Supplementary Material:
How Useful is Self-Supervised Pretraining for Visual Tasks?

1. Additional training details

We use the Ray Tune library [4, 3] for selecting hyperparameters. We tune the learning rate schedule and weight decay terms as well as task-specific parameters in the self-supervised settings. The pretrained models are given as much training time during finetuning as the model trained from scratch. Precise details and experiment configuration settings can all be found in the released code.

We apply standard data augmentation (cropping, flipping, brightness/color perturbations, Cutout [2]) in all settings with a few exceptions for specific tasks. For example, to avoid issues that would arise with padding and resizing for depth estimation we do not do random cropping.

One additional note, since Contrastive Multiview Coding (CMC) [5] uses Lab color space as input, we present all images across all methods and datasets in Lab color space for consistency. In ablations, we do not find this affects performance when training a supervised model from scratch with a large amount of labeled data, but want to control for as many differences as possible between methods.

2. Additional experiments

2.1. Cross-dataset transfer

A common concern with self-supervised pretraining is robustness to domain shifts. To get a sense of how shifts in dataset properties affect performance, we compare models pretrained and finetuned on different datasets. We evaluate downstream object classification performance in the low data regime. Results are shown in Figure 1.

Exposure to viewpoint changes appears to affect finetuning performance most. Accounting for all other image factors, models that are pretrained on datasets without viewpoint changes and finetuned on datasets with viewpoint changes suffer the most consistent drop in performance (upper right of Figure 1).

2.2. Tuning the pretraining process

Without a quick proxy for downstream performance, it is difficult to tune and improve the pretraining process. As seen in Figure 2, the downstream effect of more unlabeled images and more training time may be limited. We observe either no effect or a modest adverse effect when training for half the time or with an eighth of the unlabeled images. In practice, hyperparameter tuning is challenging as it is expensive to evaluate the impact, if any, of further adjustments to the pretraining process, and exponential increases in pretraining time and data might result in little to no change on the final task.

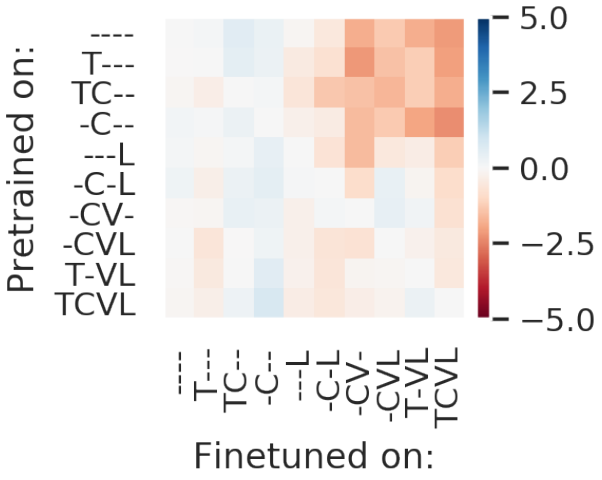

Figure 1. Change in performance when pretraining and finetuning on different dataset variations. Report average change in accuracy when finetuning on 250-4000 samples for models pretrained with AMDIM.

2.3. Network backbones

CMC: As noted in the paper, the backbone is different when pretraining with CMC as opposed to the other self-supervised methods. This is because the authors split the network in half, where each half is responsible for either processing the L channel or the ab channels of the image. This results in a restricted network with approximately half the original number of parameters. In all other ways, the network is identical.

This does lead to a modest change in performance in the single object settings (Figures 3). When evaluating utility in the paper we still measure relative to the original baseline of

