Rethinking Pre-training and Self-training

Barret Zoph,* **Golnaz Ghiasi**,* **Tsung-Yi Lin**,*
Yin Cui, **Hanxiao Liu**, **Ekin D. Cubuk**, **Quoc V. Le**
Google Research, Brain Team
{barretzoph,golnazg,tsungyi,yincui,hanxiaol,cubuk,qvl}@google.com

Abstract

Pre-training is a dominant paradigm in computer vision. For example, supervised ImageNet pre-training is commonly used to initialize the backbones of object detection and segmentation models. He et al. [1], for example, show a contrasting result that ImageNet pre-training has limited impact on COCO object detection. Here we investigate self-training as another method to utilize additional data on the same setup and contrast it against ImageNet pre-training. Our study reveals the generality and flexibility of self-training with three additional insights: 1) stronger data augmentation and more labeled data further diminish the value of pre-training, 2) unlike pre-training, self-training is always helpful when using stronger data augmentation, in both low-data and high-data regimes, and 3) in the case that pre-training is helpful, self-training improves upon pre-training. For example, on the COCO object detection dataset, pre-training benefits when we use one fifth of the labeled data, and *hurts* accuracy when we use all labeled data. Self-training, on the other hand, shows positive improvements from +1.3 to +3.4AP across all dataset sizes. In other words, self-training works well exactly on the same setup that pre-training does not work (using ImageNet to help COCO). On the PASCAL segmentation dataset, which is a much smaller dataset than COCO, though pre-training does help significantly, self-training improves upon the pre-trained model. On COCO object detection, we achieve 54.3AP, an improvement of +1.5AP over the strongest SpineNet model. On PASCAL segmentation, we achieve 90.5 mIOU, an improvement of +1.5% mIOU over the previous state-of-the-art result by DeepLabv3+.[1]

1 Introduction

Pre-training is a dominant paradigm in computer vision. As many vision tasks are related, it is expected a model, pre-trained on one dataset, to help another. It is now common practice to pre-train the backbones of object detection and segmentation models on ImageNet classification [2–5]. This practice has been recently challenged He et al. [1], among others [6, 7], who show a surprising result that such ImageNet pre-training does not improve accuracy on the COCO dataset.

A stark contrast to pre-training is self-training [8–10]. Let's suppose we want to use ImageNet to help COCO object detection; under self-training, we will first discard the labels on ImageNet. We then train an object detection model on COCO, and use it to generate pseudo labels on ImageNet. The pseudo-labeled ImageNet and labeled COCO data are then combined to train a new model. The recent successes of self-training [11–14] raise the question to what degree does self-training work better than pre-training. Can self-training work well on the exact setup, using ImageNet to improve COCO, where pre-training fails?

*Authors contributed equally.

[1]Code and checkpoints for our models are available at https://github.com/tensorflow/tpu/tree/master/models/official/detection/projects/self_training

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

Our work studies self-training with a focus on answering the above question. We define a set of control experiments where we use ImageNet as additional data with the goal of improving COCO. We vary the amount of labeled data in COCO and the strength of data augmentation as control factors. Our experiments show that as we increase the strength of data augmentation or the amount of labeled data, the value of pre-training diminishes. In fact, with our strongest data augmentation, pre-training significantly *hurts* accuracy by -1.0AP, a surprising result that was not seen by He et al. [1]. Our experiments then show that self-training interacts well with data augmentations: stronger data augmentation not only doesn't hurt self-training, but also helps it. Under the same data augmentation, self-training yields positive +1.3AP improvements using the same ImageNet dataset. This is another striking result because it shows self-training works well exactly on the setup that pre-training fails. These two results provide a positive answer to the above question.

An increasingly popular pre-training method is self-supervised learning. Self-supervised learning methods pre-train on a dataset without using labels with the hope to build more universal representations that work across a wider variety of tasks and datasets. We study ImageNet models pre-trained using a state-of-the-art self-supervised learning technique and compare to standard supervised ImageNet pre-training on COCO. We find that self-supervised pre-trained models using SimCLR [15] have similar performance as supervised ImageNet pre-training. Both methods *hurt* COCO performance in the high data/strong augmentation setting, when self-training helps. Our results suggest that both supervised and self-supervised pre-training methods fail to scale as the labeled dataset size grows, while self-training is still useful.

Our work however does not dismiss the use of pre-training in computer vision. Fine-tuning a pre-trained model is faster than training from scratch and self-training in our experiments. The speedup ranges from 1.3x to 8x depending on the pre-trained model quality, strength of data augmentation, and dataset size. Pre-training can also benefit applications where collecting sufficient labeled data is difficult. In such scenarios, pre-training works well; but self-training also benefits models with and without pre-training. For example, our experiment with PASCAL segmentation dataset shows that ImageNet pre-training improves accuracy, but self-training provides an additional +1.3% mIOU boost on top of pre-training. The fact that the benefit of pre-training does not cancel out the gain by self-training, even when utilizing the same dataset, suggests the generality of self-training.

Taking a step further, we explore the limits of self-training on COCO and PASCAL datasets, thereby demonstrating the method's flexibility. We perform self-training on COCO dataset with Open Images dataset as the source of unlabeled data, and RetinaNet [16] with SpineNet [17] as the object detector. This combination achieves 54.3AP on the COCO test set, which is +1.5AP better than the strongest SpineNet model. On segmentation, we use PASCAL `aug` set [18] as the source of unlabeled data, and NAS-FPN [19] with EfficientNet-L2 [12] as the segmentation model. This combination achieves 90.5AP on the PASCAL VOC 2012 test set, which surpasses the state-of-the-art accuracy of 89.0AP [20], who also use additional 300M labeled images. These results confirm another benefit of self-training: it's very flexible about unlabeled data sources, model architectures and computer vision tasks.

2 Related Work

Pre-training has received much attention throughout the history of deep learning (see [21] and references therein). The resurgence of deep learning in the 2000s also began with unsupervised pre-training [22–26]. The success of unsupervised pre-training in NLP [27–32] has revived much interest in unsupervised pre-training in computer vision, especially contrastive training [15, 33–37]. In practice, supervised pre-training is highly successful in computer vision. For example, many studies, e.g., [38–42], have shown that ConvNets pre-trained on ImageNet, Instagram, and JFT can provide strong improvements for many downstream tasks.

Supervised ImageNet pre-training is the most widely-used initialization method for machine vision (e.g., [2–5]). Shen et al [6] and He et al. [1], however, demonstrate that ImageNet pre-training does not work well if we consider a much different task such as COCO object detection. Ghiasi et al. [7] find model trained with random initialization outperforms the ImageNet pre-trained model on COCO object detection when strong regularization is applied. Poudel et al. [43] on the other hand show that ImageNet pre-training is not necessary for semantic segmentation with CityScapes if aggressive data augmentation is applied. Furthermore, Raghu et al. [44] show that ImageNet pre-training does not

improve medical image classification tasks. Compared to these previous works, our work takes a step further and studies the role of pre-training in computer vision in greater detail with stronger data augmentation, different pre-training methods (supervised and self-supervised), and different pre-trained checkpoint qualities.

Our paper does not study targeted pre-training in depth, e.g., using an object detection dataset to improve another object detection dataset, for two reasons. Firstly, targeted pre-training is expensive and not scalable. Secondly, there exists evidence that pre-training on a dataset that is the same as the target task still can fail to yield improvements. For example, Shao et al. [45] found that pre-training on the Open Images object detection dataset actually *hurts* COCO performance. More analysis of targeted pre-training can be found in [46].

Our work argues for the scalability and generality of self-training (e.g., [8–10]). Recently, self-training has shown significant progress in deep learning (e.g., image classification [11, 12], machine translation [13], and speech recognition [14, 47]). Most closely related to our work is Xie et al. [12] who also use strong data augmentation in self-training but for image classification. Closer in applications are semi-supervised learning for detection and segmentation (e.g., [48–52]), who only study self-training in isolation or without a comparison against ImageNet pre-training. They also do not consider the interactions between these training methods and data augmentations.

3 Methodology

3.1 Methods and Control Factors

Data Augmentation: We use four different augmentation policies of increasing strength that work for both detection and segmentation. This allows for varying the strength of data augmentation in our analysis. We design our augmentation policies based on the standard flip and crop augmentation in the literature [16], AutoAugment [53, 54], and RandAugment [55]. The standard flip and crop policy consists of horizontal flips and scale jittering [16]. The random jittering operation resizes an image to (0.8, 1.2) of the target image size and then crops it. AutoAugment and RandAugment are originally designed with the standard scale jittering. We increase scale jittering (0.5, 2.0) in AutoAugment and RandAugment and find the performances are significantly improved. For RandAugment we use a magntiude of 10 for all models [55]. We arrive at our four data augmentation policies which we use for experimentation: FlipCrop, AutoAugment, AutoAugment with higher scale jittering, RandAugment with higher scale jittering. Throughout the text we will refer to them as: **Augment-S1**, **Augment-S2**, **Augment-S3** and **Augment-S4** respectively. The last three augmentation policies are stronger than He et al. [1] who use only a FlipCrop-based strategy.

