rate of 0.0001, betas set to (0.9, 0.999), and a weight decay of 0.05. The parameter-wise learning rate decay follows a stage-wise scheme with a decay rate of 0.9 over 12 layers. The learning rate schedule consists of two phases: a linear warm-up from a factor of 1e-6 for the first 1500 steps, followed by polynomial decay from step 1500 to 160000, with a minimum learning rate of 0.0.

Semantic segmentation on out-of-domain datasets: To evaluate the pre-trained model's generalizability to out-of-domain semantic segmentation tasks, we fine-tuned a ResNet-50 encoder coupled with an FCN decoder on REFUGE [13] and ISIC 2017 [4] dataset. We also fine-tuned a ResNet-50 encoder with a deeplabv3+ decoder [2] and an FCN auxiliary decoder head on PlantSeg [20]. We evaluate generalizability to out-of-domain datasets across different training set sizes, using 5%, 25%, and 100% of randomly selected samples for the REFUGE and ISIC 2017 tasks. For the PlantSeg task, we further assess generalization by fine-tuning with 10% of randomly selected training samples.

REFUGE dataset contains 1200 retinal images in total representing 3 different classes: optic disk, optic cup, and background, divided into 400 training, 400 validation, and 400 testing samples. We train a ResNet-50-FCN network end-to-end for 80000 iterations with a batch size of 16 on the *training* setusing one NVIDIA A100 GPU (40GB). The learning rate starts at 0.01 and decreases with a polynomial scheduler with a power of 0.9. We evaluate the model on the *test* set using the last iteration checkpoint and report the results.

ISIC 2017 is a skin-lesion segmentation dataset which consists of 2000 training, 150 validation, and 600 testing images. We fine-tune the same ResNet-50-FCN architecture on the *training* set for 24000 iterations on one NVIDIA V100 with 32GB of memory. Hyper parameters are the same as for REFUGE. We evaluate on the *validation* set every 500 iterations, select the best checkpoint and report the results of this checkpoint on the *test* set.

PlantSeg is a large-scale agricultural dataset for plant disease segmentation with 150 classes, comprising 11,458 images (9,163 for training and 2,295 for testing) across 34 plant varieties and 115 disease types. We fine-tune a ResNet-50-DeepLabv3+ [2] using SGD with a learning rate of 0.01, momentum 0.9, and weight decay 0.0005 on a single NVIDIA A100 GPU (40GB). The learning rate follows a polynomial decay schedule (power 0.9, minimum 1e-4) over 160,000 iterations. Evaluation is performed on the *test* set using the final checkpoint.

Object detection on out-of-domain datasets: To evaluate generalization in object detection, we fine-tuned our pre-trained encoders on the PlantDoc [17] and Detecting Diseases [16] datasets, using 10% and 100% of their training samples. The PlantDoc dataset [17] includes 2,569 images spanning 13 plant species and 30 object detection categories, covering both healthy and diseased plants. It contains 8,851 labeled instances, split into 2,328 training and 239 testing images. The Detecting Disease dataset consists of 5,493 leaf images across 13 disease categories, divided into 2,904 training, 1,416 validation, and 1,163 test images, with the test set used for evaluation.

For both of the tasks, we fine-tuned a Faster R-CNN framework with a ResNet-50 FPN backbone. These models are implemented using Detectron2 [22] and fine-tuned end-to-end. The total training schedule consists of 20000 iterations with 0.02 learning rate, learning rate decay steps at 12000 and 16000 iterations, and batch size of 4. We evaluated the model at every 1000 iterations, and reported the result with the best checkpoint.

Panoptic segmentation, keypoint detection and dense pose estimation: For panoptic segmentation and keypoint detection on COCO, we adopt the standard configurations provided in the Detectron2 package. Keypoint detection is performed using a Faster R-CNN configuration, while panoptic segmentation employs a Mask R-CNN configura-

tion. Both tasks are trained on the COCO *train2017* set with a learning rate of 0.02, a batch size of 16, and 90,000 iterations. For panoptic segmentation, the ROI box head is modified to align with the SlotCon object detection architecture, consisting of four convolutional layers followed by a fully connected layer. Model performance is evaluated on the COCO *val2017* set using the last saved checkpoint.

Similarly we adapt the default detectron2 configuration for human dense pose estimation. We fine-tune a densepose R-CNN with a ResNet-50 encoder and an FPN decoder for 130000 iterations on COCO *train2014*, with a batch size of 16 and a learning rate of 0.01. The ROI box Head is also adapted to match SlotCon object detection architecture with four convolution layers and one fully connected layer. We evaluate performance on the COCO *val2014* using the last checkpoint saved.

3. SlotCon adaptations

Figure S1 presents the DeCon-SL adaption of SlotCon SSL framework to an encoder-decoder framework. Figure S2 and Figure S3 illustrates how we used channel dropout and decoder deep-supervision for the DeCon-ML-L and DeCon-ML-S adaptations of SlotCon framework with an FPN decoder pre-training.