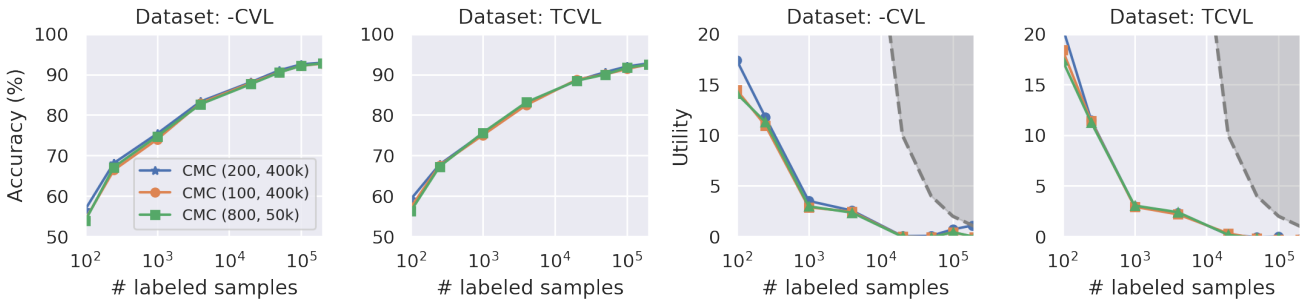

Figure 2. Change in utility with different pretraining settings for CMC. We label each pretrained model with (number of pretraining epochs, number of unlabeled images) and report downstream object classification performance. Because 50k unlabeled images is 1/8th the base 400k, the model is trained for more epochs to match total number of training iterations.

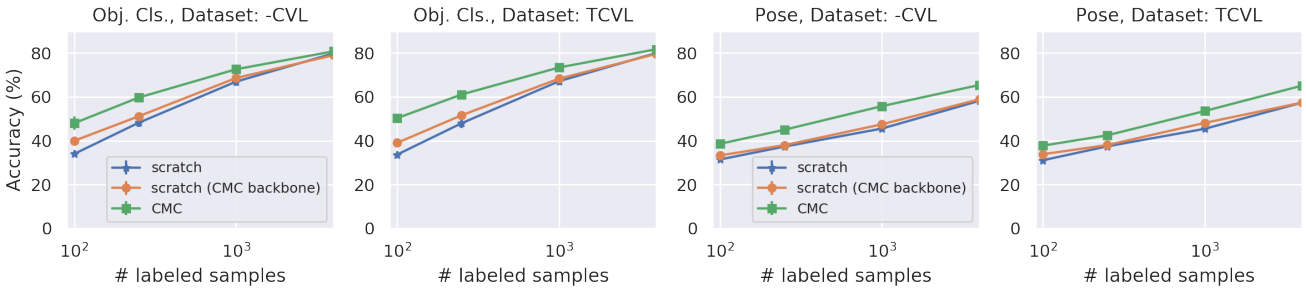

Figure 3. Training from scratch with CMC backbone (ResNet9 base) on object classification and object pose estimation.

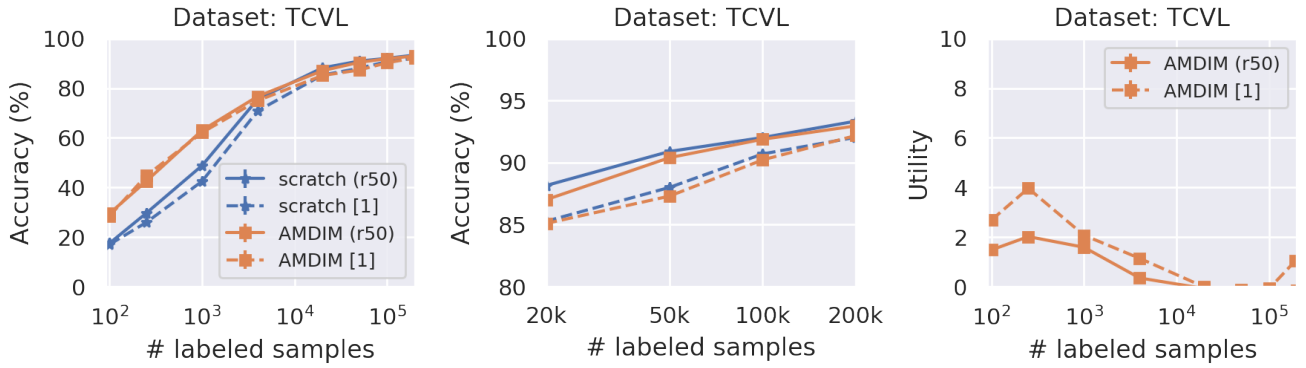

Figure 4. Comparison between ResNet50 backbone and backbone proposed in AMDIM paper [1]. Number of layers and number of channels are chosen to control for total parameters so that the two models are matched in size.

the full model trained from scratch. This does not change the conclusions of the paper that CMC provides the most utility in these settings at a low number of samples, and that the utility approaches zero as the number of labeled samples increases.