Pre-training: To evaluate the effectiveness of pre-training, we study ImageNet pre-trained checkpoints of varying quality. To control for model capacity, all checkpoints use the same model architecture but have different accuracies on ImageNet (as they were trained differently). We use the EfficientNet-B7 architecture [56] as a strong baseline for pre-training. For the EfficientNet-B7 architecture, there are two available checkpoints: 1) the EfficientNet-B7 checkpoint trained with AutoAugment that achieves 84.5% top-1 accuracy on ImageNet; 2) the EfficientNet-B7 checkpoint trained with the Noisy Student method [12], which utilizes an additional 300M unlabeled images, that achieves 86.9% top-1 accuracy.[2] We denote these two checkpoints as **ImageNet** and **ImageNet++** , respectively. Training from a random initialization is denoted by **Rand Init**. All of our baselines are therefore stronger than He et al. [1] who only use ResNets for their experimentation (EfficientNet-B7 checkpoint has an approximately 8% higher accuracy than a ResNet-50 checkpoint). Table 1 summarizes our notations for data augmentations and pre-trained checkpoints.

Self-training: We use a simple self-training method inspired by [9, 12, 48, 57] which consists of three steps. First, a teacher model is trained on the labeled data (e.g., COCO dataset). Then the teacher model generates pseudo labels on unlabeled data (e.g., ImageNet dataset). Finally, a student is trained to optimize the loss on human labels and pseudo labels jointly. Our experiments with various hyperparameters and data augmentations indicate that self-training with this standard loss function can be unstable. To address this problem, we implement a loss normalization technique, which is described in Appendix B.

[2] `https://github.com/tensorflow/tpu/tree/master/models/official/efficientnet`

Name	Description
Augment-S1	Weakest augmentation: Flips and Crops
Augment-S2	Third strongest augmentation: AutoAugment, Flips and Crops
Augment-S3	Second strongest augmentation: Large Scale Jittering, AutoAugment, Flips and Crops
Augment-S4	Strongest augmentation: Large Scale Jittering, RandAugment, Flips and Crops
Rand Init	Model initialized w/ random weights
ImageNet Init	Model initialized w/ ImageNet pre-trained checkpoint (84.5% top-1)
ImageNet++ Init	Model initialized w/ higher performing ImageNet pre-trained checkpoint (86.9% top-1)

Table 1: Notations for data augmentations and pre-trained models used throughout this work.

3.2 Additional Experimental Settings

Object Detection: We use COCO dataset [58] (118k images) for supervised learning. In self-training, we experiment with ImageNet [59] (1.2M images) and OpenImages [60] (1.7M images) as unlabeled datasets. We adopt RetinaNet detector [16] with EfficientNet-B7 backbone and feature pyramid networks [61] in the experiments. We use image size 640×640, pyramid levels from P_3 to P_7 and 9 anchors per pixel as done in [16]. The training batch size is 256 with weight decay 1e-4. The model is trained with learning rate 0.32 and a cosine learning rate decay schedule [62]. At the beginning of training the learning rate is linearly increased over the first 1000 steps from 0.0032 to 0.32. All models are trained using synchronous Batch Normalization. For all experiments using different augmentation strengths and datasets sizes, we allow each model to train until it converges (when training longer stops helping or even hurts performance on a held-out validation set). For example, training takes 45k iterations with Augment-S1 and 120k iterations with Augment-S4, when both models are randomly initialized. For results using SpineNet, we use the model architecture and hyper-parameters reported in the paper [17]. When we use SpineNet, due to memory constraints we reduce the batch size from 256 to 128 and scale the learning rate by half. The hyper-parameters, except batch size and learning rate, follow the default implementation in the SpineNet open-source repository.[3] All SpineNet models also use Soft-NMS with a sigma of 0.3 [63]. In self-training, we use a hard score threshold of 0.5 to generate pseudo box labels. We use a total 512 batch size with 256 from COCO and 256 from pseudo dataset. The other training hyper-parameters remain the same as those in supervised training. For all experiments the parameters of the student model are initialized by the teacher model to save training time. Experimental analysis studying the impact of student model initialization during self-training can be found in Appendix C.

Semantic Segmentation: We use the `train` set (1.5k images) of PASCAL VOC 2012 segmentation dataset [64] for supervised learning. In self-training, we experiment with augmented PASCAL dataset [18] (9k images), COCO [58] (240k images, combining labeled and unlabeled datasets), and ImageNet [59] (1.2M images). We adopt a NAS-FPN [19] model architecture with EfficientNet-B7 and EfficientNet-L2 backbone models. Our NAS-FPN model uses 7 repeats with depth-wise separable convolution. We use pyramid levels from P_3 to P_7 and upsample all feature levels to P_2 and then merge them by a sum operation. We apply 3 layers of 3×3 convolutions after the merged features and then attach a 1×1 convolution for 21 class prediction. The learning rate is set to 0.08 for EfficientNet-B7 and 0.2 for EfficientNet-L2 with batch size 256 and weight decay 1e-5. All models are trained with a cosine learning rate decay schedule and use synchronous Batch Normalization. EfficientNet-B7 is trained for 40k iterations and EfficientNet-L2 for 20k iterations. For self-training, we use a batch size of 512 for EfficientNet-B7 and 256 for EfficientNet-L2. Half of the batch consists of supervised data and the other half pseudo data. Other hyper-parameters follow those used in supervised training. Additionally, we use a hard score threshold of 0.5 to generate segmentation masks and pixels with a smaller score are set to the ignore label. Lastly, we apply multi-scale inference augmentation with scales of (0.5, 0.75, 1, 1.25, 1.5, 1.75) to compute the segmentation masks for pseudo labeling.

In Appendix H, we show the optimal training iterations and loss hyperparameters used for all of our experiments.

[3] `https://github.com/tensorflow/tpu/tree/master/models/official/detection`

4 Experiments

4.1 The effects of augmentation and labeled dataset size on pre-training

This section expands on the findings of He et al. [1] who study the weaknesses of pre-training on the COCO dataset as they vary the size of the labeled dataset. Similar to their study, we use ImageNet for supervised pre-training and vary the COCO labeled dataset size. Different from their study, we also change other factors: data augmentation strengths and pre-trained model qualities (see Section 3.1 for more details). As mentioned above, we use RetinaNet object detectors with the EfficientNet-B7 architecture as the backbone. Below are our key findings:

Pre-training hurts performance when stronger data augmentation is used. We analyze the impact of pre-training when we vary the augmentation strength. In Figure 1-Left, when we use the standard data augmentation (Augment-S1), pre-training helps. But as we increase the data augmentation strength, the value of pre-training diminishes.

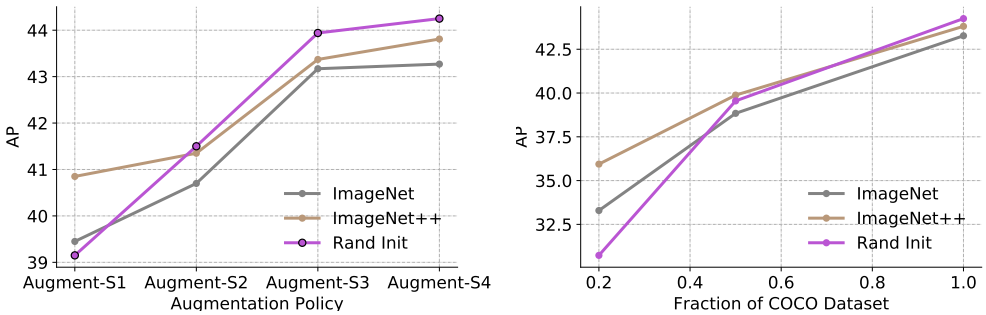

Figure 1: The effects of data augmentation and dataset size on pre-training. **Left**: Supervised object detection performance under various ImageNet pre-trained checkpoint qualities and data augmentation strengths on COCO. **Right**: Supervised object detection performance under various COCO dataset sizes and ImageNet pre-trained checkpoint qualities. All models use Augment-S4 (for similar results with other augmentation methods see Appendix D).

Furthermore, in the stronger augmentation regimes, we observe that pre-training actually *hurts* performance by a large amount (-1.0 AP). This result was not observed by He et al. [1], as pre-training only slightly hurts performance (-0.4AP) or is neutral in their experiments.

More labeled data diminishes the value of pre-training. Next, we analyze the impact of pre-training when varying the labeled dataset size. Figure 1-Right shows that pre-training is helpful in the low-data regimes (20%) and neutral or harmful in the high-data regime. This result is mostly consistent with the observation in He et al. [1]. One new finding here is that the checkpoint quality does correlate with the final performance in the low data regime (ImageNet++ performs best on 20% COCO).

