Self-Supervised Pretraining Improves Self-Supervised Pretraining Supplementary Material

Colorado J Reed[*1] Xiangyu Yue[*1] Ani Nrusimha[1] Sayna Ebrahimi[1]
Vivek Vijaykumar[3] Richard Mao[1] Bo Li[1] Shanghang Zhang[1] Devin Guillory[1]
Sean Metzger[1,2] Kurt Keutzer[1] Trevor Darrell[1]

[1]**UC Berkeley**, [2]**UCSF**, [3]**Georgia Tech**

[*]equal contribution, correspondence to cjrd@berkeley.edu

Appendix

A. Implementation details

Table 1 lists the parameters used in the various training stages of the HPT pipeline. When possible, we followed existing settings from [2]. For the finetuning parameter sweeps, we followed a similar setting as the "lightweight sweep" setting from [21]. We performed pretraining with the train and val splits. For evaluation, we used only the train split for training the evaluation task and then use the val split evaluation performance to select the top hyperparameter, training schedule, and evaluation point during the training. We then reported the performance on the test split evaluated with the best settings found with the val split.

For the linear analysis and finetuning experiments, we used `RandomResizedCrop` to 224 pixels and `RandomHorizontalFlip` augmentations (for more on these augmentations, see [2]) during training. During evaluation, we resized the long edge of the image to 256 pixels and used a center crop on the image. All images were normalized by their individual dataset's channel-wise mean and variance. For classification tasks, we used top-1 accuracy and for multi-label classification tasks we used the Area Under the ROC (AUROC) [1].

For the 1000-label semi-supervised finetuning experiments, we randomly selected 1000 examples from the training set to use for end-to-end finetuneing of all layers, where each class occurred at least once, but the classes were not balanced. Similar to [21], we used the original validation and test splits to improve evaluation consistency.

For all object detection experiments, we used the R50-C4 available in Detectron2 [17], where following [6], the backbone ends at conv4 and the box prediction head consists of conv5 using global pooling followed by an additional batchnorm layer. For PASCAL object detection experiments, we used the `train 2007+2012` split for training and the `val2012` split for evaluation. We used 24K training steps with a batch size of 16 and all hyperparameters the same as [6]. For BDD, we used the 70K BDD `train` split for training and 10K `val` split for evaluation. We used 90K training steps with a batch size of 8 on 4 GPUs. For CoCo object detection and segmentation, we used the 2017 splits, with the $1\times(\sim 12)$ epochs training schedule with a training batch size of 8 images over 180K iterations on 4 GPUs half of the default learning rate (note: many results in the literature (e.g. [6]) use a batch size of 16 images over 90K iterations on 8 GPUs with the full default learning rate, which leads to slightly improved results (+0.1-0.5 AP). For semantic segmentation, we used Mask-RCNN [7] with a C4 backbone setting as in [6].

Table 1. This table provides the parameters that were used for pretraining, linear, and finetuning analyses carried out in this paper (unless otherwise noted). Multiple values in curly braces indicate that all combinations of values were tested, i.e. in order to find an appropriate evaluation setting. 10x decay at $\frac{1}{3}$ and $\frac{2}{3}$ corresponds to decaying the learning rate by a factor of 10 after $\frac{1}{3}$ and $\frac{2}{3}$ of training steps have occurred, respectively.

Parameter	**MoCo-V2 Value**	**Linear Value**	**Finetune Value**
batch size	256	512	256
num gpus	4	4	4
lr	0.03	{0.3, 3, 30}	{0.001, 0.01}
schedule	cosine	10x decay at $\frac{1}{3}$, $\frac{2}{3}$	10x decay at $\frac{1}{3}$, $\frac{2}{3}$
optimizer	SGD	SGD	SGD
optimizer momentum	0.9	0.9	0.9
weight decay	1e-4	0.0	0.0
duration	800 epochs	5000 steps	{2500 steps, 90 epochs}
moco-dim	128	-	-
moco-k	65536	-	-
moco-m	0.999	-	-
moco-t	0.2	-	-

B. Datasets

Table 2 lists the datasets used throughout our experiments. For all evaluations, unless otherwise noted, we used top-1 accuracy for the single classification datasets and used the Area Under the ROC (AUROC) [1] for multi-label classification tasks.

Table 2. Dataset Descriptions. We use $x/y/z$ to denote train/val/test split in each dataset.

Dataset		Train/Validation/Test Size	Labels	Classification Type
BDD [20]		60K/10K/10K	6 classes	singular
Chest-X-ray-kids[10]		4186/1046/624	4 classes	singular
Chexpert [9]		178.7K/44.6K/234	5 classes	multi-class
Coco-2014 [12]		82.7K/20.2K/20.2K	80 classes	multi-class
Domain Net [14]	Clipart	27.2K/6.8K/14.8K	345 classes	singular
	Infograph	29.6K/7.4K/16.1K	345 classes	singular
	Painting	42.2K/10.5K/22.8K	345 classes	singular
	Quickdraw	96.6K/24.1K/51.7K	345 classes	singular
	Real	98K/24.5K/52.7K	345 classes	singular
	Sketch	39.2K/9.8K/21.2K	345 classes	singular
RESISC [3]		18.9K/6.3K/6.3K	45 classes	singular
VIPER [15]		13.3K/2.8K/4.9K	5 classes	singular
UC Merced [19]		1.2K/420/420	21 classes	singular
Pascal VOC [4]		13.2K/3.3K/4.9K	20 classes	multi-class
Flowers [13]		1K/1K/6.1K	103 classes	singular
xView [11]		39K/2.8K/2.8K	36 classes	multi-class

C. Additional Experiments and Ablations

C.1. HPT Learning Rate

We investigated the choice of the learning rate used during HPT and its effect on linear evaluation performance (see Table 3). Specifically, we tested initial learning rates $\{0.1, 0.03, \text{and } 0.001\}$, on datasets: RESISC, BDD, and Chexpert. The following table shows that the default learning rate of 0.03 for batch size 256 from [2] outperformed the other configurations. Based on this experiment, we used the default 0.03 learning rate for all HPT pretraining runs.

