Pruning Convolutional Neural Networks with Self-Supervision

Mathilde Caron, Ari Morcos, Piotr Bojanowski, Julien Mairal, Armand Joulin

Pruning Convolutional Neural Networks with Self-Supervision

Mathilde Caron[1,2], Ari Morcos[1], Piotr Bojanowski[1], Julien Mairal[2], and Armand Joulin[1]

[1]Facebook AI Research
[2]Univ. Grenoble Alpes, Inria, CNRS, Grenoble INP, LJK, 38000 Grenoble, France

Abstract—Convolutional neural networks trained without supervision come close to matching performance with supervised pre-training, but sometimes at the cost of an even higher number of parameters. Extracting subnetworks from these large unsupervised convnets with preserved performance is of particular interest to make them less computationally intensive. Typical pruning methods operate during training on a task while trying to maintain the performance of the pruned network on the same task. However, in self-supervised feature learning, the training objective is agnostic on the representation transferability to downstream tasks. Thus, preserving performance for this objective does not ensure that the pruned subnetwork remains effective for solving downstream tasks. In this work, we investigate the use of standard pruning methods, developed primarily for supervised learning, for networks trained without labels (i.e. on self-supervised tasks). We show that pruned masks obtained with or without labels reach comparable performance when re-trained on labels, suggesting that pruning operates similarly for self-supervised and supervised learning. Interestingly, we also find that pruning preserves the transfer performance of self-supervised subnetwork representations.

Index Terms—Deep Learning, Computer Vision, Unsupervised Feature Learning, Pruning

1 Introduction

Convolutional neural networks (convnets) pre-trained without supervision are emerging from the image recognition community with performance approaching that of supervised ImageNet pre-training [1], [2]. They usually contain a huge number of parameters: some even count hundreds of millions of weights [3], [4] which is an order of magnitude bigger than standard networks used for supervised pre-training [5]. Extracting subnetworks from these large unsupervised convnets with preserved representation power would make both training and use of their visual features less computationally intensive. A well-established solution to reduce parameter-counts of large neural networks is to prune part of their weights [6]. However, pruning has been originally developed in a fully-supervised context: it is usually operated while or after training on a supervised task with the goal of preserving the validation accuracy of a subnetwork on the same task [7], [8]. Yet, in self-supervised learning, the task used to train a network is often merely a proxy [9], [10], thus preserving performance on this surrogate does not guarantee preserved transferability of the resulting subnetwork. To the best of our knowledge, there is no study in the current literature on the impact of pruning on networks pre-trained with self-supervised tasks.

For these reasons, we explore in this work if pruning methods that were developed for supervised learning can be used for networks trained from unlabeled data only. The main questions about pruning networks trained with self-supervision we are trying to answer are the following: How subnetworks obtained from self-supervision compare to subnetworks pruned with labels? Can subnetworks pruned from unlabeled data be used for supervised tasks? Does pruning networks pre-trained with self-supervision deteriorate the quality of their subsequent features when transferred to different downstream tasks?

To investigate these questions, we propose a simple pipeline based on well-established methods from the unstructured pruning and self-supervised learning literatures. In particular, we use the magnitude-based iterative pruning technique of Han *et al*. [8], which compresses networks by alternatively training with labels and pruning the network parameters with the smallest magnitude. Recent works have shown that subnetworks uncovered by Han *et al*. [8] lead to accuracy comparable with that of an unpruned network when re-trained from a selected inherited set of weights termed the "winning tickets" [11], [12] or even from random initialization though for moderate pruning rates [13]. In our work, we build upon these works and simply replace semantic labels by pseudo-labels given by self-supervision pretext tasks. Interestingly, Morcos *et al*. [14] have shown that winning tickets initializations can be re-used across different datasets with a common domain (natural images) trained on the same task (labels classification). In contrast, in this work, we explore among other things if winning tickets initializations from self-supervised tasks can be used as initialization for training label classification on a commun dataset.

Our experiments show that pruning self-supervised networks with a standard pruning method preserves the resulting features quality when evaluated on different downstream tasks. We also observe that transferring an already

pruned pre-trained network gives better transfer performance than pruning a pre-trained network directly on the transfer objective. This is convenient since the subsequent optimal scenario is to provide already pruned pre-trained networks, thus dispensing the need for users to operate any pruning on the target task. Beside, we find that the pruned subnetworks obtained with self-supervision can be re-trained successfully on ImageNet labels classification, and are even on par with supervised pruning when randomly re-initialized. More precisely, the quality of the pruned mask alone is similar between supervised and self-supervised pruning but the winning tickets initializations of self-supervised subnetworks are not as good starting points as the ones inherited from label classification task directly. Overall, we find that pruning networks trained with self-supervision works well, in the sense that the transfer performance of the pruned subnetworks is preserved even for high pruning rates and they can be re-trained from scratch on labels. As a matter of fact, we choose to conduct most of our experiments on ImageNet; we remark indeed that deep networks trained on smaller datasets such as CIFAR-10 are already sparse at convergence, making conclusions drawn about pruning potentially misleading if this effect is not accounted for.

