A Data-Centric Revisit of Pre-Trained Vision Models for Robot Learning

Supplementary Material

Contents

A. Extended Implementation Details

A.1. Pre-Training Details

Architecture. We use ViT-B/16 [21] as our backbone. Following common practice in DINO [13], iBOT [80] and MoCo-v3 [18], the projector g is a 3-layer MLPs with hidden dimension 2048 and output dimension 256, and the predictor h is a 2-layer MLPs with hidden dimension 4096 and output dimension 256.

Augmentation and masking. We use the same augmentation strategy as in iBOT [80] except not using small local crops (multi-crop). Avoiding the use of multi-crop saves significant computational costs in our model, and the model-learned slots work in a similar role. The masking strategy follows iBOT [80], with prediction ratio r uniformly sampled from range $[0.3 - 0.2, 0.3 + 0.2]$.

Optimization. Most optimization configurations follow DINO [13] and iBOT [80]. We use AdamW optimizer with a cosine schedule for the learning rate and weight decay. The learning rate is linearly ramped up during the first 10 epochs to 1.5×10^{-4} scaled with the total batch size: $lr = lr_{\text{base}} \times \text{batch size}/256$, and then decays following the cosine schedule. The weight decay starts from 0.4 and also decays following the cosine schedule, to 0.04 when training ends. We train for 800 epochs on 241K-scale datasets and 400 epochs on 1.28M-scale datasets, with a batch size of 1024 distributed across 8 A100 GPUs. For experiments on 4M-scale datasets, we train 200 epochs.

Hyperparameters. Follow DINO [13] and iBOT [80], the teacher temperature τ_t linearly ramps up from 0.04 to 0.07 for the first 30 epochs and remains constant afterwards. The student temperature τ_s is fixed at 0.1. The number of prototypes C is set to 512 for COCO+ and 1024 for other datasets.

A.2. Evaluation Details

Manipulation tasks. Following the setup of [33], we use a shallow 4-layer MLP with hidden sizes $[512, 256, 128]$ and ReLU activations as the policy network for behavior cloning. The 5 Franka Kitchen tasks, 8 Meta-World tasks, and corresponding GT demonstrations are also taken from [33]. Following R3M [47] and VC-1 [45], the policy training involves mini-batches of 128 samples, conducted over 20000 steps with the Adam optimizer and a learning rate of 0.001. The model is evaluated every 1000 steps. All tasks and environments use 224×224 RGB images without proprioceptive input and without image augmentations, and each task uses only 25 demonstrations for training, which raises higher requirements on the quality of PVMs. For better measurement of the potential of PVMs, we follow [36] and use attentive pooling (also known as multihead attention pooling [40]), which is suggested as a strong and versatile approach [78] as opposed to the commonly used `[CLS]` token and provides better comparisons between frozen PVMs. Following VC-1 [45], we take the best checkpoint for each run on each task. For each environment, we report the average performance over all tasks (3 independent runs each task).

Navigation tasks. The settings on navigation tasks strictly follows VC-1 [45], which involves object-goal navigation [6] and image-goal navigation [81]. In both, the agent is initialized at a random location in an unknown 3D environment and is tasked to find the goal location specified by an image or object. Both tasks are conducted using Habitat [59] simulator, in which ObjectNav is conducted in the HM3D [57] environment and ImageNav is conducted in the Gibson [71] environment. For ObjectNav, the agent is trained for approximately 400M steps with 512 parallel environments. For ImageNav, the agent is trained for 500M steps with 320 parallel environments. Further details can be found in [45] and omitted here for brevity.

Semantic segmentation on ADE20K. We use UperNet [72] implemented in MMSegmentation following iBOT [80]. Specifically, we fine-tune for 160k iterations with stochastic gradient descent, with a batch size of 16 and weight decay of 0.0005. The learning rate is 0.01 and decays following

the poly schedule with power of 0.9 and min_lr of 0.0001.