Table 3. The following table shows the linear evaluation performance with HPT pretraining learning rates of $\{0.1, 0.03, \text{and } 0.001\}$. Based on this experiment, we continues to use the default 0.03 learning rate from [2].

	Learning Rate		
Datasets	0.1	0.03	0.001
RESISC	92.5	**93.7**	91.6
BDD	81.9	**83.2**	82.4
Chexpert	79.5	**85.8**	83.9

C.2. HPT with Supervised Base Model

We explored using a supervised ImageNet base model from [8] instead of the self-supervised MoCo model from [2]. Similar to the expepriments shown in Figure **??** in the main paper, Figure 1 shows the same results using a supervised base ImageNet model. We observe similar behavior as with the self-supervised base model: HPT with the supervised base model tends to lead to improved results compared to directly transferring the base model or pretraining entirely on the target data. Unlike the self-supervised base model, HPT with a supervised base model often shows improved performance after 5K iterations, e.g. at 50K iterations (RESISC, BDD, DomainNet `Sketch`, DomainNet `Quickdraw`, xView, Chexpert, CoCo-2014), indicating that the supervised base model needs longer training to obtain comparable linear evaluation results. Also unlike the self-supervised base model, these results show clearly better Target model results for BDD and Chexpert, and a larger gap with DomainNet `Quickdraw`. This indicates that the supervised pretraining is less beneficial as a base model when the domain gap is large – an observation further supported by the experiments in [18].

In Figure 2, we show results similar to the finetuning experiment results displayed in Figure **??** in the main paper, we investigate the finetuning performance on the same set of target datasets except using a supervised ImageNet base model [8]. Overall, HPT again leads to improved performance over finetuning on the Target pretrained models for all three datasets.

Different from the self-supervised base model used in Figure **??**, all results for the DomainNet framework are considerably worse, and incorporating training on the source dataset (DomainNet `Clipart`) does not demonstrate an improvement in this case. Overall, however, HPT with the supervised base model leads to improved finetuning performance with and without the source training step.