AMDIM: In the paper proposing AMDIM [1], a different backbone is used and recommended. We compare performance between this backbone and a ResNet50 model while trying to control for overall model size as much as possible (similar feature activation sizes and total parameter count). The linear performance of the proposed model is much better than that of the ResNet50, but when finetuned, the ResNet50 achieves higher accuracy. This is especially true at larger dataset sizes. For the purposes of the comparisons made in the main paper, we continued with use of the ResNet models for a fair comparison to other methods.

2.4. Results on additional datasets

In the following pages we include figures with results across more datasets for all tasks. Results are in line with those presented in the main paper. One observation is that different methods show stronger performance on the dense prediction tasks depending on the amount of data and model choice.

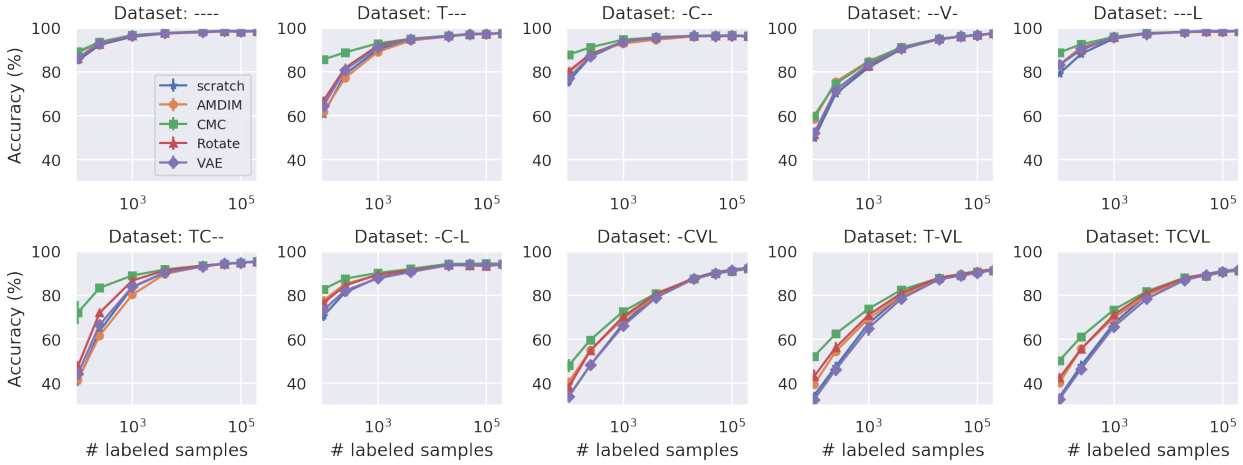

Figure 5. Object classification accuracy (ResNet9).

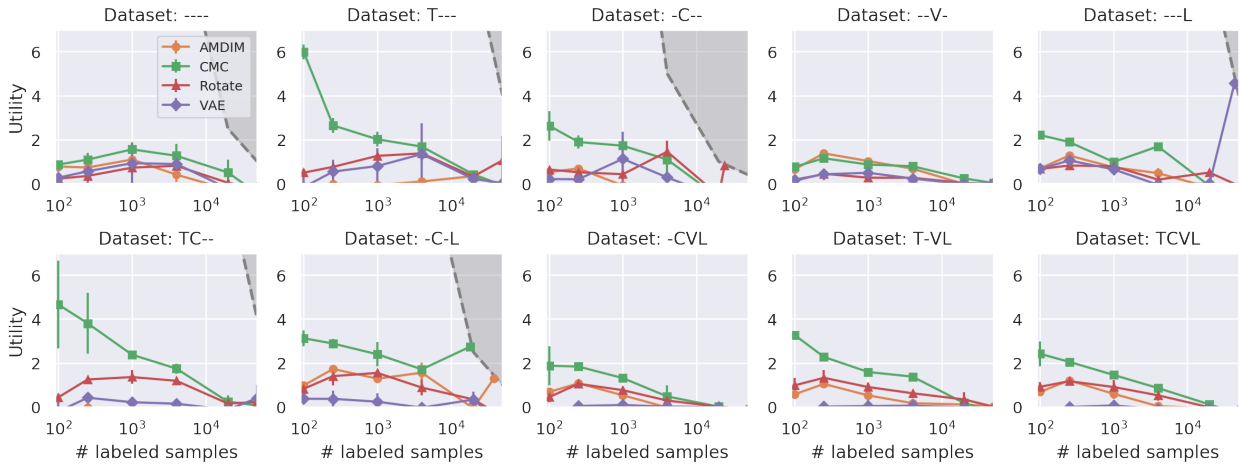

Figure 6. Object classification utility (ResNet9).