Object detection and instance segmentation on COCO. COCO object detection and instance segmentation setting also follows iBOT [80], where the pre-trained model initialized a Cascade Mask R-CNN [9]. The image scale is $[640, 800]$ during training and 800 at inference. We fine-tune all layers end-to-end on COCO [41] `train2017` set with the standard $1\times$ schedule and report AP for boxes and masks on the `val2017` set.

Analytical metrics. Some numeric indicators are considered to help analyze properties of pre-trained models, *e.g.*, object discovery ability (objectness of attention maps) measured by Pascal VOC 2012 object segmentation quality. The Jaccard index measures the overlap between predicted mask P and the ground truth mask G as $J(P, G) = \frac{G \cap P}{G \cup P}$. Following [8, 48], the attention maps of DINO and iBOT are computed between the `[CLS]` token and patch tokens in the last layer, and the attention maps of SlotMIM are computed between the prototypes and projected patch tokens. For each object of interest, the attention head/prototype producing the best Jaccard index is selected. Besides, for the ablation studies, we also report k-NN ImageNet classification ($k = 20$) accuracy following DINO [13]. Additionally, we maintain a running mean of the average number of active (assigned to at least one patch) slots $\overline{K}$ in an image during training.

A.3. Details for Fig. 1

(a) BC performance regarding dataset and model. Fig. 1a is a grouped version of Franka Kitchen and Meta-World results in Fig. 2 in the main paper. For each grid, we report the average performance over all manipulation tasks given a PVM pre-trained on a specific dataset (by row) using a specific model (by column).

(b) Correlation between objectness and BC performance. Fig. 1b is a joined view between the average manipulation performance (first two subfigures) and VOC object segmentation performance (third subfigure) in Fig. 2 in the main paper. It is presented as a scatter plot to show the (Pearson's) correlation between the two metrics.

(d) Visualization of attention maps. Fig. 1b and Fig. 1d show examples of the attention maps of DINO and SlotMIM, respectively. All models are pre-trained on 241K-scale datasets for 800 epochs. For DINO, we follow the official implementation [13] to take the attention maps between the `[CLS]` token and patch tokens in the last layer, and show the best attention head. For SlotMIM, we take the attention maps (prototype assignments) between projected patch tokens and the "cat" prototype—the prototype with the highest cosine similarity with "cat" segments.

A.4. Details for Fig. 4

(a) Visualization of attention maps. All attention maps in Fig. 4a are computed in the same manner as SlotMIM in Fig. 1d—between the prototypes and projected patch tokens. All models are pre-trained on COCO+ (scene-centric) for 800 epochs. For iBOT, we trained two variants—one with 8192 prototypes (dimension of the last layer in the DINOHead in implementation) as default, and one with 512 prototypes. In the visualization, each prototype is assigned a random color.

(b) Visualization of segmentation consistency. Fig. 4b shows the segments assigned to each prototype following the implementation of [69]. We first obtain all segments (prototype assignments) on COCO `val2017` set, pool them to slots, and then retrieve the nearest-neighbor slots for each prototype. Then for each selected prototype, we visualize the segments of the top-5 similar slots.

B. Extended Analysis and Discussion

B.1. Why Pre-Training on NOC Data?

Self-supervised pre-training have benefited numerous downstream tasks, with a key advantage in its ability to learn representations from unlabeled data, eliminating the need for human annotations and making it easier to scale up training datasets. Despite this advantage in utilizing diverse types of data, most research has focused on (single-)object-centric datasets like ImageNet for model development, leaving large volumes of non-object-centric (NOC) data, such as Open Images [39], SA-1B [38], LAION [61], and Ego4D [24], underutilized. However, many primary application domains of self-supervised learning—such as robot learning (manipulation, navigation, *etc.*), or traditional perception tasks like object detection and image segmentation—often require handling NOC data. This motivates us to explore the potential of NOC data for self-supervised learning, which could bridge the data-domain gap between self-supervised learning and real-world applications, and is rich in information, offering new opportunities for data scaling.

B.2. Why Consider Both Control and Perception?