4. Performance comparison with state of the art methods

Table S1 presents a performance comparison between our best models, DeCon-ML-L ($\alpha = 0$, dropout$= 0.5$) and DeCon-SL ($\alpha = 0.25$) adaptations of SlotCon, and existing state-of-the-art methods. This table extends Tab. 1 from the main manuscript to better reflect the range of performances of the different SSL methods.

5. Encoder and Decoder transfer

Tab. S2 presents the fine-tuning performance of the DeCon framework when transferring only the encoder versus transferring both the encoder and decoder. We observe that transferring both components does not consistently lead to a performance gain for the DeCon-SL and DeCon-ML-L adaptations of SlotCon. In contrast, a noticeable improvement is observed when both the encoder and decoder are transferred in the DeCon-SL adaptations of DenseCL and PixPro (see Tab. 4) in the main manuscript).

6. DeCon-ML-L slot selection

Figure 3 in the main manuscript illustrates the visual representation of slots extracted from both encoder and decoder outputs. An image from the COCO *val2017* dataset was used as input to SlotCon and DeCon-ML-L. Feature maps were extracted and projected using their respective projectors from the encoder bottleneck of both architectures, as well as from multiple decoder levels in the DeCon-ML-L framework. For each feature map (e.g., a 7×7 spatial map from the SlotCon encoder, a 14×14 map from level 2 of the DeCon-ML-L decoder or a 56×56 map from level 4 of the DeCon-ML-L decoder), each spatial location (pixel/vector) was assigned to the prototype with which it had the highest similarity, measured via dot product. The prototype most frequently assigned across all spatial positions was designated as the reference prototype (i.e., the slot) for that feature map.

To visualize this slot on the image, the dot product similarity between all feature vectors and the 256 learned prototypes was computed, resulting in a similarity map that was upsampled to the input image resolution (224×224). A voxel-wise argmax over the prototype dimension was used to determine prototype assignments, and a binary mask was generated by selecting only the pixels assigned to the reference prototype. This mask was overlaid on the image to highlight the most important slot/concept extracted from this image. A first mask was derived from the SlotCon encoder output and served as the reference slot.

For DeCon-ML-L, since separate prototypes—distinct from those of the SlotCon encoder—were learned at the encoder and at each decoder level, we repeated the same procedure to generate masks from corresponding feature map for all of the prototypes. For each of these, we identified the prototype whose mask showed the largest overlap with the SlotCon reference mask. This most overlapping prototype was considered to represent the same underlying concept, and its corresponding mask was overlaid on the reference image for comparison.

7. DeCon-SL with an FPN decoder

Table S3 presents the fine-tuning performance of the DeCon-SL adaptation of SlotCon with an FPN decoder. In this setting, two losses were used during the pre-training: the encoder loss and only the loss from the final layer of the FPN decoder. The results demonstrate that DeCon-SL consistently outperforms the original SlotCon when using an FPN decoder. As shown in the main manuscript, similar improvements were observed with an FCN decoder, suggesting that this approach is potentially generalizable to other decoder architectures. The greatest performance gains were observed when both the encoder and decoder were transferred. Notably, the DeCon-SL adaptation showed minimal sensitivity to the addition of dropout, even when the decoder included skip connections—a behavior that contrasts with the multi-level adaptation (DeCon-ML), where dropout had a more pronounced effect.