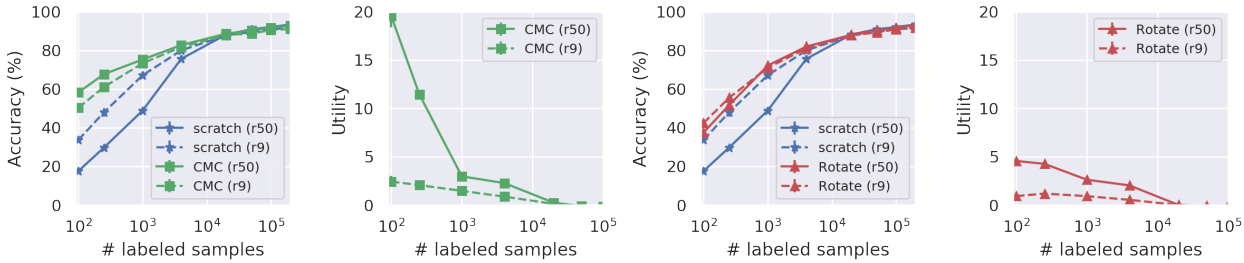

Figure 7. Object classification accuracy and utility (ResNet50).

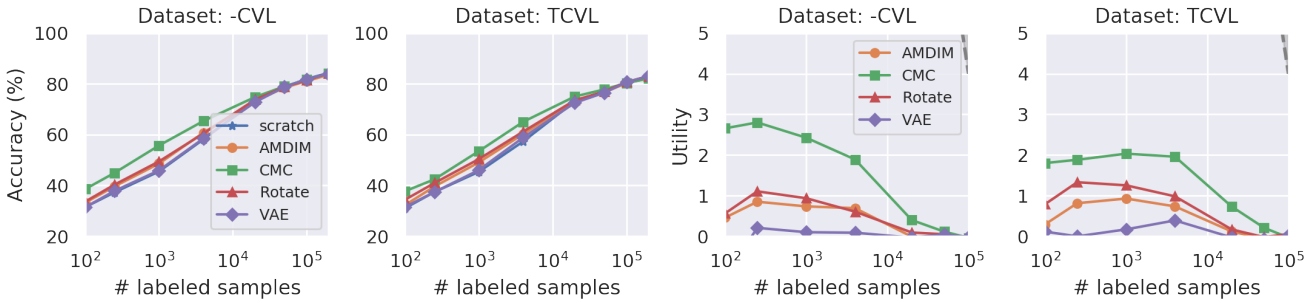

Figure 8. Object pose estimation results (ResNet9).

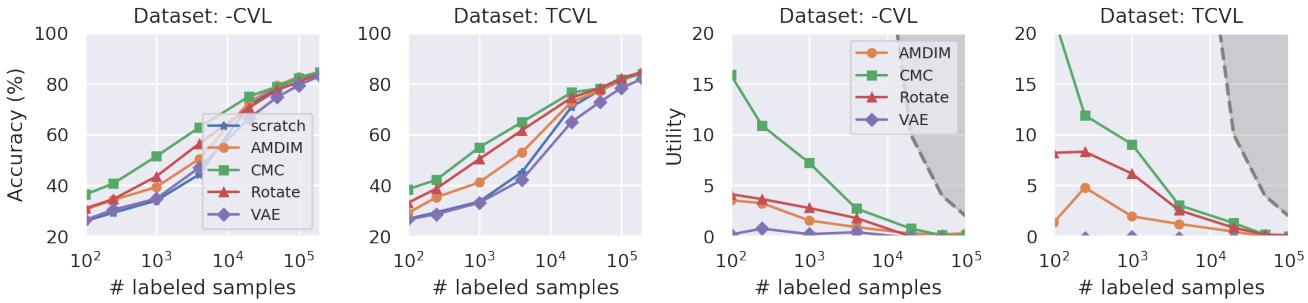

Figure 9. Object pose estimation results (ResNet50).