The latest development of robot learning has witnessed a shift towards mulit-modal generalist models that are capable of handling diverse robot tasks, in which both control and perception are indispensable. This involves integrating PVMs (*e.g.*, CLIP [54], T5 [56], and DINOv2 [52]) as multi-modal encoders [7, 15, 37, 50, 70, 82] or explicitly utilizing foundation models (*e.g.*, OWL-ViT [46], SAM [38], and GPT-4V [51]) as tools [23, 32, 34, 35, 49]. It is also shown that improvements in perception can lead to policies that generalize better to unseen environments using less data [64]. In accordance with this trend, recent research in PVM for robot

C	k-NN	ADE	Jacc	$\overline{K}$
256	45.3	**49.1**	42.2	7.8
512	**46.2**	**49.1**	**43.9**	9.4
1024	45.6	48.4	42.8	10.8

(a) **Number of prototypes**

	k-NN	ADE	Jacc	$\overline{K}$
0.3	**46.2**	**49.1**	43.9	9.4
0.4	45.8	48.6	45.0	8.1
0.5	44.3	48.2	**45.7**	7.1

(b) **Mask ratio ($\pm$0.2)**

τ_t	k-NN	ADE	Jacc	$\overline{K}$
0.04→0.07	**46.2**	**49.1**	**43.9**	9.4
0.07	45.8	48.6	42.1	8.1
0.07→0.04	45.5	**49.1**	42.6	7.1

(c) **Teacer temp. schedule**

Type	k-NN	ADE	Jacc	$\overline{K}$
center	**46.2**	49.1	**43.9**	9.4
SH	45.1	**49.3**	40.8	15.2

(d) **Patch loss**

Table 5. **Ablation studies on hyperparameters.** Default values are marked with a cyan background.

learning has also started to consider more diverse evaluation protocols [36, 45, 77]. Serving as the visual cortex of a modern robot, we believe that a good PVM should enhance both perception and control abilities.

B.3. What Determines the Objectness of SlotMIM?

We consider two quantitative metrics for the objectness of SlotMIM: 1) the Jaccard index between the attention maps and the ground truth object masks in Pascal VOC; and 2) the average number of active slots $\overline{K}$ in an image during training. The former measures the *alignment* of the attention maps to objects, and the latter roughly measures the *granularity* of the concepts represented by the attention maps—the more fine-grained the concepts are, the more parts (slots) each image is segmented into.

In the main paper, we have shown that both pre-training dataset and model hyper-parameters can affect the objectness of SlotMIM. Concerning the dataset, object-centric data leads to better alignment to common objects, scene-centric data and web-crawled data lead to slightly worse segmentation quality and similar-level granularity, and ego-centric data leads to very fine-grained segmentation. Concerning the model hyper-parameters, we will show in Appendix C.1 that there are multiple factors can affect the segmentation quality and granularity.

B.4. Fine-grained Slots for Manipulation Tasks?

In the main paper, we have shown that while scaling up non-ego-centric data for SlotMIM can improve segmentation quality and navigation/perception performance, manipulation performance drops due to over-compression. We are curious if learning fine-grained slots/concepts can improve the transferability of SlotMIM pre-trained on non-ego-centric data to manipulation tasks. To this end, we consider pre-training on COCO+ and conduct an ablation study on the number of prototypes and the use of Sinkhorn-Knopp algorithm for patch-level loss. The results are shown in Tab. 6.

Increasing the number of prototypes C loosens the constraints on the compactness of the slots, and the use of SH encourages more diverse utilization of the prototypes, both leading to more fine-grained slots. The results show that both methods can improve the success rate on manipulation tasks (averaged over all tasks). Explorations in this direction may resolve the inverse-scaling phenomenon on non-ego-centric data in the main paper, which we leave for future work.