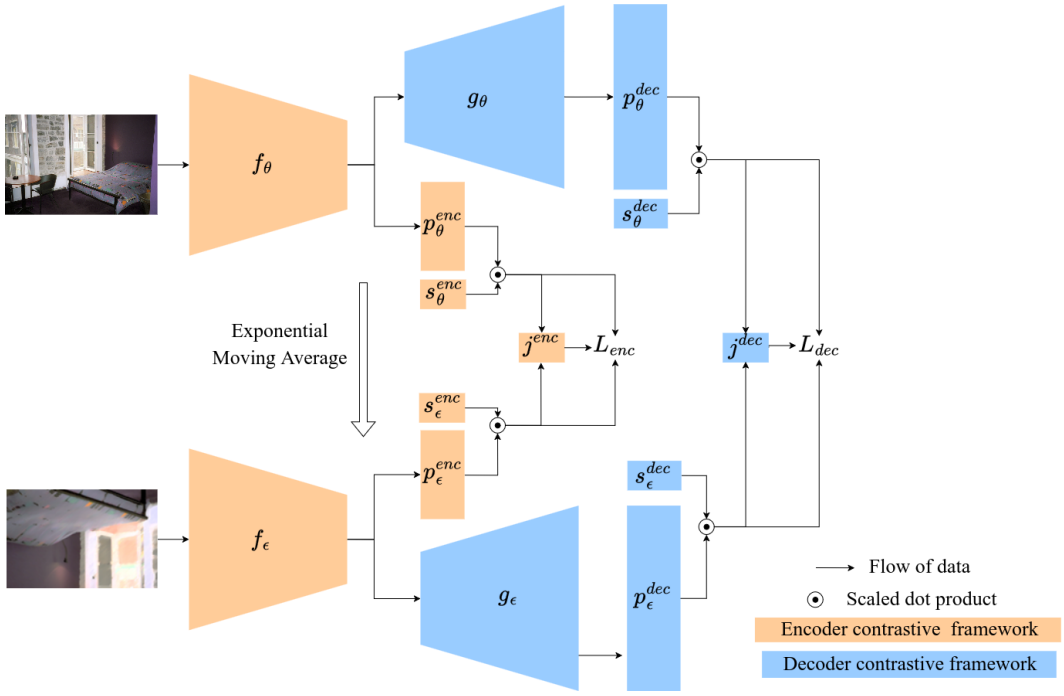

Figure S1. This figure illustrates the proposed DeCon-SL adaptation of SlotCon framework with an FCN decoder. f_θ and f_ϵ represent student and teacher encoders, respectively. p_θ^{enc} and p_ϵ^{enc} are the projector layers of the SSL frameworks of the student and teacher encoders, respectively. s_θ^{enc} and s_ϵ^{enc} are the semantic grouping layers of the SSL frameworks of the student and teacher encoders, respectively. j^{enc} is the encoder predictor slot. g_θ and g_ϵ represent student and teacher decoders, respectively. p_θ^{dec} and p_ϵ^{dec} are the projector layers of the SSL frameworks of the student and teacher decoders, respectively. s_θ^{dec} and s_ϵ^{dec} are the semantic grouping layers of the SSL frameworks of the student and teacher decoders, respectively. j^{dec} is the decoder predictor slot.

8. Decoder-specific hyperparameter tuning

In all our previous experiments, hyperparameters used in the "auxiliary layers" of the encoder were replicated at the decoder level. Table S4 shows that tuning these hyperparameters for the decoder could result in better downstream performance. Using 384 prototypes to compute the decoder pre-training loss results in better downstream performance for the DeCon-SL adaptation than using the encoder's parameter from the original SlotCon framework: 256 prototypes.

9. Randomness of the result

To account for variability arising from the stochasticity of training and the random initialization of the non-pre-trained components of the architecture, each downstream experiment was repeated three times. Table S5 reports the standard deviation of the fine-tuning performance. Furthermore, the Cohen's d values in Tab. S6 of the main paper provide additional evidence that the observed performance improvements are statistically significant.

10. Statistical Significance Tests

In order to quantify the difference between mean metrics obtained when fine-tuning two different pre-trained models, we caclucated Cohen's d value using the following formula:

$$d = \frac{M_1 - M_2}{s_p}$$

where M_1 and M_2 are the means of the two groups, and s_p is the pooled standard deviation given by:

$$s_p = \sqrt{\frac{(n_1 - 1)s_1^2 + (n_2 - 1)s_2^2}{n_1 + n_2 - 2}}$$

Here, s_1 and s_2 are the standard deviations, and n_1, n_2 are the sample sizes of the two groups that we consider. In our case the sample size was 3 as we performed 3 runs for each fine-tuning experiments. The highly positive d values obtained when comparing SlotCon and DeCon-ML-L fine-tuning performance in Tab. S6 indicates that our framework has a better average fine-tuning performance. Table S6 also shows the performance improvements achieved with our DeCon framework compared to the original framework.