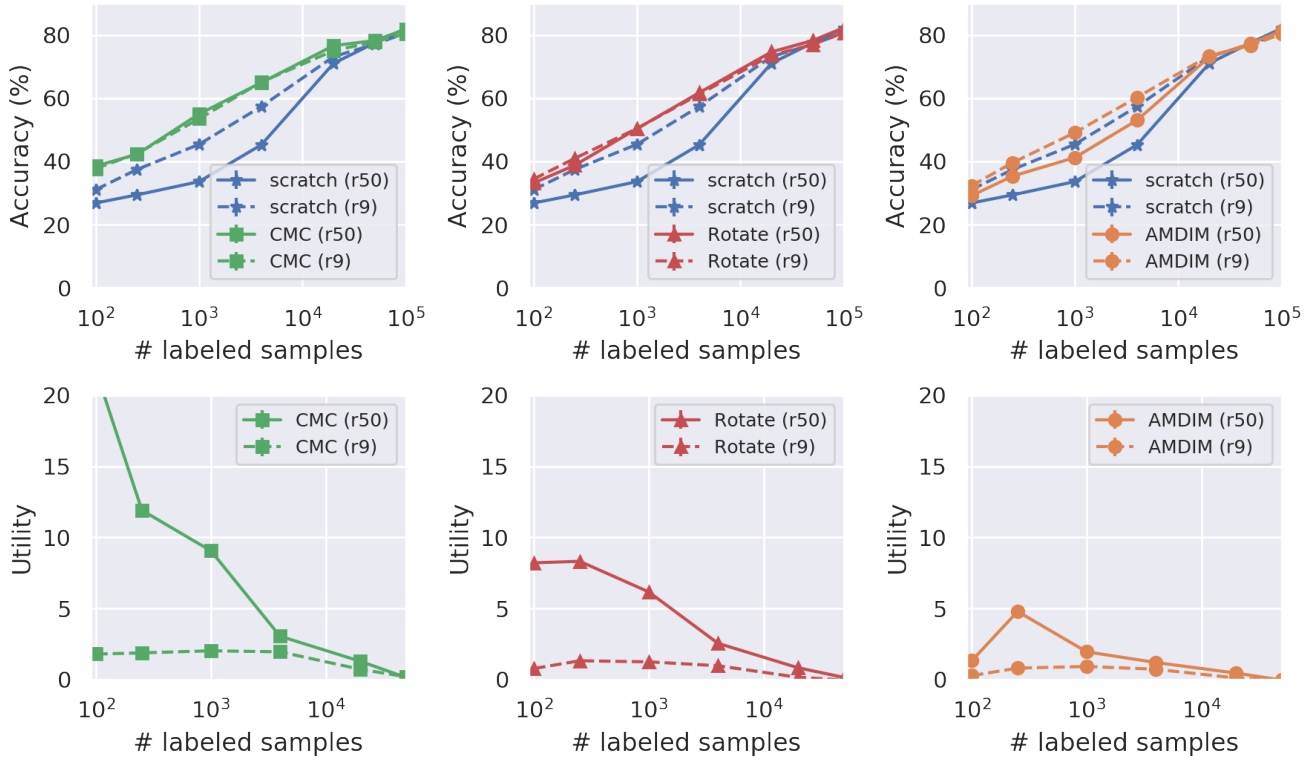

Figure 10. Direct comparison between ResNet9 and ResNet50 for object pose estimation.