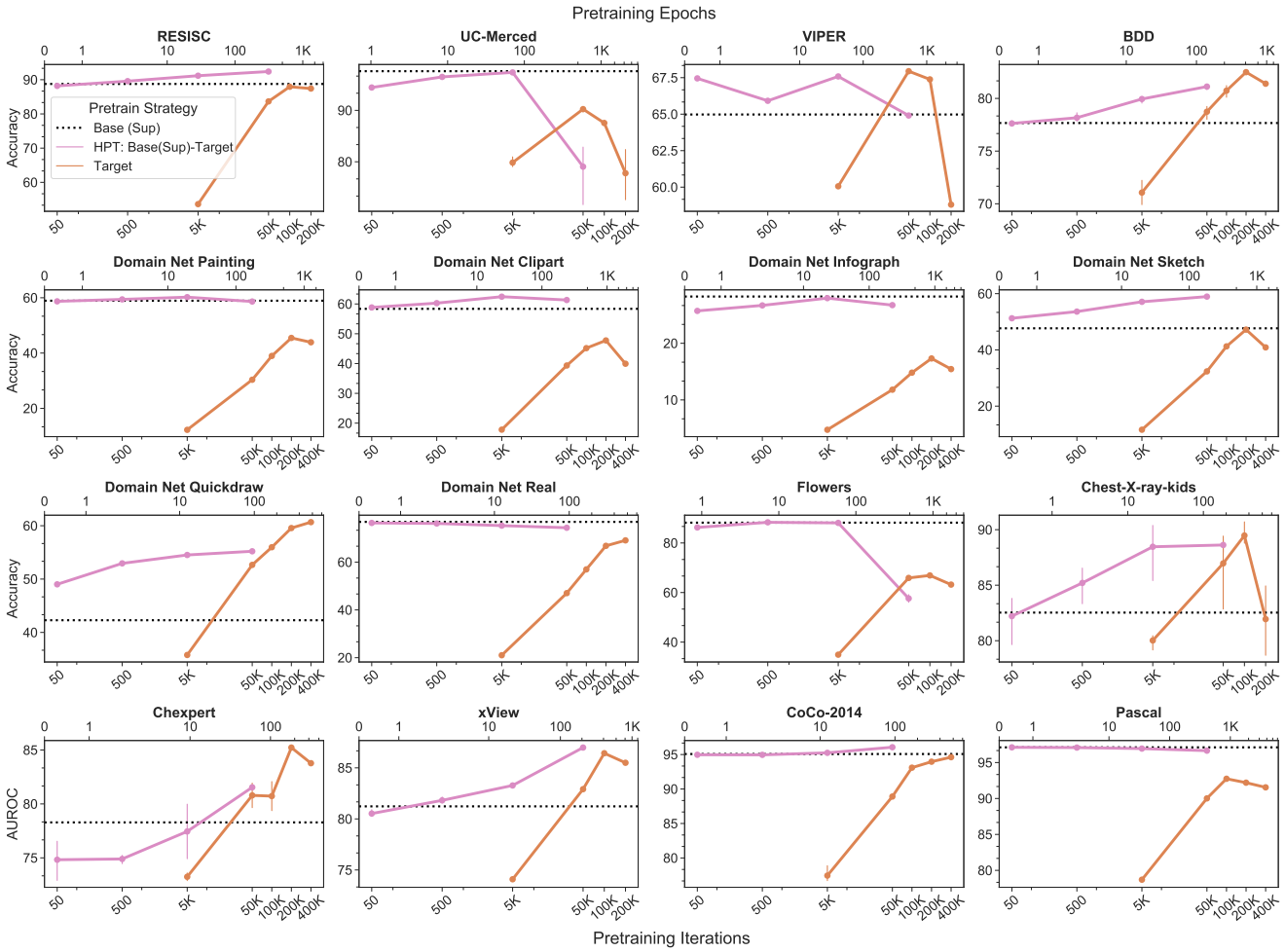

Figure 1. Linear eval: For each of the 16 datasets, we use a supervised ImageNet Base model [8]. We train the HPT framework for 50-50k iterations (HPT Base(sup)-Target). We compare it to a model trained from a random initialization (Target) trained from 5K-400K iterations. For each, we train a linear layer on top of the final representation. With a supervised base model, HPT obtains as good or better results on 13/16 datasets without hyperparameter tuning.

Augmentation robustness: We studied the augmentation robustness of HPT when using a supervised base model (HPT-sup). We followed the same experimental procedure described in Section 4.4. Figure 3 shows the results using HPT-sup, while for comparison, Figure **??** shows the results with HPT. Both HPT-sup and HPT demonstrate their robustness to the set of augmentations used while pretraining on RESISC. However, HPT-sup exhibits more variation to the augmentations used during pretraining on BDD and Chexpert. The supervised model was trained with only cropping and flipping augmentations, while the self-supervised pretraining took place with all augmentations in the shown policy. The robustness of the self-supervised base model indicates that the selection of the augmentation policy for further pretraining with the target dataset is resilient to changes in the set of augmentations used for pretraining the base model, and if these augmentations are not present, then the HPT framework loses its augmentation policy robustness.

C.3. Basetrain Robustness

We explored how the linear analysis evaluation changed with varying the amount of self-supervised pretraining on the base model, e.g. the initial ImageNet model. We tested base models trained for 20, 200, and 800 epochs, where the 200 and

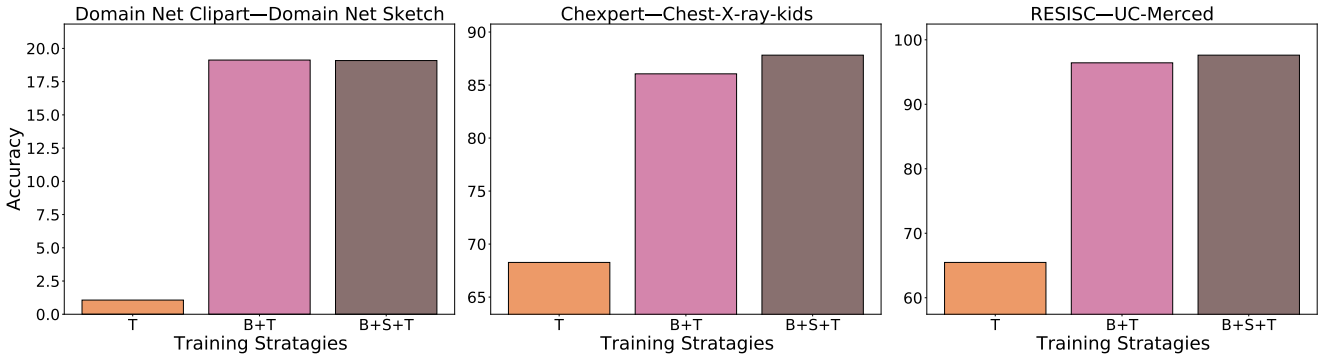

Figure 2. Finetuning performance on target datasets with supervised pretrainings. Here, we show results similar to the finetuning experiment results displayed in Figure **??**, we investigated the finetuning performance on the same set of target datasets except using a supervised ImageNet base model [8] and supervised (S)ource pretraining for B+S+T. Overall, HPT again leads to improved performance over finetuning on the Target pretrained models for all three datasets. Different from the self-supervised base model used in Figure **??**, all results for the Domain Net framework are considerably worse, and incorporating training on the source dataset (Domain Net Clipart) does not demonstrate an improvement in this case. Overall, however, HPT with the supervised base model leads to improved finetuning performance with and without the source training step.

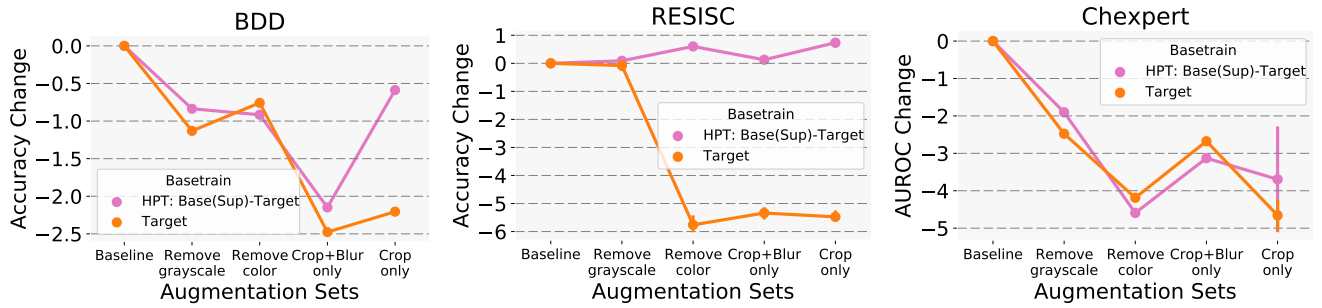

Figure 3. Supervised base model augmentation robustness. Here, we further studied the augmentation robustness of HPT when using a supervised base model (HPT-sup). We followed the same experimental procedure described in Section 4.4. As discussed in the text, these results show that HPT-sup exhibits more variation to the augmentations used during pretraining on BDD and Chexpert. As the supervised model was only trained with cropping and flipping augmentations, this indicates that the robustness from the base augmentations used in the self-supervised pretraining remain when performing further pretraining on the target dataset.

