A. More Technical Details

A.1. Pruning Algorithm

Following the routines in previous LTH [31, 9] works, the algorithm 1 outlines the full iterative magnitude pruning (IMP) procedure.

Algorithm 1 Iterative Magnitude Pruning (IMP)

1: Set the initial mask to $m = 1^{d_1}$, with the pre-training θ_p.
2: **repeat**
3: Train $f(x; m \odot \theta_p, \gamma_p)$ for t epochs with algorithm $\mathcal{A}^{\mathcal{T}}$, i.e., $\mathcal{A}_t^{\mathcal{T}}(f(x; m \odot \theta_p, \gamma_p))$
4: Prune 20% of remaining weights in $\mathcal{A}_t^{\mathcal{T}}(f(x; m \odot \theta_p, \gamma_p))$ and update m accordingly
5: **until** the sparsity of m reaches the desired sparsity level s
6: Return $f(x; m \odot \theta_p)$.

A.2. Top-1 Retrieval Accuracy

Here we presents the detailed calculation of top-1 retrieval accuracy for self-supervised pretraining tasks, including simCLR [10] and MoCo [40]. Given a batch of data with n samples, $\{z_1, \cdots, z_n\}$ and $\{z'_1, \cdots, z'_n\}$ donates the feature representations from the two branches of simCLR or MoCo models. z_i and z'_i are computed from the same input sample with different data augmentations.

For each z_i, we calculate the cosine similarity between z_i and other representations and obtain $\mathcal{D}_i = \{d(z_i, z) | z \in \{z_j, z'_j\}_{j=1}^n / \{z_i\}\}$, where $d(\cdot, \cdot)$ is the cosine similarity measurement. If $\text{argmax}_z \mathcal{D}_i = z'_i$, it suggests the top-1 retrieval is corrected. In the same way, we perform a similar retrieval process for z'_i and $\mathcal{D}'_i$. The concrete calculation formulation of top-1 retrieval accuracy is depicted as follows:

$$\frac{\sum_{i=1}^{n} \left[\mathbb{I}(\text{argmax}_z \mathcal{D}_i = z'_i) + \mathbb{I}(\text{argmax}_z \mathcal{D}'_i = z_i) \right]}{2 \times n} \times 100\%, \quad (2)$$

where $\mathbb{I}(\cdot)$ is the indicator function.

B. More Experimental Results

B.1. YOLOv4 Detection Results with Other Metrics

In this section, we report the other two evaluation metrics, i.e.,AP_{50} and AP_{75}, for YOLOv4 detection experiments. As shown in Figure 10, similar observations can be drawn that there are subnetworks $f(x; m_{\mathcal{P}} \odot \theta_{\text{p}}, \cdot)$ capable of transferring to the detection task successfully (i.e., without performance degradation compared with full unpruned models) at the (83.22%,89.26%,36.00%) and (73.79%,48.80%,79.03%) sparsity levels under the AP_{50} and AP_{75} metrics for supervised ImageNet, self-supervised simCLR and MoCo pre-training tasks, respectively.