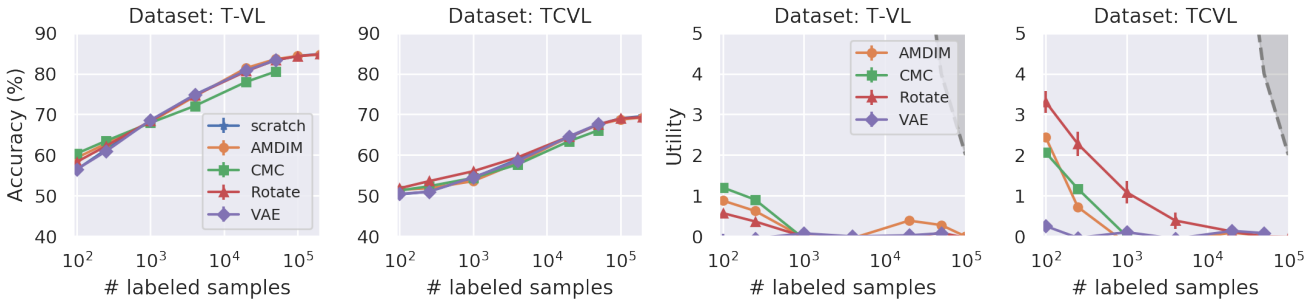

Figure 11. Semantic segmentation results (ResNet9).

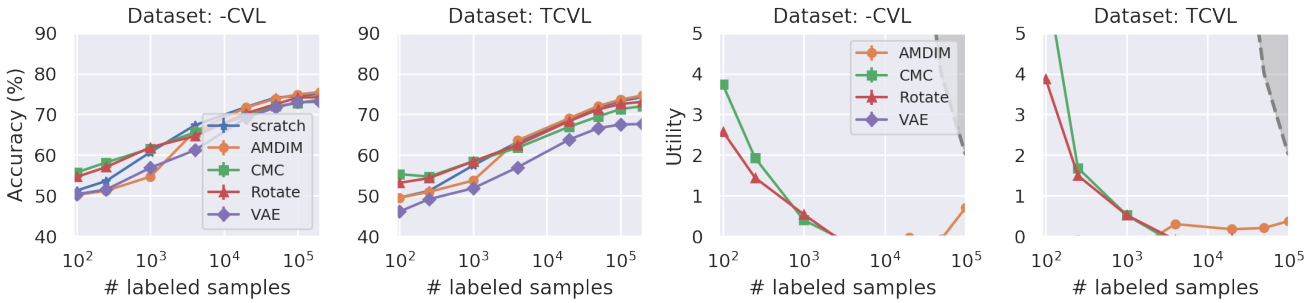

Figure 12. Semantic segmentation results (ResNet50).