800 epoch models were downloaded from the research repository from [2][1], and the 20 epoch model was created using their provided code and exact training settings. For each base model, we performed further pretraining on the target dataset for 5000 steps.

Figure 4 shows the results for the RESISC, BDD, Chexpert, Chest-X-ray-kids, and DomainNet `Quickdraw` datasets. We note several characteristics of these plots: the 200 and 800 epoch datasets performed comparably across all datasets except Chest-X-ray-kids which displayed a drop in performance at 200 epochs, indicating that the extra self-supervised pretraining needed to obtain state-of-the-art linear ImageNet classification performance is typically not be necessary for strong HPT performance. Surprisingly, BDD, `Quickdraw`, and Chexpert show similar or improved performance at 20 epochs of basetraining. This indicates that even a relatively small amount of self-supervised pretraining at the base level improves transfer performance. Furthermore, as mentioned in §**??**, the `Quickdraw` dataset has a large domain gap with ImageNet, and indeed, we observe that directly transferring ImageNet models with less base training leads to improved results on `Quickdraw`, but HPT maintains consistent performance regardless of the amount of basetraining.

The computation needed for 20 epochs of basetraining + 5000 iterations of pretraining on the target dataset is approximately equal to 100,000 iterations of pretraining on only the target dataset. For all datasets except Chest-X-ray-kids, HPT at 20 epochs of basetraining exceeded the best Target-only pretraining performance, which was $\geq$ 100k iterations for all datasets. Indeed, for RESISC, the HPT results at 20 epochs are worse than 200 and 800 epochs, but they still exceed the best Target-only pretraining results (the dashed, orange line).

[1] `https://github.com/facebookresearch/moco`

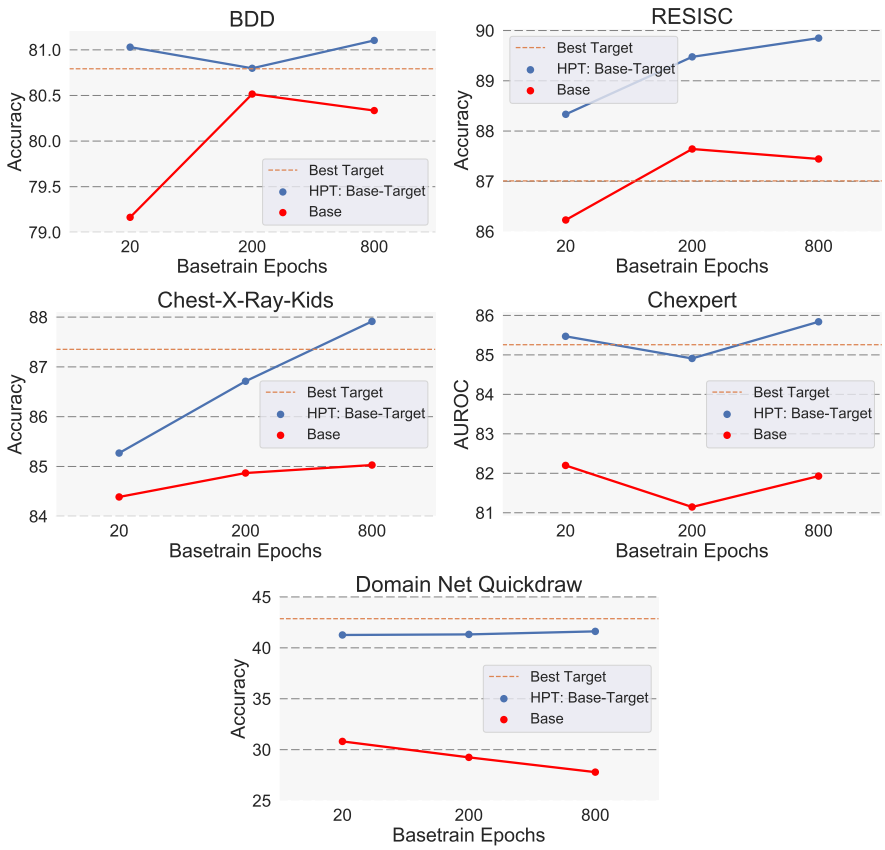

Figure 4. These figures show performance of linear evaluation on models pretrained on ImageNet for various epochs (in blue) and with addition HPT training (in red). The best baseline model pretrained only on target data for at least the equivalent of 20 ImageNet epochs and 5K HPT steps is show as the orange dotted-line.

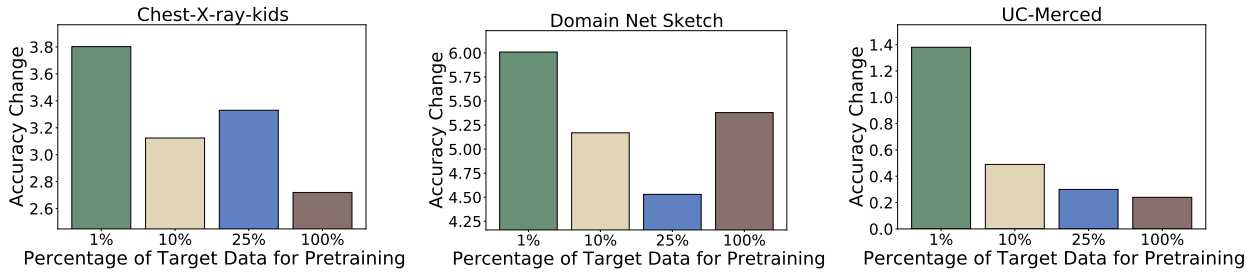

Figure 5. These figures show the change in the linear evaluation results by adding the source pretraining for each of the target data amounts for the given datasets. We observed that for all three datasets, adding the source pretraining had a larger benefit as the amount of target data was reduced. In other words, these results show that the impact of pretraining on an intermediate source dataset decreases as the size of the target dataset increases.