4.2 The effects of augmentation and labeled dataset size on self-training

In this section, we analyze self-training and contrast it with the above results. For consistency, we will continue to use COCO object detection as the task of interest, and ImageNet as the self-training data source. Unlike pre-training, self-training only treats ImageNet as unlabeled data. Again, we use RetinaNet object detectors with the EfficientNet-B7 architecture as the backbone to be compatible with previous experiments. Below are our key findings:

Self-training helps in high data/strong augmentation regimes, even when pre-training hurts. Similar to the previous section, we first analyze the performance of object detectors as we vary the data augmentation strength. Table 2 shows the performance of self-training across the four data augmentation methods, and compares them against supervised learning (Rand Init) and pre-training (ImageNet Init). Here we also show the gain/loss of self-training and pre-training to the baseline. The results confirm that in the scenario where pre-training hurts (strong data augmentations: Augment-S2,

Augment-S3, Augment-S4), self-training helps significantly. It provides a boost of more than +1.3AP on top of the baseline, when pre-training hurts by -1.0AP. Similar results are obtained on ResNet-101 (see Appendix E).

Setup	Augment-S1	Augment-S2	Augment-S3	Augment-S4
Rand Init	39.2	41.5	43.9	44.3
ImageNet Init	(+0.3) 39.5	(-0.7) 40.7	(-0.8) 43.2	(-1.0) 43.3
Rand Init w/ ImageNet Self-training	(+1.7) 40.9	(+1.5) 43.0	(+1.5) 45.4	(+1.3) 45.6

Table 2: In regimes where pre-training hurts, self-training with the same data source helps. All models are trained on the full COCO dataset.

Self-training works across dataset sizes and is additive to pre-training. Next we analyze the performance of self-training as we vary the COCO labeled dataset size. As can be seen from Table 3, self-training benefits object detectors across dataset sizes, from small to large, regardless of pre-training methods. Most importantly, at the high data regime of 100% labeled set size, self-training significantly improves all models while pre-training hurts.

In the low data regime of 20%, self-training enjoys the biggest gain of +3.4AP on top of Rand Init. This gain is bigger than the gain achieved by ImageNet Init (+2.6AP). Although the self-training gain is smaller than the gain by ImageNet++ Init, ImageNet++ Init uses 300M additional unlabeled images.

Self-training is quite additive with pre-training even when using the *same data source*. For example, in the 20% data regime, utilizing an ImageNet pre-trained checkpoint yields a +2.6AP boost. Using both pre-training and self-training with ImageNet yields an additional +2.7AP gain. The additive benefit of combining pre-training and self-training is observed across all of the dataset sizes.

Setup	20% Dataset	50% Dataset	100% Dataset
Rand Init	30.7	39.6	44.3
Rand Init w/ ImageNet Self-training	(+3.4) 34.1	(+1.8) 41.4	(+1.3) 45.6
ImageNet Init	33.3	38.8	43.3
ImageNet Init w/ ImageNet Self-training	(+2.7) 36.0	(+1.7) 40.5	(+1.3) 44.6
ImageNet++ Init	35.9	39.9	43.8
ImageNet++ Init w/ ImageNet Self-training	(+1.3) 37.2	(+1.6) 41.5	(+0.8) 44.6

Table 3: Self-training improves performance for all model initializations across all labeled dataset sizes. All models are trained on COCO using Augment-S4.

4.3 Self-supervised pre-training also hurts when self-training helps in high data/strong augmentation regimes

The previous experiments show that ImageNet pre-training hurts accuracy, especially in the highest data and strongest augmentation regime. Under this regime, we investigate another popular pre-training method: self-supervised learning.

The primary goal of self-supervised learning, pre-training without labels, is to build universal representations that are transferable to a wider variety of tasks and datasets. Since supervised ImageNet pre-training hurts COCO performance, potentially self-supervised learning techniques not using label information could help. In this section, we focus on COCO in the highest data (100% COCO dataset) and strongest augmentation (Augment-S4) setting. Our goal is to compare random initialization against a model pre-trained with a state-of-the-art self-supervised algorithm. For this purpose, we choose a checkpoint that is pre-trained with the SimCLR framework [15] on ImageNet. We use the checkpoint before it is fine-tuned on ImageNet labels. All backbones models use ResNet-50 as SimCLR only uses ResNets in their work.

The results in Table 4 reveal that the self-supervised pre-trained checkpoint hurts performance just as much as supervised pre-training on the COCO dataset. Both pre-trained models *decrease* performance by -0.7AP over using a randomly initialized model. Once again we see self-training improving performance by +0.8AP when both pre-trained models hurt performance. Even though both self-supervised learning and self-training ignore the labels, self-training seems to be more effective at using the unlabeled ImageNet data to help COCO.

Setup	COCO AP
Rand Init	41.1
ImageNet Init (Supervised)	(-0.7) 40.4
ImageNet Init (SimCLR)	(-0.7) 40.4
Rand Init w/ Self-training	(+0.8) 41.9

Table 4: Self-supervised pre-training (SimCLR) hurts performance on COCO just like standard supervised pre-training. Performance of **ResNet-50** backbone model with different model initializations on full COCO. All models use Augment-S4.

4.4 Exploring the limits of self-training and pre-training

In this section we combine our knowledge about the interactions of data augmentation, self-training and pre-training to improve the state-of-the-art. Below are our key results:

COCO Object Detection. In this experiment, we use self-training and Augment-S3 as the augmentation method. The previous experiments on full COCO suggest that ImageNet pre-training hurts performance, so we do not use it. Although the control experiments use EfficientNet and ResNet backbones, we use SpineNet [17] in this experiment as it is closer to the state-of-the-art. For self-training, we use Open Images Dataset (OID) [60] as the self-training unlabeled data, which we found to be better than ImageNet (for more information about the effects of data sources on self-training, see Appendix F). Note that OID is found to not be helpful on COCO by pre-training in [45].

Table 5 shows our results on the two largest SpineNet models, and compares them against previous best single-model, single-crop performance on this dataset. For the largest SpineNet model we improve upon the best 52.8AP SpineNet model by +1.5AP to achieve 54.3AP. Across all model variants, we obtain at least a +1.5AP gain.

Model	# FLOPs	# Params	AP (`val`)	AP (`test-dev`)
AmoebaNet+ NAS-FPN+AA (1536)	3045B	209M	50.7	—
EfficientDet-D7 (1536)	325B	52M	52.1	52.6
SpineNet-143† (1280)	524B	67M	50.9	51.0
SpineNet-143 (1280) w/ Self-training	524B	67M	(+1.5) **52.4**	(+1.6) **52.6**
SpineNet-190† (1280)	1885B	164M	52.6	52.8
SpineNet-190 (1280) w/ Self-training	1885B	164M	(+1.6) **54.2**	(+1.5) **54.3**

Table 5: Comparison with the strong models on COCO object detection. Self-training results use the Open Images Dataset. Parentheses next to the model name denote the training image size. † The SpineNet baselines don't use **Augment-S3** that is used in the Self-training experiments as the models were found to be too unstable and were unable to finish training.

PASCAL VOC Semantic Segmentation. For this experiment, we use NAS-FPN architecture [19] with EfficientNet-B7 [56] and EfficientNet-L2 [12] as the backbone architectures. Due to PASCAL's small dataset size, pre-training still matters much here. Hence, we use a combination of pre-training, self-training and strong data augmentation for this experiment. For pre-training, we use the ImageNet++ initialization for the EfficientNet backbones. For augmentation, we use Augment-S4. We use the `aug` set of PASCAL [18] as the additional data source for self-training, which we found to be more effective than ImageNet.

Table 6 shows that our method improves state-of-the-art by a large margin. We achieve 90.5% mIOU on the PASCAL VOC 2012 test set using single-scale inference, outperforming the old state-of-the-art 89% mIOU which utilizes multi-scale inference. For PASCAL, we find pre-training with a good checkpoint to be crucial, without it we achieve 41.5 % mIOU. Interestingly, our model improves the previous state-of-the-art by 1.5% mIOU even using much less human labels in training. Our method uses labeled data from ImageNet (1.2M images) and PASCAL train segmentation (1.5k images). In contrast, previous state-of-the-art models [65] used 250x additional labeled classification data for pre-training: JFT (300M images), and 86x additional labeled segmentation data: COCO (120k images), and PASCAL `aug` (9k images). For a visualization of pseudo labeled images, see Appendix G.

Model	Pre-trained	# FLOPs	# Params	mIOU (val)	mIOU (test)
ExFuse †	ImageNet, COCO			85.8	87.9 ‡
DeepLabv3+	ImageNet	177B		80.0	—
DeepLabv3+	ImageNet, JFT, COCO	177B		83.4	—
DeepLabv3+ †	ImageNet, JFT, COCO	3055B		84.6	89.0 ‡
Eff-B7	ImageNet++	60B	71M	85.2	—
Eff-B7 w/ Self-training	ImageNet++	60B	71M	(+1.5) **86.7**	—
Eff-L2	ImageNet++	229B	485M	88.7	—
Eff-L2 w/ Self-training	ImageNet++	229B	485M	(+1.3) **90.0**	**90.5**

Table 6: Comparison with state-of-the-art models on PASCAL VOC 2012 val/test set. † indicates multi-scale/flip ensembling inference. ‡ indicates fine tuning the model on the train+val with hard classes being duplicated [20]. EfficientNet models (Eff) are trained on PASCAL train set for validation results and train+val for test results. Self-training uses the aug set of PASCAL.