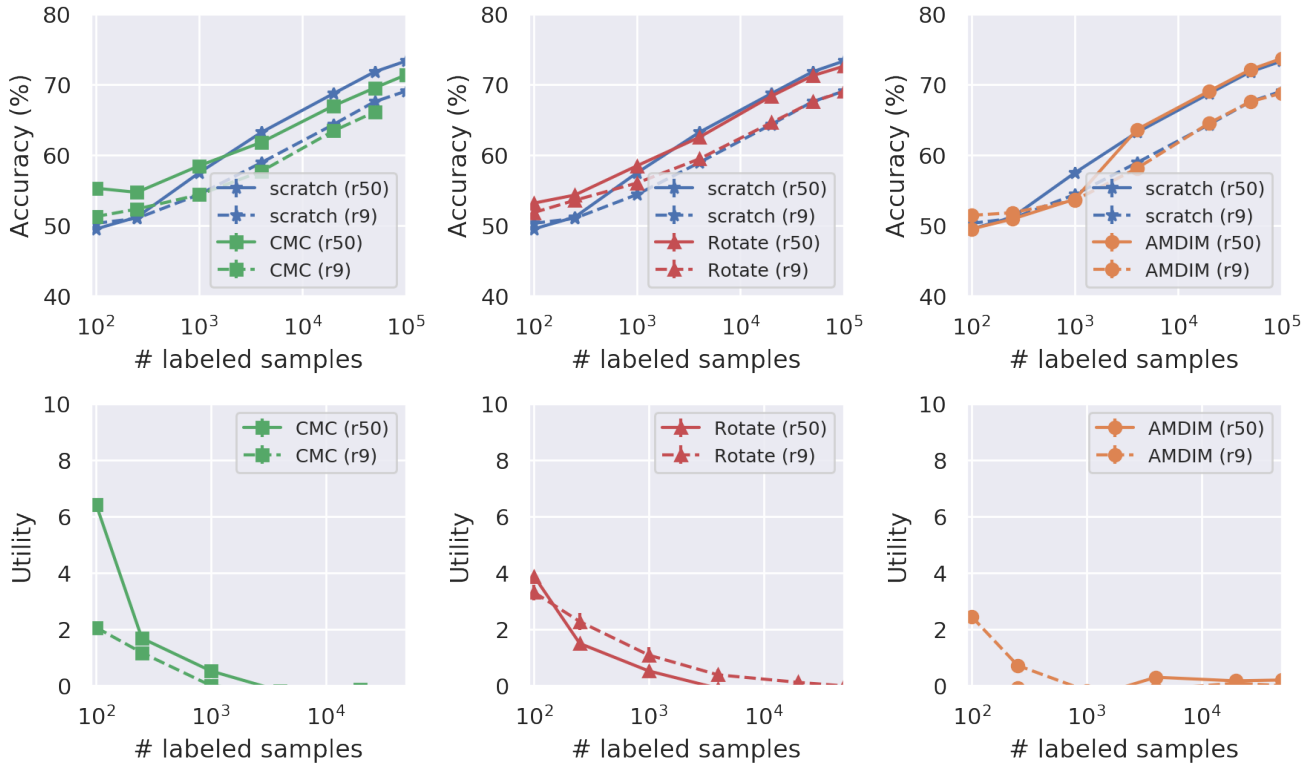

Figure 13. Direct comparison between ResNet9 and ResNet50 for semantic segmentation.

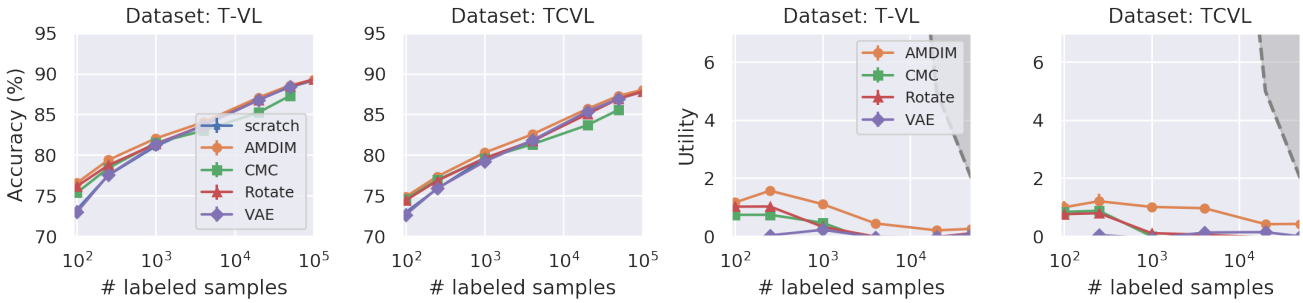

Figure 14. Depth estimation results (ResNet9).

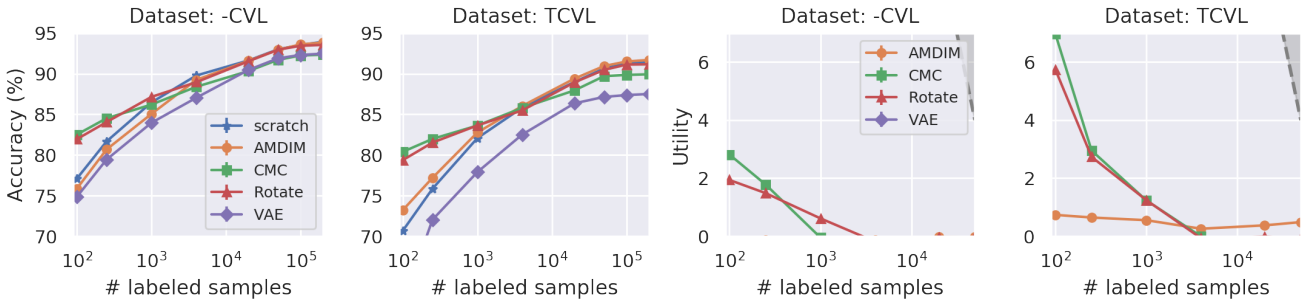

Figure 15. Depth estimation results (ResNet50).