C.4. Source Pretraining

In this ablation, we investigated the impact of the source pretraining stage in the HPT framework as the amount of target data changes. Intuitively, we expected that the source pretraining stage would have less impact as the amount of target data increased. For this ablation, we used the three HPT frameworks studied in §**??**: ImageNet (base) then Chexpert (source) then Chest-X-ray-kids (target), ImageNet (base) then DomainNet `Clipart` (source) then DomainNet `Sketch` (target), and ImageNet (base) then RESISC (source) then UC-Merced (target). For each framework, we pretrained with {1%,10%,25%,100%} of the target data on top of the base+source model and on top of only the source model, before performing a linear evaluation with the target data.

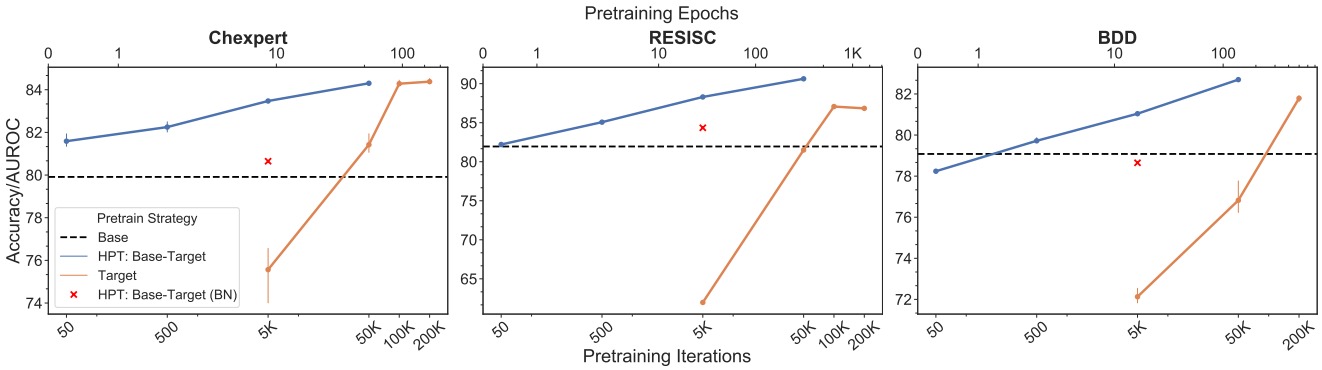

Figure 6. This figure shows the HPT linear evaluation performance on BDD, Chexpert, and RESISC using a ResNet-18. For these datasets, we observe similar behavior as with ResNet-50 though the evaluation performance is lower for all datasets and HPT shows improved performance at 50K iterations for all datasets. Generalizing from this experiment, we expect HPT to be broadly applicable across architectures, and we will report additional, ongoing results in our online code repository.

Figure 5 shows the change in the linear evaluation results by adding the source pretraining for each of the target data amounts. We observed that for all three datasets, adding the source pretraining had a larger benefit as the amount of target data was reduced. In other words, these results show that the impact of pretraining on an intermediate source dataset decreases as the size of the target dataset increases.

C.5. ResNet-18 Experiments

Similar to Figure **??**, Figure 6 shows the same results using a ResNet-18 on BDD, Chexpert, and RESISC. For these datasets, we observe similar behavior as with ResNet-50, though the evaluation performance is lower. Generalizing from this experiment, we expect HPT to be broadly applicable across architectures, and we will report additional, community results in our online code repository.

C.6. BYOL Experiments

All of the pretraining results in the main paper are based on MoCo, here we use BYOL [5] for pretraining and perform linear evaluation on RESISC, BDD and Chexpert. As shown in Figure 7, we observe similar results as Figure **??** in the main paper. Generalizing from this experiment, we expect HPT to be broadly applicable across different Self-supervised pretraining methods, and we will report additional, community results in our online code repository.

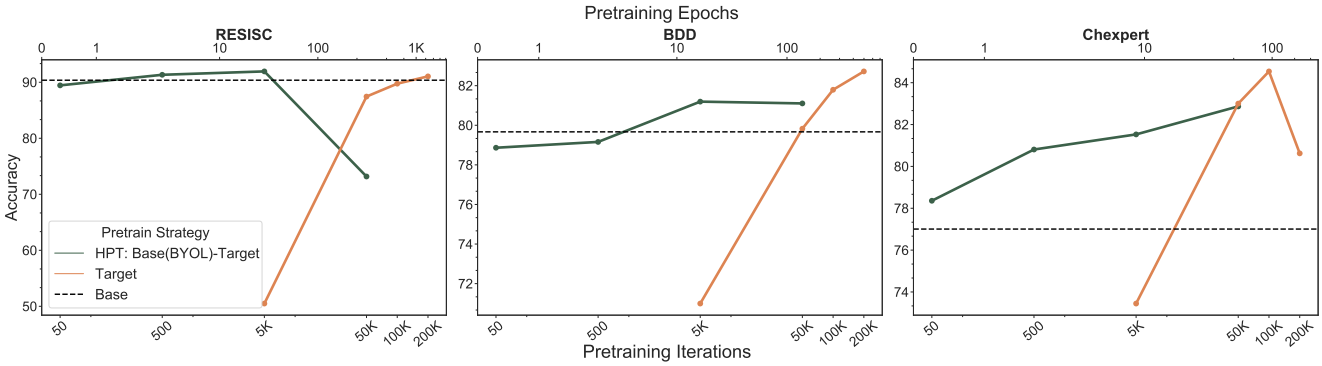

Figure 7. Linear eval with BYOL [5] pretraining on RESISC, BDD and Chexpert. For each dataset, we we train a generalist model for 200 epochs on ImageNet (Base). We then train the whole model from 50-50K iterations (HPT: Base (BYOL)-Target). We compare the HPT model with a model trained from a random initialization on the target data (Target). We use a linear evaluation to evaluate the quality of the learned representations.