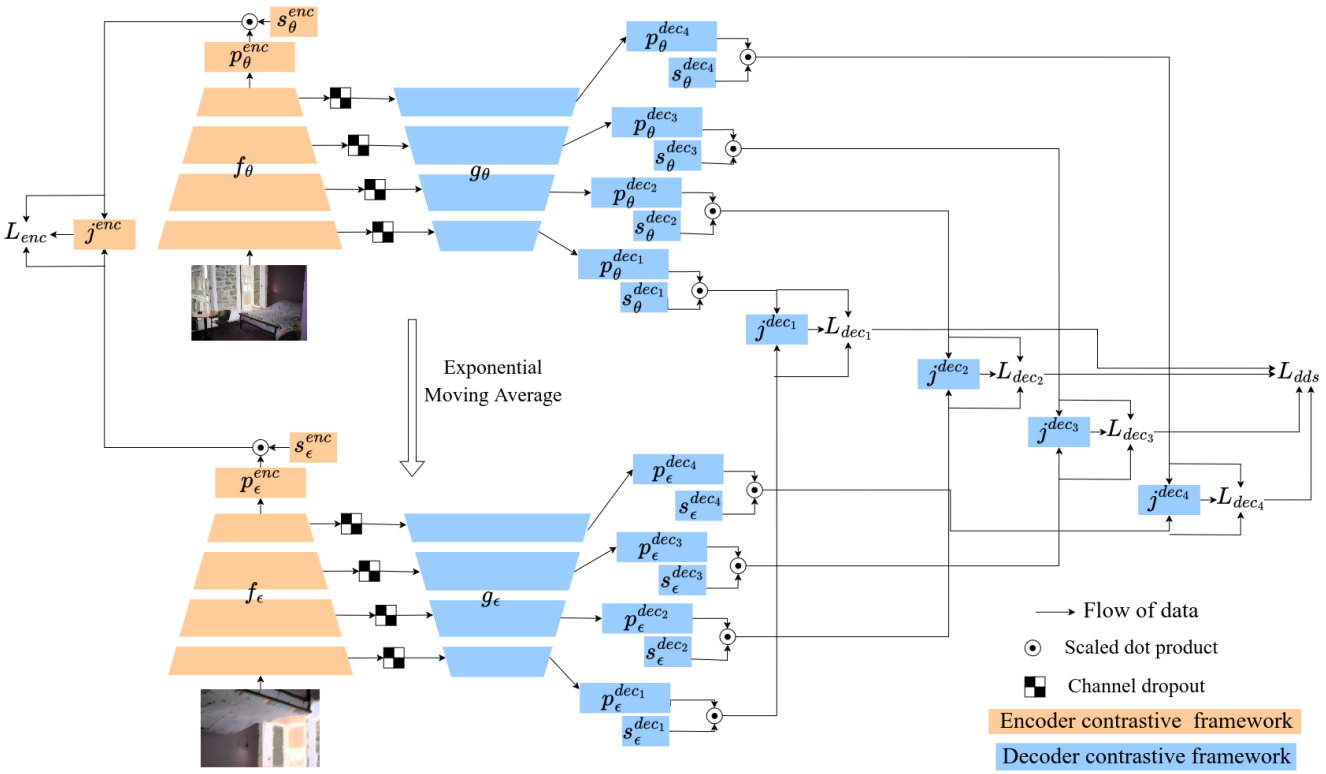

Figure S2. This schema illustrates the DeCon-ML-L adaptation of SlotCon framework with an FPN decoder. It also depicts the proposed decoder deep supervision and channel dropout. f_θ and f_ϵ represent student and teacher encoders, respectively. p_θ^{enc} and p_ϵ^{enc} are the projector layers of the SSL frameworks of student and teacher encoders, respectively. s_θ^{enc} and s_ϵ^{enc} are the semantic grouping layers of the SSL frameworks of the student and teacher encoders, respectively. j^{enc} is the encoder predictor slot. g_θ and g_ϵ represent student and teacher decoders, respectively. $p_\theta^{dec_i}$ and $p_\epsilon^{dec_i}$ are the projector layers of the SSL frameworks of the student and teacher decoders, respectively. $s_\theta^{dec_i}$ and $s_\epsilon^{dec_i}$ are the semantic grouping layers of the SSL frameworks of the student and teacher decoders, respectively. j^{dec} is the decoder predictor slot.

To evaluate the significance of the performance improvement of our framework over the original framework SlotCon, we performed a Wilcoxon signed-rank test on the Average Precision (AP) scores. We first computed per-image Average Precision (AP) scores for the three fine-tuning runs and stored the mean AP value of each image. We then perform the statistical test on these values. The p-values obtained, 0.012 for COCO object detection and 0.048 for instance segmentation, show the significance of the performance improvement of the proposed framework.

References

[1] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jegou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9630–9640, 2021. 7

[2] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 801–818, 2018. 2

[3] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning, 2020. 7

[4] Noel CF Codella, David Gutman, M Emre Celebi, Brian Helba, Michael A Marchetti, Stephen W Dusza, Aadi Kalloo, Konstantinos Liopyris, Nabin Mishra, Harald Kittler, et al. Skin lesion analysis toward melanoma detection: A challenge at the 2017 international symposium on biomedical imaging (isbi), hosted by the international skin imaging collaboration (isic). In *2018 IEEE 15th international symposium on biomedical imaging (ISBI 2018)*, pages 168–172. IEEE, 2018. 1, 2