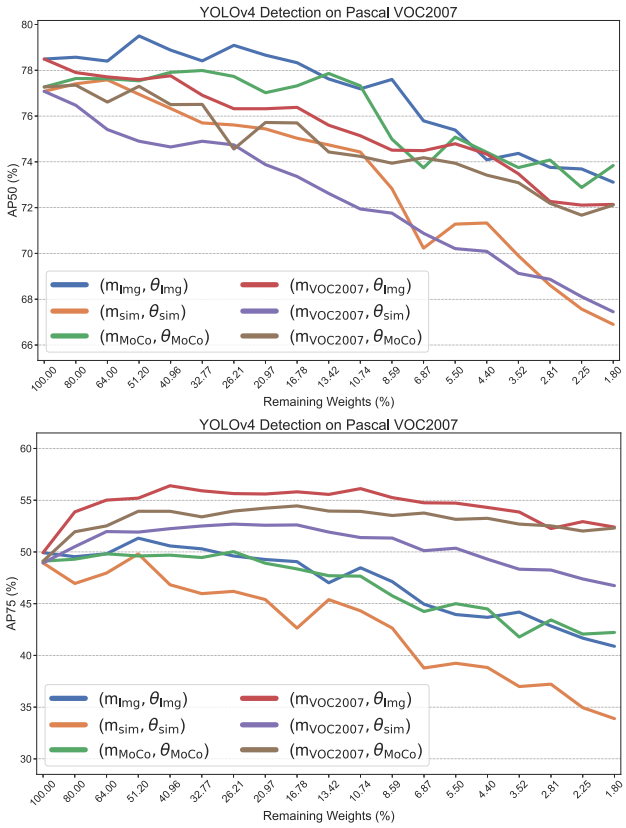

Figure 10. Performance (AP_{50} and AP_{75}) of IMP subnetworks with a range of sparsity from 0.00% to 98.20% on the downstream tasks. Subnetworks $(m_{\text{VOC2007}}, \theta_p)$, $\theta_p \in \{\theta_{\text{Img}}, \theta_{\text{sim}}, \theta_{\text{MoCo}}\}$ are identified on the detection task with pre-trained weights θ_p.

B.2. Faster RCNN and SSD Detection Results

In this section, we conduct extra experiments with Faster RCNN [69] and SSD [53] on Pascal VOC datasets. Specifically, we train detection models for 24K/120K iterations with a batch size 4/32, a polynomial learning rate (LR) decay (with power 0.9 and initial LR 0.005) / a multi-step learning rate decay (with initial LR 0.001 amd $\times 0.1$ at the 80K, 110K iterations), SGD optimizer with 0.9 momentum, and 0.0001/0.0005 weight decay for Faster RCNN/SSD detectors, respectively. As shown in 11, the most different observation is the winning tickets $f(x; m_{\mathcal{P}} \odot \theta_p, \cdot)$ found on the pre-training tasks are almost no longer matching subnetworks on the detection task with both Faster RCNN and SSD, which incurs performance degradation compared to unpruned dense models $f(x; \theta_p, \cdot)$. There is an exception that the subnetworks $f(x; m_{\text{MoCo}} \odot \theta_{\text{MoCo}}, \cdot)$ successfully transfer to the detection task with Faster RCNN at the 59.04% sparsity level only under the AP_{50} metric, and with SSD at the 83.22%, 86.58%, 83.22% sparsity level under AP, AP_{50}, AP_{75} metrics, respectively. Other conclusions are consistent with the ones of YOLOv4 detector in the main text.