5 Discussion

Rethinking pre-training and universal feature representations. One of the grandest goals of computer vision is to develop universal feature representations that can solve many tasks. Our experiments show the limitation of learning universal representations from both classification and self-supervised tasks, demonstrated by the performance differences in self-training and pre-training. Our intuition for the weak performance of pre-training is that pre-training is not aware of the task of interest and can fail to adapt. Such adaptation is often needed when switching tasks because, for example, good features for ImageNet may discard positional information which is needed for COCO. We argue that jointly training the self-training objective with supervised learning is more adaptive to the task of interest. We suspect that this leads self-training to be more generally beneficial.

The benefit of joint-training. A strength of the self-training paradigm is that it jointly trains the supervised and self-training objectives, thereby addressing the mismatch between them. But perhaps we can jointly train ImageNet and COCO to address this mismatch too? Table 7 shows results for *joint-training*, where ImageNet classification is trained jointly with COCO object detection (we use the exact setup as self-training in this experiment). Our results indicate that ImageNet pre-training yields a +2.6AP improvement, but using a random initialization and joint-training gives a comparable gain of +2.9AP. This improvement is achieved by training *19 epochs* over the ImageNet dataset. Most ImageNet models that are used for fine-tuning require much longer training. For example, the ImageNet Init (supervised pre-trained model) needed to be trained for 350 epochs on the ImageNet dataset.

Moreover, pre-training, joint-training and self-training are all additive using the same ImageNet data source (last column of the table). ImageNet pre-training gets a +2.6AP improvement, pre-training + joint-training gets +0.7AP improvement and doing pre-training + joint-training + self-training achieves a +3.3AP improvement.

Setup	Sup. Training	w/ Self-training	w/ Joint Training	w/ Self-training w/ Joint Training
Rand Init	30.7	(+3.4) 34.1	(+2.9) 33.6	(+4.4) 35.1
ImageNet Init	33.3	(+2.7) 36.0	(+0.7) 34.0	(+3.3) 36.6

Table 7: Comparison of pre-training, self-training and joint-training on COCO. All three methods use ImageNet as the additional dataset source. All models are trained on 20% COCO with Augment-S4.

The importance of task alignment. One interesting result in our experiments is ImageNet pre-training, even with additional human labels, performs worse than self-training. Similarly, we verify the same phenomenon on PASCAL dataset. On PASCAL dataset, the aug set is often used as an additional dataset, which has much noisier labels than the train set. Our experiment shows that with strong data augmentation (Augment-S4), training with train+aug actually hurts accuracy. Meanwhile, pseudo labels generated by self-training on the same aug dataset significantly improves accuracy. Both results suggest that noisy (PASCAL) or un-targeted (ImageNet) labeling is worse than targeted pseudo labeling.

It is worth mentioning that Shao et al. [45] report pre-training on Open Images hurts performance on COCO, despite both of them being annotated with bounding boxes. This means that not only

Setup	`train`	`train` + `aug`	`train` + `aug` w/ Self-training
ImageNet Init w/ Augment-S1	83.9	(+0.8) 84.7	(+1.7) 85.6
ImageNet Init w/ Augment-S4	85.2	(-0.4) 84.8	(+1.5) 86.7

Table 8: Performance on PASCAL VOC 2012 using `train` or `train` and `aug` for the labeled data. Training on `train` + `aug` hurts performance when strong augmentation (Augment-S4) is used, but training on `train` while utilizing `aug` for self-training improves performance.

we want the task to be the same but also the annotations to be the same for pre-training to be really beneficial. Self-training on the other hand is very general and can use Open Images successfully to improve COCO performance in Appendix F, a result that suggests self-training can align to the task of interest well.

Limitations. There are still limitations to current self-training techniques. In particular, self-training requires more compute than fine-tuning on a pre-trained model. The speedup thanks to pre-training ranges from 1.3x to 8x depending on the pre-trained model quality, strength of data augmentation, and dataset size. Good pre-trained models are also needed for low-data applications like PASCAL segmentation.

The scalability, generality and flexibility of self-training. Our experimental results highlight important advantages of self-training. First, in terms of flexibility, self-training works well in every setup that we tried: low data regime, high data regime, weak data augmentation and strong data augmentation. Self-training also is effective with different architectures (ResNet, EfficientNet, SpineNet, FPN, NAS-FPN), data sources (ImageNet, OID, PASCAL, COCO) and tasks (Object Detection, Segmentation). Secondly, in terms of generality, self-training works well even when pre-training fails but also when pre-training succeeds. In terms of scalability, self-training proves to perform well as we have more labeled data and better models. One bitter lesson in machine learning is that most methods fail when we have more labeled data or more compute or better supervised training recipes, but that does not seem to apply to self-training.

Broader and Social Impact

Our paper studies self-training, a machine learning technique, with applications in object detection and segmentation. As a core machine learning method, self-training can enable machine learning methods to work better and with less data. So it should have broader applications in computer vision, and other fields such as speech recognition, NLP, bioinformatics etc. The datasets in our study are generic and publicly available, which do not tie to any specific application. We foresee positive impacts if the method is applied to datasets in self-driving or healthcare. But the method can also be applied to other datasets and sensitive applications that have ethical implications such as mass surveillance.

Acknowledgements

We thank Anelia Angelova, Aravind Srinivas, and Mingxing Tan for comments and suggestions.

References

[1] Kaiming He, Ross Girshick, and Piotr Dollár. Rethinking imagenet pre-training. In *ICCV*, 2019.

[2] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014.

[3] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *ICML*, 2014.

[4] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015.

[5] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *PAMI*, 2017.

[6] Zhiqiang Shen, Zhuang Liu, Jianguo Li, Yu-Gang Jiang, Yurong Chen, and Xiangyang Xue. Object detection from scratch with deep supervision. *IEEE transactions on pattern analysis and machine intelligence*, 42(2):398–412, 2019.

[7] Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V Le. Dropblock: A regularization method for convolutional networks. In *Advances in Neural Information Processing Systems*, 2018.

[8] H Scudder. Probability of error of some adaptive pattern-recognition machines. *IEEE Transactions on Information Theory*, 1965.

[9] David Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *ACL*, 1995.

[10] Ellen Riloff. Automatically generating extraction patterns from untagged text. In *Proceedings of the national conference on artificial intelligence*, 1996.

[11] I Zeki Yalniz, Hervé Jégou, Kan Chen, Manohar Paluri, and Dhruv Mahajan. Billion-scale semi-supervised learning for image classification. *arXiv preprint arXiv:1905.00546*, 2019.

[12] Qizhe Xie, Eduard Hovy, Minh-Thang Luong, and Quoc V Le. Self-training with noisy student improves imagenet classification. In *CVPR*, 2020.

[13] Junxian He, Jiatao Gu, Jiajun Shen, and Marc'Aurelio Ranzato. Revisiting self-training for neural sequence generation. In *ICLR*, 2020.

[14] Jacob Kahn, Ann Lee, and Awni Hannun. Self-training for end-to-end speech recognition. In *ICASSP*, 2019.

[15] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*, 2020.

[16] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *ICCV*, 2017.

[17] Xianzhi Du, Tsung-Yi Lin, Pengchong Jin, Golnaz Ghiasi, Mingxing Tan, Yin Cui, Quoc V. Le, and Xiaodan Song. Spinenet: Learning scale-permuted backbone for recognition and localization. In *CVPR*, 2020.

[18] Bharath Hariharan, Pablo Arbeláez, Lubomir Bourdev, Subhransu Maji, and Jitendra Malik. Semantic contours from inverse detectors. In *ICCV*, 2011.

[19] Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V Le. NAS-FPN: Learning scalable feature pyramid architecture for object detection. In *CVPR*, 2019.

[20] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017.

[21] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 2015.

[22] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 2006.

[23] Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy layer-wise training of deep networks. In *Advances in Neural Information Processing Systems*, 2007.

[24] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *ICML*, 2008.

[25] Honglak Lee, Alexis Battle, Rajat Raina, and Andrew Y Ng. Efficient sparse coding algorithms. In *Advances in Neural Information Processing Systems*, 2007.

[26] Marc'Aurelio Ranzato, Fu Jie Huang, Y-Lan Boureau, and Yann LeCun. Unsupervised learning of invariant feature hierarchies with applications to object recognition. In *CVPR*, 2007.

[27] Andrew M Dai and Quoc V Le. Semi-supervised sequence learning. In *Advances in Neural Information Processing Systems*, 2015.

[28] Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *NAACL*, 2018.

[29] Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. In *ACL*, 2018.

[30] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *ACL*, 2018.

[31] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems*, 2019.

[32] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[33] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *CVPR*, 2006.

[34] Alexey Dosovitskiy, Jost Tobias Springenberg, Martin Riedmiller, and Thomas Brox. Discriminative unsupervised feature learning with convolutional neural networks. In *Advances in Neural Information Processing Systems*, 2014.

[35] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[36] Philip Bachman, R Devon Hjelm, and William Buchwalter. Learning representations by maximizing mutual information across views. In *Advances in Neural Information Processing Systems*, 2019.