Model	Dataset	#Proto.	SH	Jacc	$\overline{K}$	Success (%)
SlotMIM	COCO+	256	✗	42.2	7.8	74.3
SlotMIM	COCO+	512	✗	**43.9**	9.4	74.8
SlotMIM	COCO+	512	✓	40.8	15.2	76.8
SlotMIM	COCO+	1024	✗	42.8	10.8	**78.4**

Table 6. **Can fine-grained slots improve manipulation performance?** We consider increasing the number of prototypes C and using SH for patch-level loss. Both ways enforce the emergence of fine-grained slots and improve success rate on manipulation tasks.

B.5. Limitation and Future Work

This work explores the interaction between pre-training data and algorithms for robotic manipulation, navigation, and perception-oriented tasks. With these of interest, it requires extremely intensive computation to provide a complete comparison of all existing PVMs, scale pre-training data to larger scales, and evaluate more complex robotic tasks (*e.g.*, language-conditioned manipulation and real-world tasks). The results present in the paper is thus a result of trading-off. Will SlotMIM work well for generalist robotic models like Octo [50] and OpenVLA [37]? Will the next-gen PVM for robotics be pre-trained on a mixture of SA-1B [38], Ego4D [24], and/or LAION-400M [60], LVD-142M [52]? Will the preceptive module bestly to be supervised by self-supervision, language, action trajectories, or a mixture of them? There are yet much to explore, but still, we believe that the insights and methods proposed in this work can serve as a stepping stone for future research in this direction.

C. Extended Experiments

C.1. Extended Ablation Study

In Tab. 5 we present ablations on some numeric design choices. Generally speaking, a smaller number of prototypes, a higher mask ratio, and the use of centering [13] instead of Sinkhorn-Knopp algorithm [12] encourage the network to discover more holistic concepts/objects, while the opposite discovers more fine-grained ones. Optimal representation is highly related to object discovery quality.

Method	Dataset	Scale	Kitchen	MW	ObjNav	ImgNav
MVP	EgoSoup	4.6M	49.7	70.1	51.2	64.7
VC-1	Ego4D+MNI	5.6M	58.1	73.3	55.4	67.9
DINOv2	LVD	142M	64.0	38.8	**65.8**	59.1
SlotMIM	Ego4D	1.28M	**86.0**	**84.2**	48.4	65.4
SlotMIM	DetSoup	4M	46.7	75.1	62.0	**69.8**

Table 7. **Comparison with DINOv2.** SlotMIM achieves comparable or better performance on robot tasks compared to DINOv2, especially on manipulation tasks. (DINOv2 is ViT-B/14, while other models are ViT-B/16)

C.2. Comparison with DINOv2

One might argue that the state-of-the-art self-supervised model, DINOv2 [52], already utilizes NOC data with a vision transformer backbone. However, its success heavily depends on data curation techniques that leverage the object-centric ImageNet dataset to select neighboring data from web-crawled data, keeping its data distribution closely tied to object-centric approaches. We also evaluate DINOv2 on the robot learning tasks considered in this paper. As shown in Tab. 7, DINOv2 does not perform as well on these tasks, possibly also due to over-compression of the representations for manipulation tasks.

C.3. ImageNet Linear Probing and Fine-tuning

Setting. We follow MAE [29] for details on ImageNet evaluations. For linear probing, we insert an extra BatchNorm layer without affine transformation between the features and the linear classifier. We train with batch size 4096, initial learning rate 0.1, and optimize using SGD for 90 epochs. We sweep between `[CLS]` token and average pooling and report the best results of pre-trained models. For fine-tuning, we train a linear classifier on frozen features for 100 epochs using SGD with momentum 0.9, batch size 1024, and initial learning rate 1e-3 with cosine decay. We follow MAE [29] to adopt average pooling. For both settings, accuracy is evaluated on a single 224×224 crop.

We first evaluate models pre-trained on 241K-scale datasets, and show that NOC data can be good learning resources if used properly. The results are present in Fig. 8. Overall, SlotMIM achieves the best performance across classification and segmentation tasks, no matter learning from object-centric data or not. Below, we discuss some other interesting findings.

Features learned from NOC data can be linear separatable on ImageNet. From Fig. 8 (left), our models trained on COCO and CC achieve similarly good linear probing performance on ImageNet with best prior ImageNet-trained methods. As a clear contrast, all previous methods trained on NOC datasets (COCO, CC, and Ego4D) fall behind the best ImageNet counterpart.