C.7. Representational Similarity

We examine how the similarity of representations change during pretraining with the self-supervised base model HPT, supervised base model HPT, and self-supervised training on only the target data.

C.7.1 Defining metrics

We explore different metrics for measuring the similarity of two different pretrained models. The first metric we explore is the Intersection over Union (IoU) of misclassified images. The IoU is the ratio between the number of images misclassified by both models and the number of images misclassified by at least one model.

Algorithm 1 IoU

Require: data, labels, modelA, modelB
predictionA = modelA(data)
predictionB = modelB(data)
commonErrors, totalErrors $\leftarrow 0$
for all $l, pA, pB \in$ zip(labels, predictionA, predictionB) **do**
  if $l == pA$ **and** $l == pB$ **then**
    commonErrors+=1
  end if
  if $l == pA$ **or** $l == pB$ **then**
    totalErrors+=1
  end if
end for
return: $\frac{\text{commonErrors}}{\text{totalErrors}}$

The activation similarity metric we used wass RV2 [16] which, instead of comparing predictions, aims to compare the similarity between two different layers' outputs computed on the same data. The pseudocode for our algorithm is shown in Algorithm 2.

Algorithm 2 RV2

Require: activation A, activation B {Both activations are size $n \times p$, where n is the number of data points, and p is the size of the layer output}
A' = AA^T
B' = BB^T
$A'' = A' - diag(A')$
$B'' = B' - diag(B')$
return

$$\frac{\text{tr}(A''B''^T)}{\sqrt{\text{tr}(A''A''^T) * \text{tr}(B''B''^T)}}$$

Because many of our evaluations were performed by finetuning a linear layer over the outputs of the final convolutional layer of a pretrained model, we evaluated activations for the final convolutional layer and the linear layer finetuned on top of it.

For this analysis, we studied Resisc, UC Merced, Domain Net, Chexpert, Chest-X-ray-kids, and BDD datasets.

C.7.2 Effect of model initialization?

In this section, we present the results of two sets of experiments intended to analyze the representation of HPT models initialized from different base models. Overall, we found that HPT models with different base models learn different representations.

Random effects: In order to examine the effect of the random seed on model errors, we trained each combination of (basetrain, pretrain steps) five times for the RESISC dataset.

Representations which share the same basetrain and number of pretrain steps result in more similar errors and have much more similar representations than other combinations, see Figure 8. The IoU is typically between 0.5 and 0.6, meaning that roughly half of the total error caused by mispredictions are unique between these runs. The similarity is generally much higher, but is varies depending on the dataset.

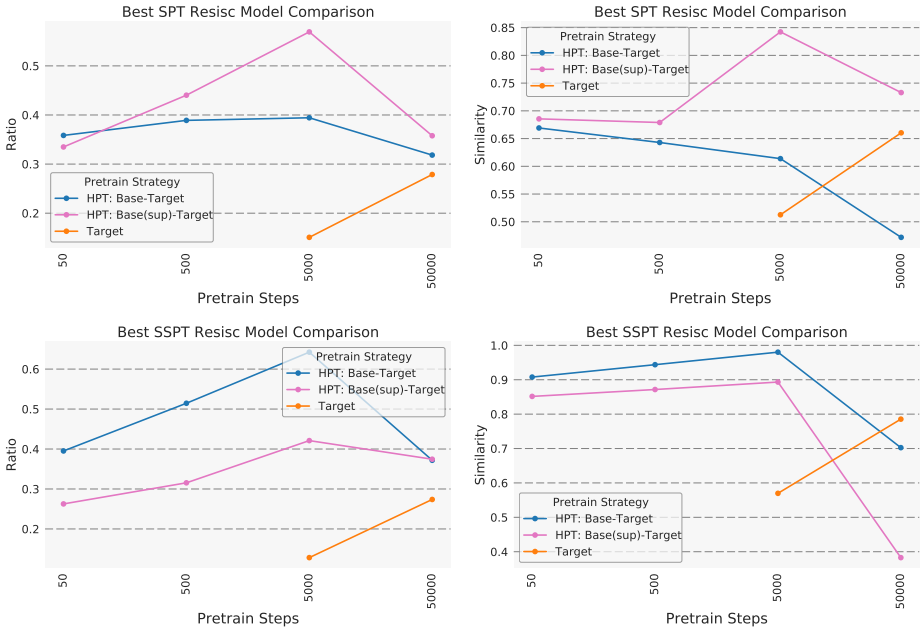

Figure 8. Supervised and semi-supervised models trained with random seeds. Left IoU, right final linear layer similarity. Top supervised, bottom semi-supervised

Same basetrain: Out of three different basetrain configurations (random-initialized, supervised basetrain, and self-supervised basetrain), different runs starting from the same basetrain are typically more similar to each other by both metrics than those with different basetrains. This holds true across models trained for different number of iterations with different random seeds, and further adds to the notion that models learn different representations based on their starting point. However, after overfitting, models are less likely to follow this trend and become dissimilar to ones with the same basetrain. We believe this is most likely due to models learning representations that mainly distinguish input data and thus all become dissimilar to the best performing model.

We determined how similar models were using our two metrics of layer similarity and IoU errors of models with different basetrains and number of pretrain steps compared to the best performing model. All models used the same pretrain dataset and target dataset for evaluation. We focused on similarity to the highest performing model (in terms of both basetrain and training iterations) to see if different basetrains converged to the same representation.