[37] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *CVPR*, 2020.

[38] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. CNN features off-the-shelf: an astounding baseline for recognition. In *CVPR Workshops*, 2014.

[39] Simon Kornblith, Jonathon Shlens, and Quoc V Le. Do better imagenet models transfer better? In *CVPR*, 2019.

[40] Dhruv Mahajan, Ross Girshick, Vignesh Ramanathan, Kaiming He, Manohar Paluri, Yixuan Li, Ashwin Bharambe, and Laurens van der Maaten. Exploring the limits of weakly supervised pretraining. In *ECCV*, 2018.

[41] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *ICCV*, 2017.

[42] Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Joan Puigcerver, Jessica Yung, Sylvain Gelly, and Neil Houlsby. Big transfer (bit): General visual representation learning. *arXiv preprint arXiv:1912.11370*, 2019.

[43] Rudra PK Poudel, Stephan Liwicki, and Roberto Cipolla. Fast-scnn: Fast semantic segmentation network. *arXiv preprint arXiv:1902.04502*, 2019.

[44] Maithra Raghu, Chiyuan Zhang, Jon Kleinberg, and Samy Bengio. Transfusion: Understanding transfer learning for medical imaging. In *Advances in neural information processing systems*, pages 3347–3357, 2019.

[45] Shuai Shao, Zeming Li, Tianyuan Zhang, Chao Peng, Gang Yu, Xiangyu Zhang, Jing Li, and Jian Sun. Objects365: A large-scale, high-quality dataset for object detection. In *ICCV*, 2019.

[46] Hengduo Li, Bharat Singh, Mahyar Najibi, Zuxuan Wu, and Larry S Davis. An analysis of pre-training on object detection. *arXiv preprint arXiv:1904.05871*, 2019.

[47] Sree Hari Krishnan Parthasarathi and Nikko Strom. Lessons from building acoustic models with a million hours of speech. In *ICASSP*, 2019.

[48] Chuck Rosenberg, Martial Hebert, and Henry Schneiderman. Semi-supervised self-training of object detection models. In *WACV*, 2005.

[49] Ilija Radosavovic, Piotr Dollár, Ross Girshick, Georgia Gkioxari, and Kaiming He. Data distillation: Towards omni-supervised learning. In *CVPR*, 2018.

[50] Aayush Bansal. An intriguing influence of visual data in learning a representation. Technical report, CMU, 2018.

[51] Liang-Chieh Chen, Raphael Gontijo Lopes, Bowen Cheng, Maxwell D. Collins, Ekin D. Cubuk, Barret Zoph, Hartwig Adam, and Jonathon Shlens. Semi-supervised learning in video sequences for urban scene segmentation. *arXiv preprint arXiv:2005.10266*, 2020.

[52] Kihyuk Sohn, Zizhao Zhang, Chun-Liang Li, Han Zhang, Chen-Yu Lee, and Tomas Pfister. A simple semi-supervised learning framework for object detection. *arXiv preprint arXiv:2005.04757*, 2020.

[53] Ekin D. Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V. Le. AutoAugment: Learning augmentation policies from data. In *CVPR*, 2019.

[54] Barret Zoph, Ekin D. Cubuk, Golnaz Ghiasi, Tsung-Yi Lin, Jonathon Shlens, and Quoc V. Le. Learning data augmentation strategies for object detection. In *ECCV*, 2020.

[55] Ekin D. Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V. Le. RandAugment: Practical automated data augmentation with a reduced search space. In *Advances in Neural Information Processing Systems*, 2020.

[56] Mingxing Tan and Quoc V Le. EfficientNet: Rethinking model scaling for convolutional neural networks. In *ICML*, 2019.

[57] Dong-Hyun Lee. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *ICML Workshops*, 2013.

[58] 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 *ECCV*, 2014.

[59] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

[60] Alina Kuznetsova, Hassan Rom, Neil Alldrin, Jasper Uijlings, Ivan Krasin, Jordi Pont-Tuset, Shahab Kamali, Stefan Popov, Matteo Malloci, Alexander Kolesnikov, Tom Duerig, and Vittorio Ferrari. The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale. *IJCV*, 2020.

[61] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017.

[62] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. In *ICLR*, 2017.

[63] Navaneeth Bodla, Bharat Singh, Rama Chellappa, and Larry S Davis. Soft-nms–improving object detection with one line of code. In *ICCV*, 2017.

[64] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *IJCV*, 2010.

[65] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *ECCV*, 2018.

[66] Ahmet Iscen, Giorgos Tolias, Yannis Avrithis, and Ondrej Chum. Label propagation for deep semi-supervised learning. In *CVPR*, 2019.

[67] Weiwei Shi, Yihong Gong, Chris Ding, Zhiheng MaXiaoyu Tao, and Nanning Zheng. Transductive semi-supervised deep learning using min-max features. In *ECCV*, 2018.

[68] Eric Arazo, Diego Ortego, Paul Albert, Noel E O'Connor, and Kevin McGuinness. Pseudo-labeling and confirmation bias in deep semi-supervised learning. *arXiv preprint arXiv:1908.02983*, 2019.

[69] Philip Bachman, Ouais Alsharif, and Doina Precup. Learning with pseudo-ensembles. In *Advances in Neural Information Processing Systems*, 2014.

[70] Antti Rasmus, Mathias Berglund, Mikko Honkala, Harri Valpola, and Tapani Raiko. Semi-supervised learning with ladder networks. In *Advances in Neural Information Processing Systems*, 2015.

[71] Samuli Laine and Timo Aila. Temporal ensembling for semi-supervised learning. In *ICLR*, 2017.

[72] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *Advances in Neural Information Processing Systems*, 2017.

[73] Takeru Miyato, Shin-ichi Maeda, Shin Ishii, and Masanori Koyama. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *PAMI*, 2018.

[74] Yucen Luo, Jun Zhu, Mengxi Li, Yong Ren, and Bo Zhang. Smooth neighbors on teacher graphs for semi-supervised learning. In *CVPR*, 2018.

[75] Siyuan Qiao, Wei Shen, Zhishuai Zhang, Bo Wang, and Alan Yuille. Deep co-training for semi-supervised image recognition. In *ECCV*, 2018.

[76] Yanbei Chen, Xiatian Zhu, and Shaogang Gong. Semi-supervised deep learning with memory. In *ECCV*, 2018.

[77] Kevin Clark, Minh-Thang Luong, Christopher D Manning, and Quoc V Le. Semi-supervised sequence modeling with cross-view training. In *EMNLP*, 2018.

[78] Sungrae Park, JunKeon Park, Su-Jin Shin, and Il-Chul Moon. Adversarial dropout for supervised and semi-supervised learning. In *AAAI*, 2018.

[79] Ben Athiwaratkun, Marc Finzi, Pavel Izmailov, and Andrew Gordon Wilson. There are many consistent explanations of unlabeled data: Why you should average. In *ICLR*, 2018.

[80] Yiting Li, Lu Liu, and Robby T Tan. Decoupled certainty-driven consistency loss for semi-supervised learning. *arXiv preprint arXiv:1901.05657*, 2019.

[81] Vikas Verma, Alex Lamb, Juho Kannala, Yoshua Bengio, and David Lopez-Paz. Interpolation consistency training for semi-supervised learning. In *IJCAI*, 2019.

[82] Qizhe Xie, Zihang Dai, Eduard Hovy, Minh-Thang Luong, and Quoc V Le. Unsupervised data augmentation for consistency training. *arXiv preprint arXiv:1904.12848*, 2019.

[83] David Berthelot, Nicholas Carlini, Ian Goodfellow, Nicolas Papernot, Avital Oliver, and Colin Raffel. MixMatch: A holistic approach to semi-supervised learning. In *Advances in Neural Information Processing Systems*, 2019.

[84] Xiaohua Zhai, Avital Oliver, Alexander Kolesnikov, and Lucas Beyer. S^4L: Self-supervised semi-supervised learning. In *ICCV*, 2019.

[85] Guokun Lai, Barlas Oguz, and Veselin Stoyanov. Bridging the domain gap in cross-lingual document classification. *arXiv preprint arXiv:1909.07009*, 2019.

[86] David Berthelot, Nicholas Carlini, Ekin D Cubuk, Alex Kurakin, Kihyuk Sohn, Han Zhang, and Colin Raffel. Remixmatch: Semi-supervised learning with distribution alignment and augmentation anchoring. *arXiv preprint arXiv:1911.09785*, 2019.

[87] Kihyuk Sohn, David Berthelot, Chun-Liang Li, Zizhao Zhang, Nicholas Carlini, Ekin D. Cubuk, Alex Kurakin, Han Zhang, and Colin Raffel. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. *arXiv preprint arXiv:2001.07685*, 2020.

[88] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive multiview coding. *CoRR*, abs/1906.05849, 2019.