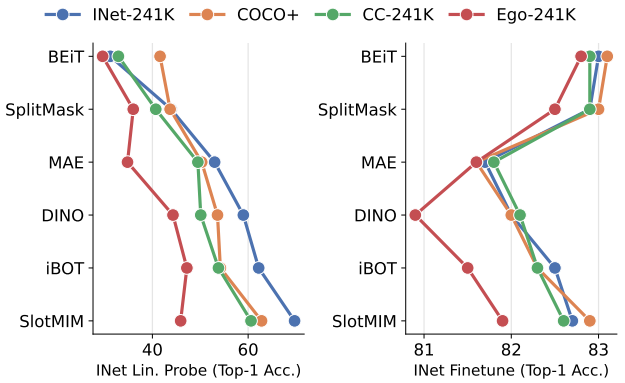

Figure 8. **Results on ImageNet tasks.** SlotMIM consistently outperforms prior arts whether pre-trained on object-centric data or not. Notably, when trained on COCO+, it transfers better than most ImageNet models despite the domain gap.

NOC data can be worth more than ImageNet for ImageNet. As shown in Fig. 8 (right), under ImageNet fine-tuning setting, the top-3 methods (BEiT, SplitMask, and SlotMIM) have the best performance when trained on COCO+ instead of ImageNet. For MAE and DINO, training on CC also transfers better than ImageNet. Note that this is uncommon given the domain gap between NOC pre-training data and OC downstream task, demonstrating that NOC data are information-rich learning resources.

C.4. Scaling Up for ImageNet Tasks

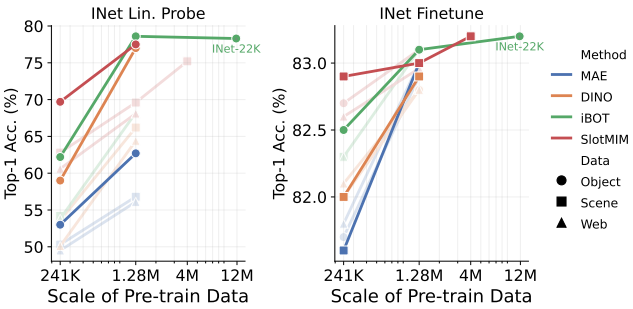

Figure 9. **Scaling on different data sources.** We scale up object-centric, scene-centric, and web-crawled data, and highlight the best (model, data) combinations. Our method learns strong and transferable representations with significant data efficiency and continues to improve with more data.

Superior data efficiency allows us to explore larger-scale pre-training data. In Fig. 9, we show that SlotMIM achieves strong performance with remarkable data efficiency.

Comparable or better performance with small data scale. As shown in Figure 9, SlotMIM achieves comparable or su-

perior performance to other methods using significantly less data. Our INet-241K model for ImageNet linear probing, and COCO+/INet-241K models for ImageNet fine-tuning outperform or match most models trained on 1.28M ImageNet images across various tasks. This remarkable data efficiency demonstrates our approach's effectiveness in extracting rich, transferable features from limited data.

NOC pre-training rivals ImageNet pre-training for ImageNet. Interestingly, we observe that pre-training on NOC datasets like OpenImages-1.28M can lead to performance better than pre-training on ImageNet for the ImageNet classification task (fine-tuning setting). When scaled up to 4M scale, this trend becomes more pronounced. This aligns with the trend in Fig. 8 that NOC data can provide more information-rich features, which can be better-utilized by models like SlotMIM.

NOC data also possesses stronger scalability. We extend experiments to 4M scale by combining ImageNet [20], COCO+ [41], OpenImages [39], Objects365 [63], and LVIS [25]. Compared with previous efforts on scaling up with ImageNet-22K [58] (12M images), the performance of SlotMIM models continues to grow and surpasses them with $3\times$ less data. This suggests that NOC data can be a more scalable learning resource.