2 Related work

Pruning. Pruning is an approach to model compression [8] and regularization [6] in which weights or nodes/filters are removed, typically by clamping them to zero (see the work of Liu *et al*. [13] for a review of the different pruning methods). It is an active research which has primarily focused on pruning an already trained network [8], [15] or pruning while training [16]. In particular, the iterative pruning during training of Han *et al*. [8] has been extended to continuous pruning [17], layer-wise pruning [18] and with weight sharing [19]. Pruning during training has been considered with ℓ_0 regularization [7], binary convolution [20] or using the hashing trick for weight sharing [21].

The lottery tickets hypothesis. The lottery tickets hypothesis of Frankle and Carbin [11] explores the possibility of pruning early in training by revealing that some sparse subnetworks inside neural networks can reach accuracy matching that of the full network when trained in isolation. Setting the weights of the sparse architecture appropriately is critical to reach good performance. Frankle and Carbin [11] provide a proof of concept of the lottery ticket hypothesis on small vision benchmarks while Frankle *et al*. [12] conduct further experiments with deeper networks, which result in the introduction of rewinding and a consequent revision to the original hypothesis. Rewinding indeed consists in resetting the parameters to their value "early in training" rather than their value from before training. Liu *et al*. [13] question the importance of weights resetting and observe for moderate pruning rates and without rewinding that the pruned architecture alone is responsible for successful training. Zhou *et al*. [22] conduct ablation studies on the lottery tickets hypothesis and show, among others, the importance of the signs of the reset weights. Yu *et al*. [23] investigate the lottery ticket hypothesis in reinforcement learning problems. These works aim at better understanding winning ticket initializations properties; however none of them investigate their relationship with self-supervised tasks. Interestingly, Morcos *et al*. [14] show that winning tickets initializations transfer across different image classification datasets, thus suggesting that winning tickets do not entirely overfit to the particular data distribution on which they are found.

Learning without supervision. In self-supervised learning, a network is trained on a pretext task that does not require any manual annotations. Two main broad types of self-supervised learning approaches appear in the litterature. The first one consists of methods where the pretext task is created by manipulating the input data. This includes predicting relative spatial location, colorizing grayscale images or predicting the rotation applied to an image [4], [9], [10], [24], [25], [26], [27], [28]. The second one is composed of methods [29], [30], [31], [32] where images are treated as different instances that should be discriminated from one another. Representations learnt using self-supervision are most often evaluated via transfer learning to a supervised task. The better the pre-training with self-supervised learning, the better the performance on the transfer task. In this work, we broaden the scope of evaluation of self-supervised methods by evaluating subnetworks pruned with these pretext tasks.

3 Approach

In this work, our goal is to study how a standard pruning method primarily developed for supervised learning applies to networks trained without annotations. We use well-established methods from the unstructured pruning and self-supervised literatures to do so. Specifically, we adopt the magnitude-based unstructured iterative pruning process of Han *et al*. [8] to extract sparse subnetworks from an over-parameterized network. Following recent works, we reset the subsequent subnetworks to a selected set of weights [11], [12] or randomly re-initialize them [13]. The self-supervised tasks we consider are rotation classification of Gidaris *et al*. [10] and the "Exemplar" approach of Dosovitskiy *et al*. [29]. We provide details about our implementation at the end of this section.

Preliminaries. We represent a subnetwork (W, m) by the association of a mask m in $\{0,1\}^d$ and weight W in $\mathbb{R}^d$. The convolutional network, or convnet, function associated with a subnetwork is denoted by $f_{m\odot W}$, where $\odot$ is element-wise product. We refer to the vector obtained at the penultimate layer of a convnet as feature or representation. Such a representation is pruned if m has at least one zero component. The subnetwork weights W may be *pre-trained*, such that the corresponding feature function can be transferred to downstream tasks. On the other hand, the subnetwork weights W may be *initialization weights*, such that the corresponding convnet $f_{m\odot W}$ can be re-trained from scratch.