[89] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

A Other Related Work

Self-training is related to the method of pseudo labels [57, 66–68] and consistency training [69–88]. There are many differences between these works and ours. First, self-training is different from consistency training in that self-training uses two models (a teacher and a student) whereas consistency training uses only one model. Secondly, previous works focus on image classification, whereas our work studies object detection and segmentation. Finally, previous works do not study the interactions between self-training and pre-training under modern data augmentation.

B Loss Normalization Analysis

In this work we noticed that the standard loss for self-training $\hat{L} = L_h + \alpha L_p$ can be quite unstable. This is caused by the total loss magnitude drastically changing as α is varied. We thus implement a Loss Normalization method, which stabilizes self-training as we vary α: $\hat{L} = \frac{1}{1+\alpha}(L_h + \alpha \frac{\bar{L}_h}{\bar{L}_p} L_p)$, where L_h, L_p, $\bar{L}_h$ and $\bar{L}_p$ are human loss, pseudo loss and their respective moving averages over training. All moving averages are an exponential moving average with a decay rate of 0.9997.

Figure 2 shows the Loss Normalization performance as we vary the data augmentation strength, training iterations, learning rate and α. These experiments are performed on RetinaNet with a ResNet-101 backbone architecture on COCO object detection. ImageNet is used as the dataset for self-training. As can be seen from the figure, Loss Normalization gets better results in almost all settings, and more importantly, helps avoid training instability when α is large. Across all settings of varying augmentations, training iterations and learning rates we find Loss Normalization achieves an average of +0.4 AP performance over the standard loss combination. Importantly, it also helps in our highest performing Augment-S4 setting by +1.3 AP.

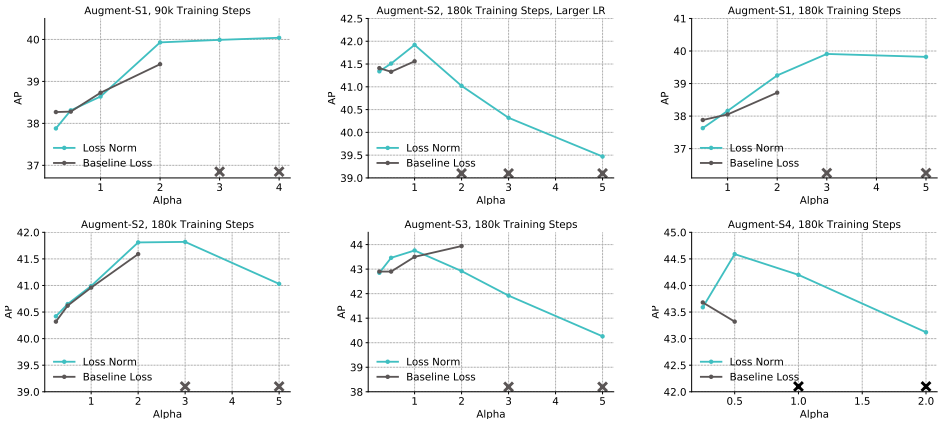

Figure 2: Performance of Loss Normalization across different data augmentation strengths, training iterations and learning rates. All results are on COCO object detection using ImageNet for self-training. The $\times$ symbol represents a run got NaNs and was unable to finish training.

Recent self-training works typically fix the α parameter to be one across all of their experiments [12, 87]. We find in many of our experiments that setting α to one is sub-optimal and that the optimal α changes as the training iterations and augmentation strength varies. Table 9 shows the optimal α's as augmentation and training iterations vary. As the augmentation strength increases the optimal α decreases. Additionally, as the training iterations increases the optimal α increases.

Optimal Alpha	Augment-S1	Augment-S2	Augment-S3
90k Iterations	3.0	2.0	0.5
180k Iterations	4.0	3.0	1.0

Table 9: Optimal α as a function of augmentation strength and training iterations. For each augmentation and training iteration settings the following α were tried: 0.25, 0.5, 1.0, 2.0, 3.0, 4.0.

C Student Model Initialization Study for Self-training

Backbone	Initialization	Augment-S1	Augment-S2	Augment-S3	Augment-S4
Eff-B7	Random	39.2	41.5	43.9	44.3
Eff-B7 w/ Self-training	Teacher	40.9	43.0	45.4	45.6
Eff-B7 w/ Self-training	Random	41.0	42.7	45.0	45.2

Table 10: Study on whether to initialize the student model in self-training with the teacher checkpoint or from random initialization. All models use the training methodology in Section 3.1.

In this section we study how the student model should be initialized in self-training. Table 10 shows the results of initializing the student from the teacher weights and using random initialization. Across all four augmentation regimes we observe similar performance between the two settings, with initializing from the teacher weights doing slightly better (0.3-0.4 AP). One added benefit of initializing the student with the teacher weights is not only due to the increased performance, but the speedup in convergence. Across all augmentation regimes the student model trained from scratch had to train on average 2.25 times as long as the student model initialized with the teacher weights. Therefore for all experiments in the paper we initialize the student with the teacher weights.

D Further Study of Augmentation, Supervised Dataset Size, and Pre-trained Model Quality

We expand upon our previous analysis in Section 4.1 and show how all four augmentation strengths across different COCO sizes interact with pre-trained checkpoint quality on COCO. Figure 3 shows the interaction of all these factors. We again observe the same three points: 1) stronger data augmentation diminishes the value of pre-training, 2) pre-training hurts performance if stronger data augmentation is used, and 3) more supervised data diminishes the value of pre-training. Across all augmentations and data sizes we also observe the better ImageNet pre-trained checkpoint, ImageNet++ , outperforming the standard ImageNet pre-trained model. Interestingly, in the three out of four augmentation regimes where pre-training hurts, the better the pre-trained checkpoint quality, the less it hurts.

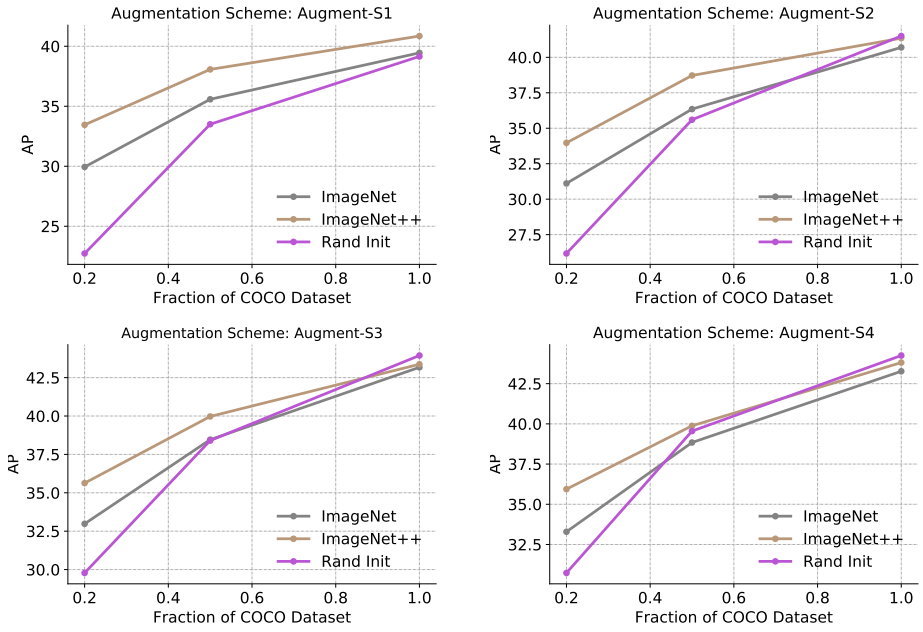

Figure 3: Supervised object detection performance under various COCO dataset sizes, ImageNet pre-trained checkpoint qualities and data augmentation strengths.

As a case study in the low data regime, we study the impact of pre-trained checkpoint quality and augmentation strength on PASCAL VOC 2012. The results in Table 11 indicate that for training on

the PASCAL `train` dataset, which only has 1.5k images, checkpoint quality is very important and improves results significantly. We observe that the gain from checkpoint quality begins to diminish as the augmentation strength increases. Additionally, the performance of the ImageNet checkpoint is again correlated with the performance on PASCAL VOC.

Setup	Augment-S1	Augment-S4
Rand Init	28.4	41.5
ImageNet Init	(+51.8) 80.2	(+39.9) 81.4
ImageNet++ Init	(+55.5) 83.9	(+43.7) 85.2

Table 11: Supervised semantic segmentation performance on PASCAL with different ImageNet pre-trained checkpoint qualities and data augmentation strengths.

E ResNet-101 Self-training Performance on COCO

In the paper we presented our experimental results on COCO with RetinaNet using EfficientNet-B7 and SpineNet backbones. Self-training also works well on other backbones, such as ResNet-101 [89]. Our results are presented in Table 12. Again, self-training achieves strong improvements across all augmentation strengths.

Setup	Augment-S1	Augment-S2	Augment-S3	Augment-S4
Supervised	37.0	39.5	41.9	42.6
Self-training w/ **ImageNet**	(+2.0) 39.0	(+1.8) 41.3	(+0.9) 42.8	(+1.0) 43.6
Self-training w/ **OID**	(+2.5) 39.5	(+2.4) 41.9	(+1.5) 43.4	(+1.3) 43.9

Table 12: Performance of our four different strength augmentation policies. The supervised model is a ResNet-101 with image size 640×640 with RetinaNet using the same training protocol as EfficientNet described in 3.1 with a few minor details. Float32 instead of bfloat16 precision is used and batch norm beta/gamma are included in the weight decay regularization. This helps to improve the training stability. Also the RandAugment magntiude was increased from 10 to 15.