[5] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3223, 2016. 1

[6] Mark Everingham, SM Ali Eslami, Luc Van Gool, Christo-

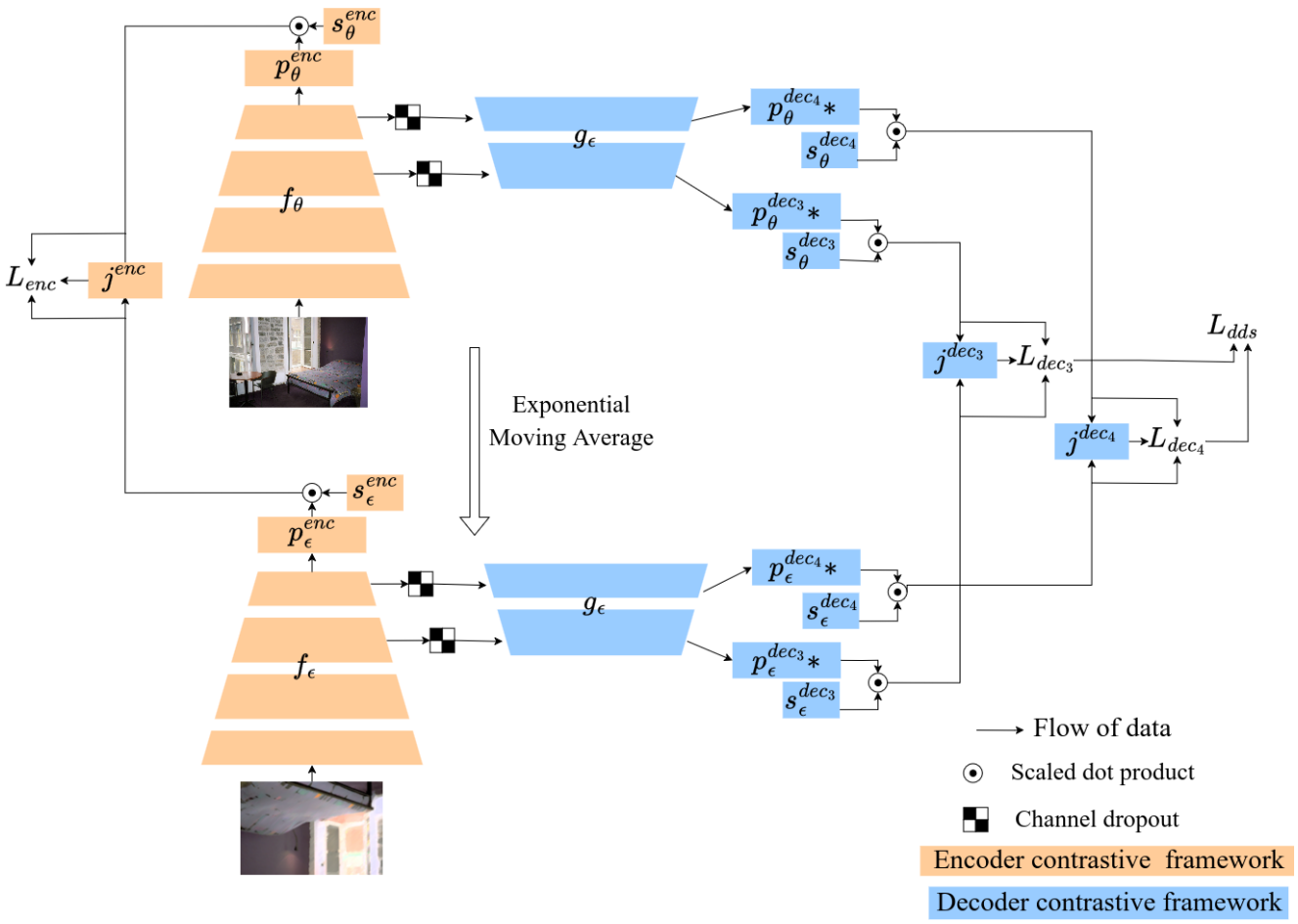

Figure S3. This schema illustrates the DeCon-ML-S adaptation of SlotCon framework with an FPN decoder. DeCon-ML-S is a smaller version of DeCon-ML-L where only the two first decoder levels are used and the decoder projector hidden dimension is reduced. * means the hidden dimension was altered from 4096 to 2048.

pher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes challenge: A retrospective. *International journal of computer vision*, 111:98–136, 2015. 1

[7] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, Bilal Piot, koray kavukcuoglu, Remi Munos, and Michal Valko. Bootstrap your own latent - a new approach to self-supervised learning. In *Advances in Neural Information Processing Systems*, pages 21271–21284. Curran Associates, Inc., 2020. 7

[8] Olivier J. Hénaff, Skanda Koppula, Jean-Baptiste Alayrac, Aaron van den Oord, Oriol Vinyals, and João Carreira. Efficient visual pretraining with contrastive detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 10086–10096, 2021. 7

[9] Olivier J. Hénaff, Skanda Koppula, Evan Shelhamer, Daniel Zoran, Andrew Jaegle, Andrew Zisserman, João Carreira, and Relja Arandjelović. Object discovery and representation networks. In *Computer Vision – ECCV 2022*, pages 123–143, Cham, 2022. Springer Nature Switzerland. 7

[10] Junqiang Huang, Xiangwen Kong, and Xiangyu Zhang. Revisiting the critical factors of augmentation-invariant representation learning, 2022. 7

[11] Zhaowen Li, Yousong Zhu, Fan Yang, Wei Li, Chaoyang Zhao, Yingying Chen, Zhiyang Chen, Jiahao Xie, Liwei Wu, Rui Zhao, Ming Tang, and Jinqiao Wang. Univip: A unified framework for self-supervised visual pre-training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14627–14636, 2022. 7

[12] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft COCO: Common Objects in Context, 2015. arXiv:1405.0312 [cs]. 1

[13] José Ignacio Orlando, Huazhu Fu, João Barbosa Breda,

Table S1. Performance comparison with state-of-the-art SSL frameworks pre-trained on COCO, COCO+ and ImageNet-1K. DeCon-ML-L and DeCon-ML-S were pre-trained with FPN decoder ($\alpha = 0, dropout = 0.5$) and DeCon-SL was pre-trained with FCN decoder ($\alpha = 0.25$) adaptation of SlotCon. We only transferred the encoder for downstream tasks. We averaged fine-tuning results over three runs. †: Collected from PixCon paper. ♢ Collected from SlotCon Paper ‡: Full re-implementation.