Linear classification layers from the same basetrain are consistently more similar than those with different basetrains. This trend becomes less consistent after around 50,000 iterations of training, which is also when the self-supervised models we examined start overfitting. In Figure 9 we plot the similarity of linear layers for each model relative to the best performing models on four sample datasets.

This same observation held when comparing the similarity of the final convolutional layers instead of the linear classification layers as shown in Figure 10. Overall, the convolutional layers trained from the same basetrain were more similar to each other than other basetrains. There were just a few points of comparison that deviated from the trend in a little under half of the datasets we tested.

The IoU error comparisons in Figure 11 showed a similar trend to the linear layers, with models with the same basetrain being more similar on almost all random seeds and datasets until 50,000 iterations.

Finally, we performed a significance test to demonstrate the significant difference between representations learned from different basetrain models. We trained five pairs of models with identical hyperparameters, but with different basetrains (supervised vs self-supervised) and a different random seed. We also trained five pairs of models with identical hyperparameters, but with the same basetrain (self-supervised) and a different random seed. All models used ImageNet for the basetrain and

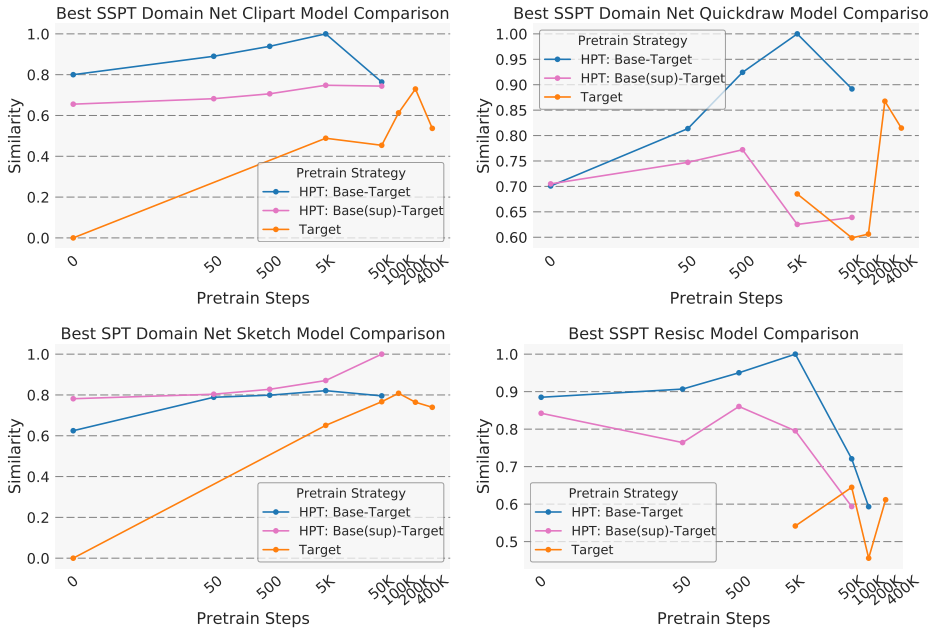

Figure 9. Linear Classification Layer comparison. For all the following graphs, SPT refers to Supervised Pretrain with ImageNet, and SSPT refers to Self-Supervised Pretrain with MoCo. The best model is the model that attains 1.0 similarity, and every other model is compared to that point. Up until the target model's pretrain steps (before overfitting), we can see that the similarity between the linear classification layers of every model with a different basetrain is much less than the models with the same basetrains.

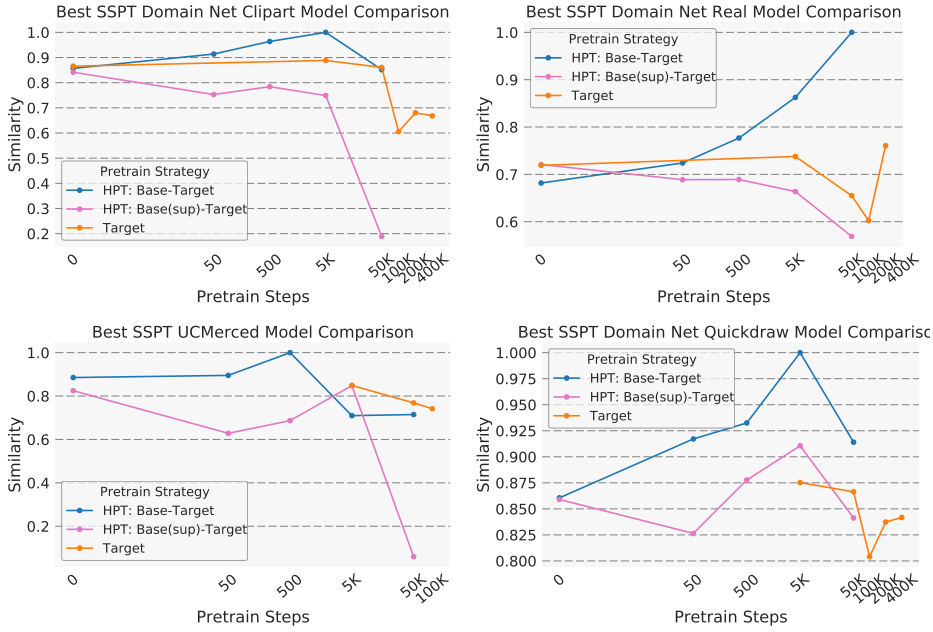

Figure 10. Final Convolutional Layer comparison. We can see that the models with basetrains similar to the best model consistently have higher similarity scores.