F The Effects of Unlabeled Data Sources on Self-Training

An important question raised from recent experiments is how changing the additional dataset source affects self-training performance. In our analysis we use ImageNet, a dataset designed for image classification that mostly contains *iconic* images. The image contents are known to be quite different from COCO, PASCAL, and Open Images, which contain more *non-iconic* images. Iconic images typically only have one object with its conical view, while non-iconic images capture multiple objects in a scene with their natural views [58]. Table 13 studies how changing the additional data from ImageNet to Open Images Dataset [60] impacts the performance of self-training. Switching the additional dataset source improves performance of self-training up to +0.6 AP over using ImageNet across varying data augmentation strengths on COCO. Interestingly the Open Images Dataset was found to not help COCO by pre-training in [45], but we do see improvements using it over ImageNet for self-training.

Setup	Augment-S1	Augment-S2	Augment-S3	Augment-S4
Supervised	39.2	41.5	43.9	44.3
Self-training w/ **ImageNet**	(+1.7) 40.9	(+1.5) 43.0	(+1.5) 45.4	(+1.3) 45.6
Self-training w/ **OID**	(+2.0) 41.2	(+2.1) 43.6	(+1.6) 45.5	(+1.7) 46.0

Table 13: Performance on different self-training dataset sources with varying augmentation strengths. All models use an EfficientNet-B7 backbone model on COCO object detection starting from a random initialization.

We also study the effects of changing the additional dataset source on PASCAL VOC 2012. In Table 14, we observe that changing the additional data source from ImageNet to COCO improves performance across all of our augmentation strengths. The best self-training dataset is PASCAL `aug`

set, which is in-domain data for the PASCAL task. The PASCAL `aug` set which has only about 9k images improves performance more than COCO with 240k images.

Setup	Augment-S1	Augment-S4
Supervised	83.9	85.2
Self-training w/ **ImageNet**	(+1.1) 85.0	(+0.8) 86.0
Self-training w/ **COCO**	(+1.4) 85.3	(+1.4) 86.6
Self-training w/ **PASCAL**(`aug` set)	(+1.7) 85.6	(+1.5) 86.7

Table 14: Performance on different source datasets for PASCAL Segmentation. All models are initialized using EfficientNet-B7 ImageNet++ checkpoint.

G Visualization of Pseudo Labels in Self-training

PASCAL VOC dataset: The original PASCAL VOC 2012 dataset contains 1464 labeled in `train` set. Extra annotations are provided by [18] resulting in 10582 images in `train+aug`. Most previous works have used the `train+aug` set for training. However, we find that with strong data augmentation training with the `aug` set actually hurts performance (see Table 8). Figure 4 includes selected examples from the `aug` set. We observe the annotations in `aug` set are less accurate compared to the `train` set. For example, some of the images do not include annotation for all the objects in the image and segmentation masks are not precise. The third column of the figure shows pseudo labels generated from our teacher model. From the visualization, we observe that the pseudo labels can have more precise segmentation masks. Empirically, we find that training with this pseudo label set improves performance more than training with the human annotated `aug` set (see Table 8).

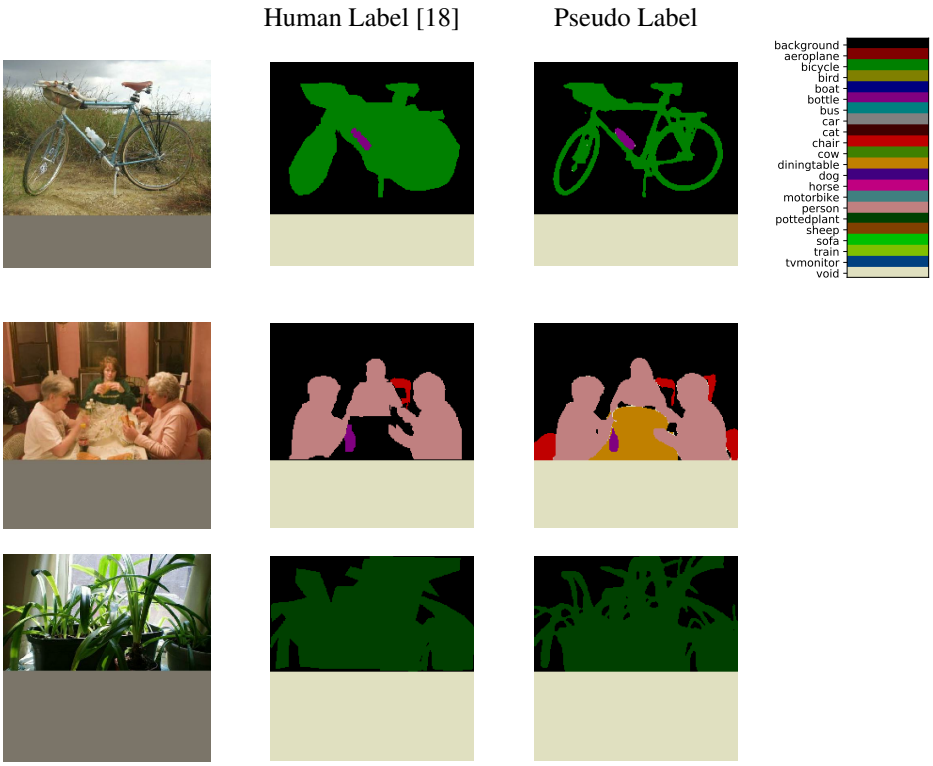

Figure 4: Human labels and pseudo labels on examples selected from PASCAL `aug` dataset. We select the examples where pseudo labels are more accurate than noisy human labels from [18].

ImageNet dataset: Figure 5 shows segmentation pseudo labels generated by the teacher model on 14 randomly-selected images from ImageNet. Interestingly, some of the ImageNet classes that

don't exist in the PASCAL VOC 2012 dataset are predicted as one of its 20 classes. For instance, saw and lizard are predicted as bird. Although pseudo labels are noisy they still improve accuracy of the student model (Table 14).

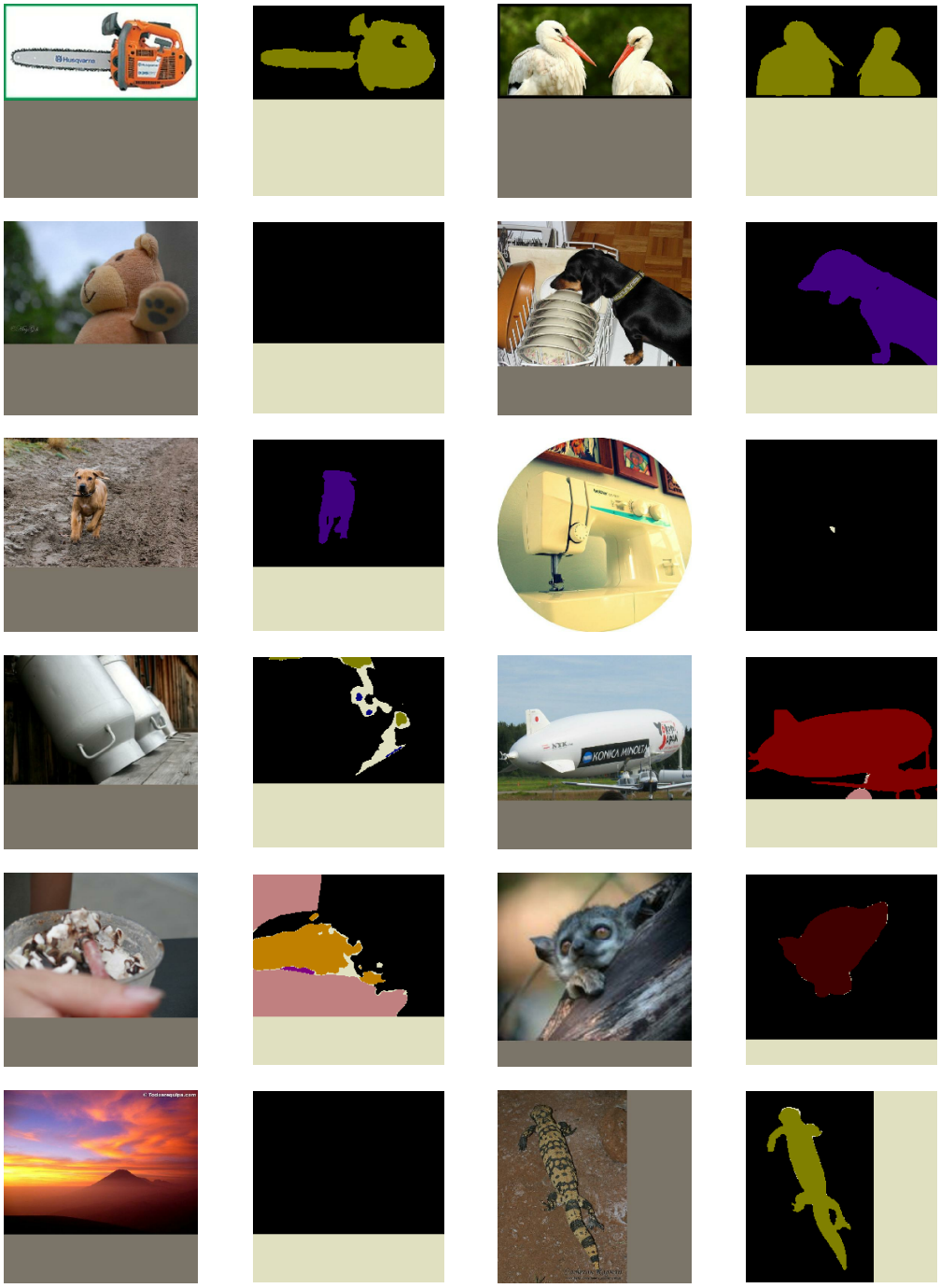

Figure 5: Pseudo segmentation masks on images randomly selected from ImageNet dataset.