Pret. Dataset	Framework	Pret. Dec.	Object Detection COCO			Instance Segmentation COCO			Semantic Segmentation VOC	City	ADE
			AP	AP50	AP75	AP	AP50	AP75	mIoU	mIoU	mIoU
	Random init.♢	-	32.8	50.9	35.3	29.9	47.9	32.0	39.5	65.3	29.4
COCO	**MoCo-v2 (2020) ♢ [3]**	-	38.5	58.1	42.1	34.8	55.3	37.3	69.2	73.8	36.2
	BYOL (2020) † [7]	-	39.5	59.4	43.3	35.6	56.6	38.2	70.2	75.3	-
	MoCo-v2 + (2022) † [10]	-	39.8	59.7	43.6	35.9	57.0	38.5	71.1	75.6	-
	ORL (2021) ♢ [25]	-	40.3	60.2	44.4	36.3	57.3	38.9	70.9	75.6	36.7
	PixPro (2021) ♢ [26]	-	40.5	60.5	44.0	36.6	57.8	39.0	72.0	75.2	38.3
	DetCon (2021) ♢ [8]	-	39.8	59.5	43.5	35.9	56.4	38.7	70.2	76.1	38.1
	UniVIP (2022) † [11]	-	40.8	-	-	36.8	-	-	-	-	-
	Odin (2022) † [9]	-	40.4	60.4	44.6	36.6	57.5	39.3	70.8	75.7	-
	DenseCL (2021) ♢ [18]	-	39.6	59.3	43.3	35.7	56.5	38.4	71.6	75.8	37.1
	DenseCL-D (2024) [15]	-	39.3	58.7	42.6	34.2	55.7	36.5	-	-	-
	SoCo-D (2024) [15]	-	40.3	60.1	44.0	35.1	56.9	37.6	-	-	-
	PixCon-SR (2024) † [14]	-	40.81	60.97	44.80	36.80	57.93	39.62	72.95	**76.62**	38.0
	Slotcon (2022) ‡ [21]	-	40.81	60.95	44.37	36.80	57.98	39.54	71.50	75.95	38.57
	DeCon-SL (SlotCon adapt.) [ours]	FCN	40.97	61.22	44.81	36.92	58.12	39.78	73.01	76.21	**38.81**
	DeCon-ML-S (SlotCon adapt.) [ours]	FPN	40.97	61.20	44.71	36.94	58.20	39.63	72.80	76.21	38.36
	DeCon-ML-L (SlotCon adapt.) [ours]	FPN	**41.18**	**61.38**	**44.91**	**37.12**	**58.35**	**39.94**	72.92	76.45	38.70
COCO+	**ORL (2021) † [25]**	-	40.6	-	-	36.7	-	-	-	-	-
	UniVIP (2022) † [11]	-	41.1	-	-	37.1	-	-	-	-	-
	PixCon-SR (2024) † [14]	-	41.2	-	-	37.1	-	-	73.9	77.0	38.8
	Slotcon (2022) ‡ [21]	-	41.63	62.10	45.67	37.57	59.07	40.45	73.93	76.43	39.11
	DeCon-SL (SlotCon adapt.) [ours]	FCN	41.86	62.43	45.73	37.75	59.40	40.48	74.46	76.65	**39.25**
	DeCon-ML-L (SlotCon adapt.) [ours]	FPN	**42.08**	**62.42**	**46.13**	**37.84**	**59.41**	**40.75**	**75.36**	77.00	39.04
ImageNet-1K	**Supervised ♢**	-	39.7	59.5	43.3	35.9	56.6	38.6	74.4	74.6	37.9
	MoCo-v2 (2020) ♢ [3]	-	40.4	60.1	44.2	36.5	57.2	39.2	73.7	76.2	36.9
	DetCo (2021) ♢ [24]	-	40.1	61.0	43.9	36.4	58	38.9	72.6	76	37.8
	InsLoc (2021) ♢ [27]	-	40.9	60.9	44.7	36.8	57.8	39.4	72.9	75.4	37.3
	DenseCL (2021) ♢ [18]	-	40.3	59.9	44.3	36.4	57	39.2	72.8	76.2	38.1
	PixPro (2021) ♢ [26]	-	40.7	60.5	44.8	36.8	57.4	39.7	73.9	**76.8**	38.2
	DetCon (2021) ♢ [8]	-	40.6	-	-	36.4	-	-	72.6	75.5	-
	DINO (2021) ‡ [1]	-	40.24	60.25	44.13	36.47	57.49	39.20	73.09	75.57	37.30
	SoCo (2022) ♢ [19]	-	41.6	61.9	45.6	37.4	58.8	40.2	71.9	76.5	37.8
	Slotcon (2022) ‡ [21]	-	41.69	62.07	45.59	37.59	58.97	40.49	75.02	76.15	38.97
	DeCon-ML-L (SlotCon adapt.) [ours]	FPN	**41.80**	**62.12**	**45.73**	**37.73**	**59.08**	**40.68**	**75.40**	76.51	**39.01**