RESISC for the pretrain. We calculated the linear layer similarity and IoU for each pair of models, and performed a Welsh's t-test on the results. We found that the similarities and IoUs were significantly different. The different basetrains had a mean similarity of 0.78 while the identical basetrains had a mean similarity of 0.98 ($p = 2 \times 10^{-4}$). The different basetrains had a mean IoU of 0.40 while the identical basetrains had a mean IoU of 0.61 ($p = 2 \times 10^{-6}$).

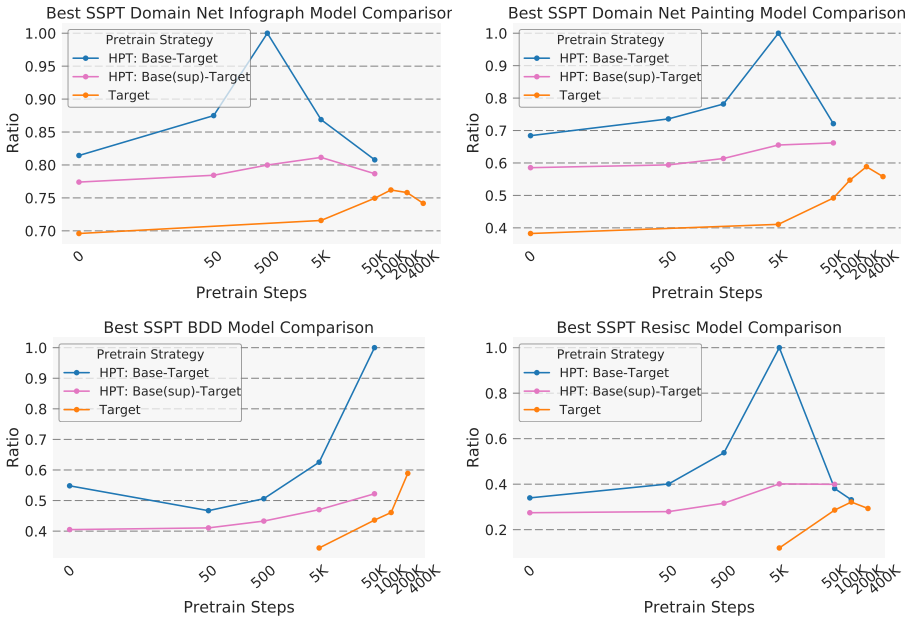

Figure 11. IoU comparison. IoU scores are consistently larger for models with the same basetrain as the best model compared to those of different basetrains, indicating that more similar errors are made by models with a similar initialization.

References

[1] Andrew P Bradley. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern recognition*, 30(7):1145–1159, 1997.

[2] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved Baselines with Momentum Contrastive Learning. *arXiv:2003.04297 [cs]*, Mar. 2020. arXiv: 2003.04297.

[3] Gong Cheng, Junwei Han, and Xiaoqiang Lu. Remote sensing image scene classification: Benchmark and state of the art. *Proceedings of the IEEE*, 105(10):1865–1883, 2017.

[4] Mark Everingham, SM Ali Eslami, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes challenge: A retrospective. *International journal of computer vision*, 111(1):98–136, 2015.

[5] 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, 2020.

[6] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[7] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[9] Jeremy Irvin, Pranav Rajpurkar, Michael Ko, Yifan Yu, Silviana Ciurea-Ilcus, Chris Chute, Henrik Marklund, Behzad Haghgoo, Robyn Ball, Katie Shpanskaya, et al. Chexpert: A large chest radiograph dataset with uncertainty labels and expert comparison. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 590–597, 2019.

[10] Daniel Kermany, Kang Zhang, and Michael Goldbaum. Large dataset of labeled optical coherence tomography (oct) and chest x-ray images. *Mendeley Data, v3 http://dx. doi. org/10.17632/rscbjbr9sj*, 3, 2018.

[11] Darius Lam, Richard Kuzma, Kevin McGee, Samuel Dooley, Michael Laielli, Matthew Klaric, Yaroslav Bulatov, and Brendan McCord. xview: Objects in context in overhead imagery. *arXiv preprint arXiv:1802.07856*, 2018.

[12] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[13] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing*, pages 722–729. IEEE, 2008.

[14] Xingchao Peng, Qinxun Bai, Xide Xia, Zijun Huang, Kate Saenko, and Bo Wang. Moment matching for multi-source domain adaptation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1406–1415, 2019.

[15] Stephan R Richter, Zeeshan Hayder, and Vladlen Koltun. Playing for benchmarks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2213–2222, 2017.

[16] Jessica A. F. Thompson, Yoshua Bengio, and Marc Schönwiesner. The effect of task and training on intermediate representations in convolutional neural networks revealed with modified RV similarity analysis. *CoRR*, abs/1912.02260, 2019.

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

[18] Xingyi Yang, Xuehai He, Yuxiao Liang, Yue Yang, Shanghang Zhang, and Pengtao Xie. Transfer learning or self-supervised learning? a tale of two pretraining paradigms. *arXiv preprint arXiv:2007.04234*, 2020.

[19] Yi Yang and Shawn Newsam. Bag-of-visual-words and spatial extensions for land-use classification. In *Proceedings of the 18th SIGSPATIAL international conference on advances in geographic information systems*, pages 270–279, 2010.

[20] Fisher Yu, Haofeng Chen, Xin Wang, Wenqi Xian, Yingying Chen, Fangchen Liu, Vashisht Madhavan, and Trevor Darrell. Bdd100k: A diverse driving dataset for heterogeneous multitask learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2636–2645, 2020.

[21] Xiaohua Zhai, Joan Puigcerver, Alexander Kolesnikov, Pierre Ruyssen, Carlos Riquelme, Mario Lucic, Josip Djolonga, Andre Susano Pinto, Maxim Neumann, Alexey Dosovitskiy, et al. A large-scale study of representation learning with the visual task adaptation benchmark. *arXiv preprint arXiv:1910.04867*, 2019.