H Optimal Model Training Iterations and Alpha Weighting

In all experiments, we allow our models to train until convergence (validation set performance no longer improves). For the self-training experiments we search over a few different alpha values: [0.25, 0.5, 1.0, 2.0, 3.0] (see Appendix B for more details). Below we list all of the optimal training iterations and alphas to promote reproducibility for all of our experiments. For each table the optimal training iteration found is represented by **(45k)**, which means the model was trained for 45000 steps. The optimal alpha is represented as **(1.0)**. An alpha value of **(—)** represents that no alpha is used in the experiment as our supervised learning experiments do not make use of pseudo labeled data. One table (Table 7) jointly trains ImageNet classification and COCO at the same time. For this setup we simply use a scalar to combine the ImageNet classification loss and the COCO loss, which is represented as **(0.2)**. The total training loss is computed by $\text{Loss}_{\text{COCO}} + 0.2 \cdot \text{Loss}_{\text{ImageNet}}$.

Setup	Augment-S1	Augment-S2	Augment-S3	Augment-S4
Rand Init	39.2 **(45k)** **(—)**	41.5 **(90k)** **(—)**	43.9 **(90k)** **(—)**	44.3 **(120k)** **(—)**
ImageNet Init	39.5 **(22.5k)** **(—)**	40.7 **(45k)** **(—)**	43.2 **(90k)** **(—)**	43.3 **(90k)** **(—)**
Rand Init w/ ImageNet Self-training	40.9 **(45k)** **(1.0)**	43.0 **(90k)** **(1.0)**	45.4 **(90k)** **(0.5)**	45.6 **(90k)** **(0.5)**

Optimal training iterations and alpha for **Table 2**.

Setup	20% Dataset	50% Dataset	100% Dataset
Rand Init	30.7 **(45k)** **(—)**	39.6 **(90k)** **(—)**	44.3 **(120k)** **(—)**
Rand Init w/ ImageNet Self-training	34.1 **(90k)** **(3.0)**	41.4 **(90k)** **(1.0)**	45.6 **(90k)** **(0.5)**
ImageNet Init	33.3 **(5.625k)** **(—)**	38.8 **(22.5k)** **(—)**	43.3 **(90k)** **(—)**
ImageNet Init w/ ImageNet Self-training	36.0 **(90k)** **(3.0)**	40.5 **(45k)** **(1.0)**	44.6 **(90k)** **(1.0)**
ImageNet++ Init	35.9 **(5.625k)** **(—)**	39.9 **(11.25k)** **(—)**	43.8 **(45k)** **(—)**
ImageNet++ Init w/ ImageNet Self-training	37.2 **(90k)** **(3.0)**	41.5 **(45k)** **(1.0)**	44.6 **(45k)** **(0.25)**

Optimal training iterations and alpha for **Table 3**.

Setup	COCO AP
Rand Init	41.1 **(200k)** **(—)**
ImageNet Init (Supervised)	40.4 **(160k)** **(—)**
ImageNet Init (SimCLR)	40.4 **(160k)** **(—)**
Rand Init w/ Self-training	41.9 **(120k)** **(0.25)**

Optimal training iterations and alpha for **Table 4**.

Model	# FLOPs	# Params	AP (`val`)	AP (`test-dev`)
SpineNet-143† (1280)	524B	67M	50.9 **(472k)** **(—)**	51.0
SpineNet-143 (1280) w/ Self-training	524B	67M	**52.4** **(472k)** **(0.25)**	**52.6**
SpineNet-190† (1280)	1885B	164M	52.6 **(370k)** **(—)**	52.8
SpineNet-190 (1280) w/ Self-training	1885B	164M	**54.2** **(370k)** **(0.5)**	**54.3**

Optimal training iterations and alpha for **Table 5**. Note due to the high computational demands of this experiment only a subset of alphas and training iterations were tried. For SpineNet-143 alphas of (0.25, 0.5, 1.0) were tried and only a single training iteration. For SpineNet-190 only a single alpha and training iteration were tried.

Model	Pre-trained	# FLOPs	# Params	mIOU (`val`)	mIOU (`test`)
Eff-B7	ImageNet++	60B	71M	85.2 **(40k)** **(1.0)**	—
Eff-B7 w/ Self-training	ImageNet++	60B	71M	**86.7** **(40k)** **(1.0)**	—
Eff-L2	ImageNet++	229B	485M	88.7 **(20k)** **(1.0)**	—
Eff-L2 w/ Self-training	ImageNet++	229B	485M	**90.0** **(20k)** **(1.0)**	**90.5**

Optimal training iterations and alpha for **Table 6**.

Setup	Sup. Training	w/ Self-training	w/ Joint Training	w/ Self-training w/ Joint Training
Rand Init	30.7 **(45k)** **(—)** **(—)**	34.1 **(90k)** **(3.0)** **(—)**	33.6 **(45k)** **(—)** **(0.5)**	35.1 **(90k)** **(2.0)** **(0.2)**
ImageNet Init	33.3 **(5.625k)** **(—)** **(—)**	36.0 **(90k)** **(3.0)** **(—)**	34.0 **(90k)** **(—)** **(0.5)**	36.6 **(90k)** **(2.0)** **(0.2)**

Optimal training iterations, alpha and ImageNet loss weighting for **Table 7**.

Setup	`train`	`train` + `aug`	`train` + `aug` w/ Self-training
ImageNet Init w/ Augment-S1	83.9 (40k) (1.0)	84.7 (40k) (1.0)	85.6 (40k) (1.0)
ImageNet Init w/ Augment-S4	85.2 (40k) (1.0)	84.8 (40k) (1.0)	86.7 (40k) (1.0)

Optimal training iterations and alpha for **Table 8**.

Backbone	Initialization	Augment-S1	Augment-S2	Augment-S3	Augment-S4
EfficientNet-B7	Random	39.2	41.5	43.9	44.3
EfficientNet-B7 w/ Self-training	Teacher	(45k) 40.9	(90k) 43.0	(90k) 45.4	(90k) 45.6
EfficientNet-B7 w/ Self-training	Random	(180k) 41.0	(135k) 42.7	(180k) 45.0	(135k) 45.2

Optimal training iterations and alpha for **Table 10**.

Setup	Augment-S1	Augment-S2	Augment-S3	Augment-S4
Supervised	37.0 (45k) (—)	39.5 (90k) (—)	41.9 (90k) (—)	42.6 (180k) (—)
Self-training w/ **ImageNet**	39.0 (90k) (1.0)	41.3 (90k) (1.0)	42.8 (90k) (0.5)	43.6 (180k) (0.25)
Self-training w/ **OID**	39.5 (90k) (2.0)	41.9 (90k) (2.0)	43.4 (90k) (0.5)	43.9 (180k) (0.5)

Optimal training iterations and alpha for **Table 12**.

Setup	Augment-S1	Augment-S2	Augment-S3	Augment-S4
Supervised	39.2 (45k) (—)	41.5 (90k) (—)	43.9 (90k) (—)	44.3 (120k) (—)
Self-training w/ **ImageNet**	40.9 (45k) (1.0)	43.0 (90k) (1.0)	45.4 (90k) (0.5)	45.6 (90k) (0.5)
Self-training w/ **OID**	41.2 (90k) (3.0)	43.6 (90k) (2.0)	45.5 (90k) (0.5)	46.0 (120k) (0.5)

Optimal training iterations, alpha and ImageNet loss weighting for **Table 13**.

Setup	Augment-S1	Augment-S4
Supervised	83.9	85.2
Self-training w/ **ImageNet**	85.0 (40k) (1.0)	86.0 (40k) (1.0)
Self-training w/ **COCO**	85.3 (40k) (1.0)	86.6 (40k) (1.0)
Self-training w/ **PASCAL**(aug set)	85.6 (40k) (1.0)	86.7 (40k) (1.0)

Optimal training iterations and alpha for **Table 14**.