Table S2. Performance of DeCon-SL ($\alpha = 0.25$) and DeCon-ML-L ($\alpha = 0$ and $dropout = 0.5$) adaptations of SlotCon trained with a ResNet-50 backbone. The pre-trained encoder and decoder were used in the fine-tuning. All fine-tuning results were averaged over three runs.

Pret. Dataset	Framework	Pret. Dec.	Transfer	Obj. Det. COCO	Inst. Seg. COCO	Sem. Seg. VOC	Sem. Seg. City
				AP	AP	mIoU	mIoU
COCO	SlotCon	-	Enc	40.81	36.80	71.50	75.95
	DeCon-SL	FCN	Enc	-	-	**73.01**	76.21
			Enc + Dec	-	-	72.96	**76.28**
	DeCon-ML-L	FPN	Enc	41.18	**37.12**	-	-
			Enc + Dec	**41.21**	37.11	-	-

Karel Van Keer, Deepti R Bathula, Andrés Diaz-Pinto, Ruogu Fang, Pheng-Ann Heng, Jeyoung Kim, JoonHo Lee, et al. Refuge challenge: A unified framework for evaluating automated methods for glaucoma assessment from fundus photographs. *Medical image analysis*, 59:101570, 2020. 1, 2

[14] Zongshang Pang, Yuta Nakashima, Mayu Otani, and Hajime Nagahara. Revisiting pixel-level contrastive pre-training on scene images. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 1784–1793, 2024. 1, 7

[15] Congpei Qiu, Tong Zhang, Yanhao Wu, Wei Ke, Mathieu Salzmann, and Sabine Süsstrunk. Mind your augmentation: The key to decoupling dense self-supervised learning. In *The Twelfth International Conference on Learning Representations*, 2024. 7

[16] Roboflow. Detecting diseases dataset. https://universe.roboflow.com/artificial-

Table S3. Performance of a DeCon-SL adaptation of SlotCon using a ResNet-50 encoder and an FPN decoder. The architecture was pre-trained for 800 epochs on COCO, and then fine-tuned for object detection and instance segmentation tasks. The results are obtained with varying dropout rates (0 or 0.5) as well as transferring only the pre-trained encoder or both the pre-trained encoder and decoder. All fine-tuning results were averaged over three runs.

Pret. Dataset	Framework	Pret. Dec.	Transfer	α	Dropout	Object Detection COCO			Instance Segmentation COCO		
						AP	AP50	AP75	AP	AP50	AP75
COCO	SlotCon	-	enc	1	-	40.81	60.95	44.37	36.80	57.98	39.54
	DeCon-SL	FPN	enc	0.5	0	40.94	60.97	44.82	36.92	58.03	39.80
			enc	0.5	0.5	40.90	61.23	44.64	36.91	58.30	39.67
			enc+dec	0.5	0	41.05	61.34	44.85	37.03	58.33	39.86

Table S4. Decoder specific hyperparameter tuning. We experiment on the number of prototypes to be used for the decoder pre-training in DeCon-SL adaptation. α is fixed to 0.5 and the number of prototypes used to compute the encoder loss is fixed to 256. We report the fine-tuning mIoU as an average over three runs.

Dataset	SSL	Loss		Transfer		Number of Dec. Prototypes					
		L_{enc}	L_{dec}	enc	dec	0	64	128	256	384	512
VOC	SlotCon	✓	✗	✓	✗	71.50	-	-	-	-	-
	DeCon-SL	✓	✓	✓	✗	-	72.10	71.95	72.42	72.43	72.32
		✓	✓	✓	✓	-	72.64	72.60	72.42	**72.79**	72.75
City	SlotCon	✓	✗	✓	✗	75.95	-	-	-	-	-
	DeCon-SL	✓	✓	✓	✗	-	75.67	75.79	75.67	75.87	75.97
		✓	✓	✓	✓	-	75.69	75.57	76.00	**76.16**	76.14

Table S5. COCO and COCO+ pre-training downstream performance with standard deviation over 3 runs. Results for PixCon were obtained directly from the original PixCon publication, while SlotCon results were derived from full re-implementation. PixCon standard deviation of COCO+ pre-training was not provided due to unavailability of the checkpoint or reported standard deviation in the main paper.