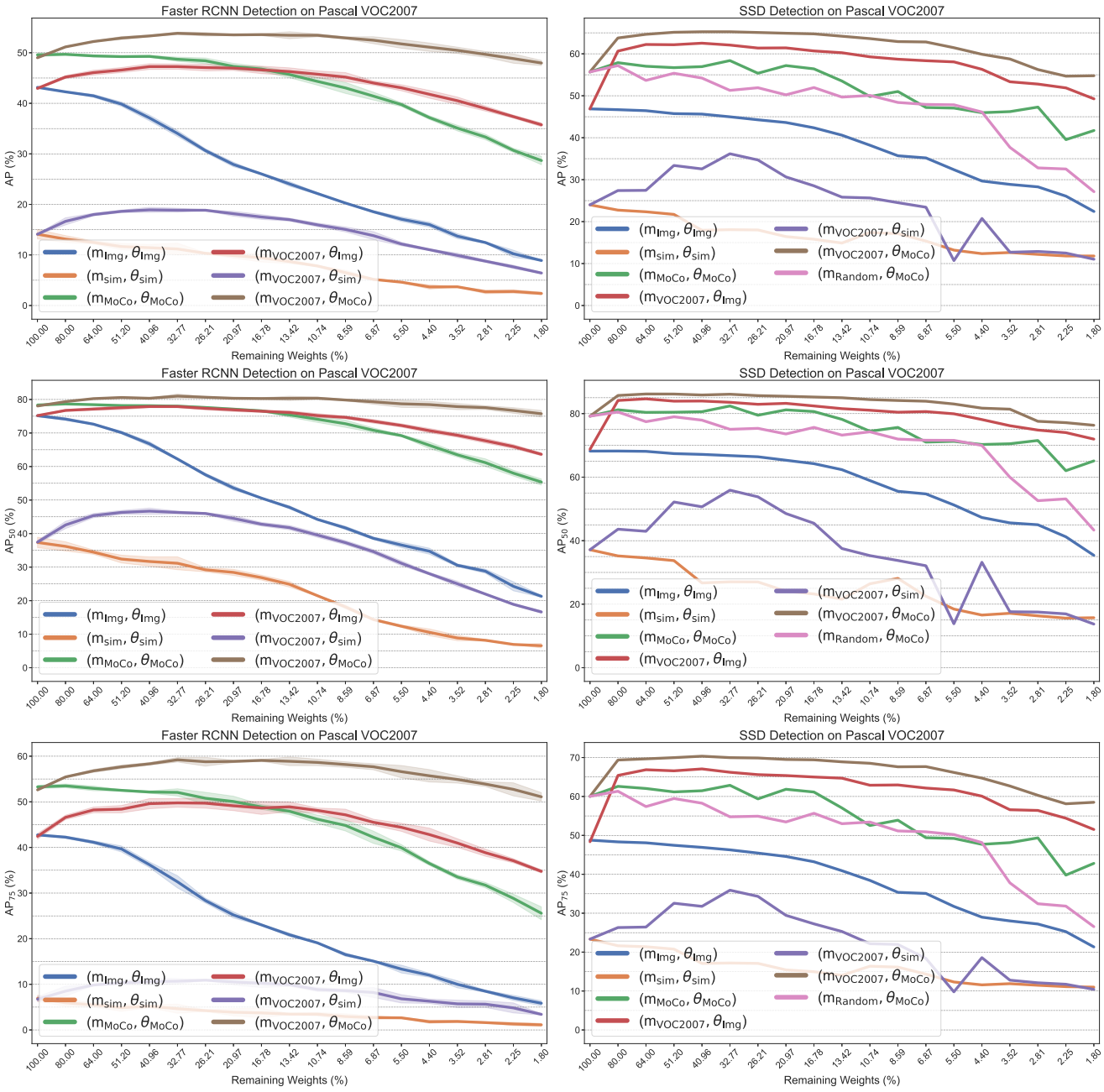

Figure 11. Performance (AP, AP_{50} and AP_{75}) of IMP subnetworks with a range of sparsity from 0.00% to 98.20% on the downstream tasks. Subnetworks $(m_{\text{VOC2007}}, \theta_p)$, $\theta_p \in \{\theta_{\text{Img}}, \theta_{\text{sim}}, \theta_{\text{MoCo}}\}$ are identified on the detection task with pre-trained weights θ_p. Results of Faster RCNN with three independent runs and SSD with one run are presented here.

We notice that detection results of Faster RCNN and SSD with the simCLR pre-training show inferior and unsatisfactory performances, compared with the reported number in BYOL [37]. Although BYOL is implemented with Tensorflow (we use Pytorch) and also has an extra residual block for backbone network, the performance gap is not neglectable. To address this, multiple authors have worked to carefully tune all hyperparameters (learning rate, batch size, training iterations), and thoroughly compared implementation details side-to-side (batch norm, input resolution, etc.). However, we still cannot close the gap. Hence while our results on MoCo and ImageNet are very consistent, we cannot exclude the marginal possibility that simCLR implementation is specifically sensitive to Pytorch versus Tensorflow frameworks (unfortunately, not uncommon) for some reason. Therefore, we put Faster RCNN and SSD detection

results with the simCLR pre-training in the appendix as failure cases, and note that it hardly affects any of our main observations/conclusions.

B.3. Ablation about Larger Pre-training Models

Figure 12 collects the pre-training task performance of subnetworks generated from small- and large-scale pre-trained simCLR models. We observe that heavily compressed, large simCLR models (e.g., ResNet-50) obtain superior performance to lightly compressed, small simCLR models (e.g., ResNet-152), which is consistent with [52]. However, subnetworks found on the small-scale pre-trained simCLR model show a slightly better top-1 retrieval accuracy after the sparsity approaches an extreme level.

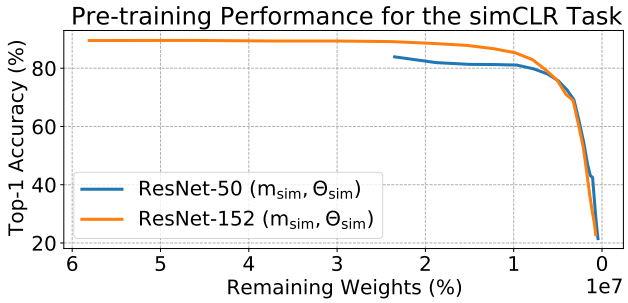

Figure 12. Pre-training performance (top-1 retrieval accuracy as defined in Equation 2) over the number of remaining weights. Subnetworks are found on the simCLR pre-training task with pre-trained ResNet-50 and ResNet-152 weights.