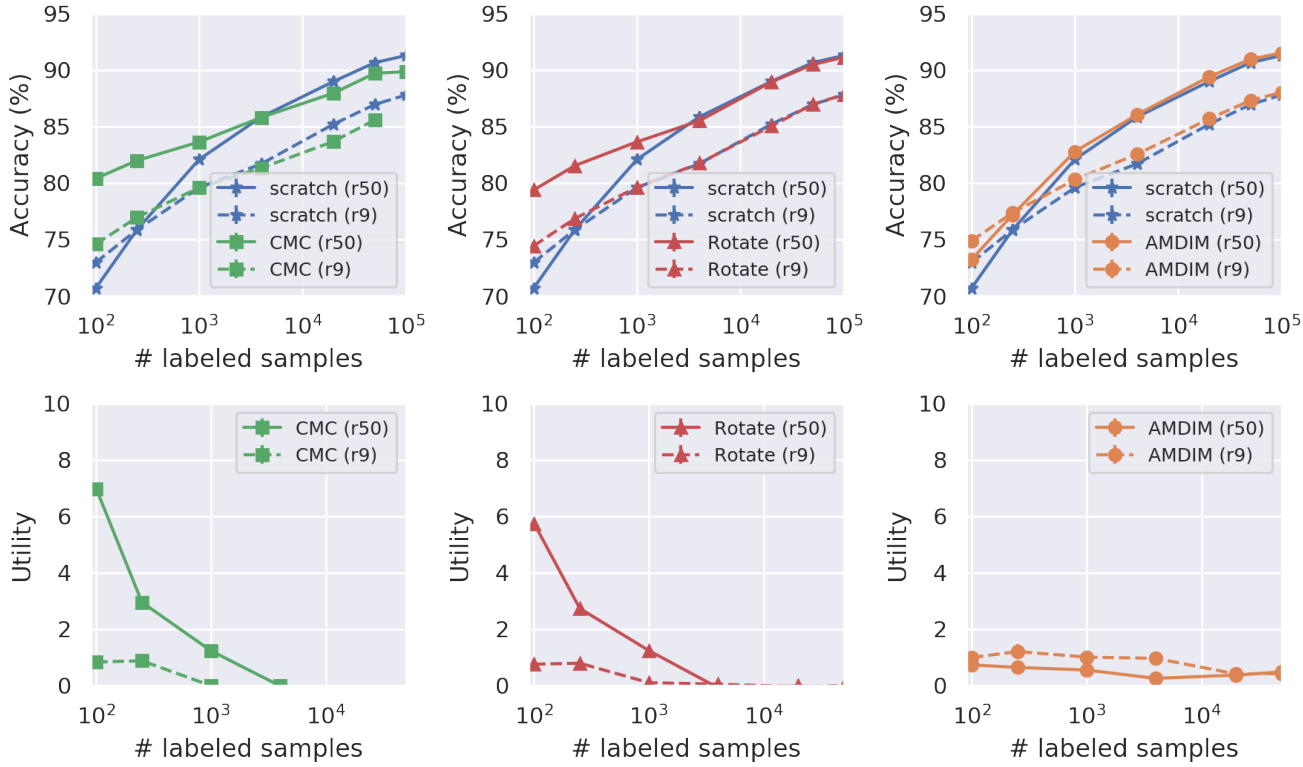

Figure 16. Direct comparison between ResNet9 and ResNet50 for depth estimation.

References

[1] Philip Bachman, R Devon Hjelm, and William Buchwalter. Learning representations by maximizing mutual information across views. *arXiv preprint arXiv:1906.00910*, 2019. 2

[2] Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017. 1

[3] Richard Liaw, Eric Liang, Robert Nishihara, Philipp Moritz, Joseph E Gonzalez, and Ion Stoica. Tune: A research platform for distributed model selection and training. *arXiv preprint arXiv:1807.05118*, 2018. 1

[4] Philipp Moritz, Robert Nishihara, Stephanie Wang, Alexey Tumanov, Richard Liaw, Eric Liang, Melih Elibol, Zongheng Yang, William Paul, Michael I Jordan, et al. Ray: A distributed framework for emerging {AI} applications. In *13th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 18)*, pages 561–577, 2018. 1

[5] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive multiview coding. *arXiv preprint arXiv:1906.05849*, 2019. 1