Method	COCO obj det. AP	COCO inst. Seg. AP	VOC sem. Seg. mIoU	City sem. Seg. mIoU
(a) COCO Pre-training Performance- ResNet50 backbone- Presented as mean over 3 runs ± std				
SlotCon (2022)	40.81±0.16	36.80±0.18	71.50±0.27	75.95±0.23
PixCon (2024)	40.81±0.09	36.84±0.11	72.95±0.29	76.62±0.10
DeCon-ML (Comp. to SlotCon)	41.18±0.15	37.12±0.14	72.92±0.16	76.45±0.16
(b) COCO+ Pre-training performance- ResNet50 backbone - Presented as mean over 3 runs ± std				
SlotCon (2022)	41.63±0.09	37.57±0.11	73.93±0.33	76.43±0.40
PixCon (2024)	41.2	37.1	73.9	77.0
DeCon-ML (Comp. to SlotCon)	42.08±0.08	37.84±0.05	75.36±0.21	77.00±0.53

intelligence-82oex/detecting-diseases/dataset/6, 2022. visited on 2025-03-21. 2

[17] Davinder Singh, Naman Jain, Pranjali Jain, Pratik Kayal, Sudhakar Kumawat, and Nipun Batra. Plantdoc: A dataset for visual plant disease detection. In *Proceedings of the 7th ACM IKDD CoDS and 25th COMAD*, pages 249–253. 2020. 2

[18] Xinlong Wang, Rufeng Zhang, Chunhua Shen, Tao Kong, and Lei Li. Dense contrastive learning for self-supervised visual pre-training. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3023–3032, 2021. 7

[19] Fangyun Wei, Yue Gao, Zhirong Wu, Han Hu, and Stephen Lin. Aligning pretraining for detection via object-level contrastive learning. In *Advances in Neural Information Processing Systems*, pages 22682–22694. Curran Associates, Inc., 2021. 7

[20] Tianqi Wei, Zhi Chen, Xin Yu, Scott Chapman, Paul Melloy, and Zi Huang. Plantseg: A large-scale in-the-wild dataset for plant disease segmentation. *arXiv preprint arXiv:2409.04038*, 2024. 2

[21] Xin Wen, Bingchen Zhao, Anlin Zheng, Xiangyu Zhang, and

Table S6. COCO and COCO+ pre-training downstream performance improvement with DeCon-ML-L and Cohen's d value. PixCon performance was extracted from the original paper.

Method	COCO obj det.	COCO Inst. Seg.	VOC Sem. Seg.	City Sem. Seg.
	ΔAP	ΔAP	ΔmIoU	ΔmIoU
COCO Pre-training Performance Improvement Δ (calculated from the respective papers)				
SlotCon (2022) (Comp. to prev. works)	+0.50	+0.40	-0.40	+0.10
PixCon (2024) (Comp. to SlotCon)	0.00	+0.06	+1.30	+0.51
DeCon-ML-L (Comp. to SlotCon)	+0.37	+0.32	+1.42	+0.50
Cohen's d (DeCon-ML-L VS SlotCon)	2.35	1.94	6.37	2.51
COCO+ Pre-training Performance Improvement Δ (calculated from the respective papers)				
SlotCon (2022) (Comp. to prev. works)	+0.60	+0.50	Not avail.	Not avail.
PixCon (2024) (Comp. to SlotCon)	-0.50	-0.50	-0.20	+0.40
DeCon-ML-L (Comp. to SlotCon)	+0.45	+0.27	+1.43	+0.57
Cohen's d (DeCon-ML-L VS SlotCon)	5.42	3.21	5.25	1.22

Xiaojuan Qi. Self-supervised visual representation learning with semantic grouping. In *Advances in Neural Information Processing Systems*, pages 16423–16438. Curran Associates, Inc., 2022. 1, 2, 7

[22] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. https://github.com/facebookresearch/detectron2, 2019. 1, 2

[23] Tete Xiao, Yingcheng Liu, Bolei Zhou, Yuning Jiang, and Jian Sun. Unified perceptual parsing for scene understanding. In *Proceedings of the European conference on computer vision (ECCV)*, pages 418–434, 2018. 2

[24] Enze Xie, Jian Ding, Wenhai Wang, Xiaohang Zhan, Hang Xu, Peize Sun, Zhenguo Li, and Ping Luo. Detco: Unsupervised contrastive learning for object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 8392–8401, 2021. 7

[25] Jiahao Xie, Xiaohang Zhan, Ziwei Liu, Yew Soon Ong, and Chen Change Loy. Unsupervised object-level representation learning from scene images. *Advances in Neural Information Processing Systems*, 34:28864–28876, 2021. 7

[26] Zhenda Xie, Yutong Lin, Zheng Zhang, Yue Cao, Stephen Lin, and Han Hu. Propagate yourself: Exploring pixel-level consistency for unsupervised visual representation learning. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16679–16688, 2021. 7

[27] Ceyuan Yang, Zhirong Wu, Bolei Zhou, and Stephen Lin. Instance localization for self-supervised detection pretraining. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3987–3996, 2021. 7

[28] Bolei Zhou, Hang Zhao, Xavier Puig, Tete Xiao, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Semantic understanding of scenes through the ade20k dataset. *International Journal of Computer Vision*, 127(3):302–321, 2019. 1