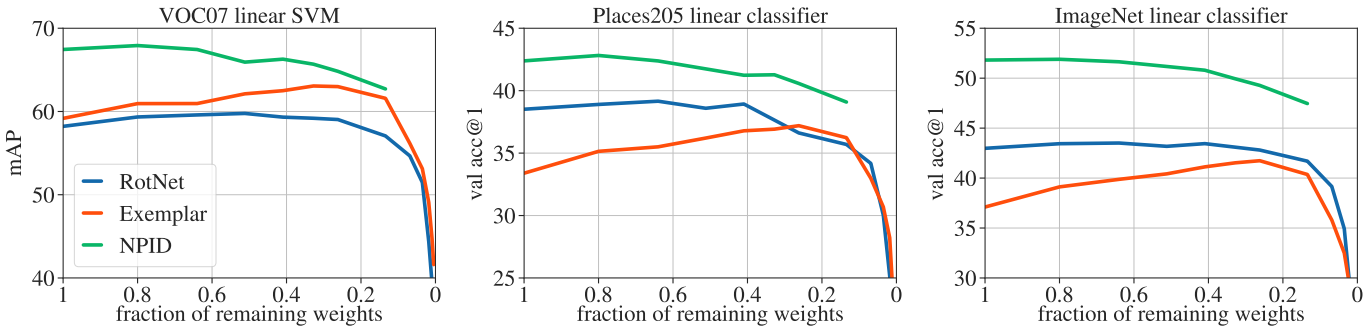


Fig. 1: Transfer learning performance of pruned representations from self-supervised methods as we vary the pruning rate. We show performance when training linear SVMs on VOC2007 and linear classifiers with mini-batch stochastic gradient descent on Places205 and ImageNet. For reference, on VOC07 dataset, the performance for supervised ImageNet features and for random features are respectively 88mAP and 7.7mAP (numbers from Goyal *et al*. [2]). We do not prune NPID networks for extreme rates (i.e. less than 0.1 of remaining weights) because this is too computationally intensive.

3.1 Unstructured magnitude-based pruning

Pruned mask. Han *et al*. [8] propose an algorithm to prune networks by estimating which weights are important. This approach consists of compressing networks by alternatively minimizing a training objective and pruning the network parameters with the smallest magnitude, hence progressively reducing the network size. At each pruning iteration, the network is first trained to convergence, thus arriving at weights W^*. Then, the mask m is updated by setting to zero the elements already masked plus the smallest elements of $\{|W^*[j]| \,|\, m[j] \neq 0\}$.

Weight resetting. Frankle and Carbin [11] refine this approach and propose to also find a good initialization W for each subnetwork such that it may be re-trained from scratch. On small-scale computer vision datasets and with shallow architectures, they indeed show that sub-architectures found with iterative magnitude pruning can be re-trained from the start, as long as their weights are reset to their initial values. Further experiments, however, have shown that this observation does not exactly hold for more challenging benchmarks such as ImageNet [12], [13]. Specifically, Frankle *et al*. [12] found that resetting weights to their value from an early stage in optimization can still lead to good trainable subnetworks. Formally, at each pruning iteration, the subnetwork is reset to weights W_k obtained after k weight updates from the first pruning iteration. Liu *et al*. [13], on the other hand, argue that the mask m only is responsible for the good performance of the subnetwork and thus its weights W may be randomly drawn at initialization. In our work, we consider both weights initialization schemes: winning tickets of Frankle *et al*. [12] or random re-initialization [13].

3.2 Self-supervised learning

We prune networks without supervision by simply setting the training objective in the method of Han *et al*. [8] to a self-supervised pretext task. We consider two prominent self-supervised methods: RotNet [10] and the Exemplar approach of Dosovitskiy *et al*. [29] following the implementation of Doersch *et al*. [33]. RotNet consists in predicting the rotation which was applied to the input image among a set of 4 possible large rotations: $\{0°, 90°, 180°, 270°\}$. Exemplar is a classification problem where each image and its transformations form a class, leading to as many classes as there are training examples. We choose these two self-supervised tasks because they have opposite characteristics: RotNet encourages discriminative features to data transformations and has a small number of classes, while Exemplar encourages invariance to data transformations and its output space dimension is large. We also investigate the non-parametric instance discrimination (NPID) approach of Wu *et al*. [31], which is a variant of the Exemplar method that uses a non-parametric softmax layer and a memory bank of feature vectors.

3.3 Implementation

Pruning. We follow closely the winning tickets setup of Morcos *et al*. [14]. At each pruning iteration, we globally prune 20% of the remaining weights. The last fully-connected and batch-norm layers parameters are left unpruned. We apply up to 30 pruning iterations to reach extreme pruning rates where only 0.1% of the weights remain. Overall we report results for 14 different pruning rates ranging from 20% to 99.9%, thus covering both moderate and extreme sparsity. The weight resetting parameter is set to $3 \times 1,3$M samples, which corresponds to 3 epochs on full ImageNet. More details about this late resetting parameter are in the supplementary material.

Datasets and models. In this work, we choose to mostly work with ImageNet dataset [34], though we also report some results for CIFAR-10 [35]. We use ResNet-50 on ImageNet and ResNet-18 for CIFAR-10 [5]. In supplementary material, we also report results with more architectures: AlexNet [36] and the modified VGG-19 [37] of Morcos *et al*. [14] (multilayer perceptron (MLP) is replaced by a fully connected layer). Our experiments on ImageNet are computationally-demanding since pruning can involve training deep networks from scratch up to 31 times. For this reason, we distribute most of our runs across several GPUs. Models are trained with weight decay and stochastic gradient descent with a momentum of 0.9. We use PyTorch [38]

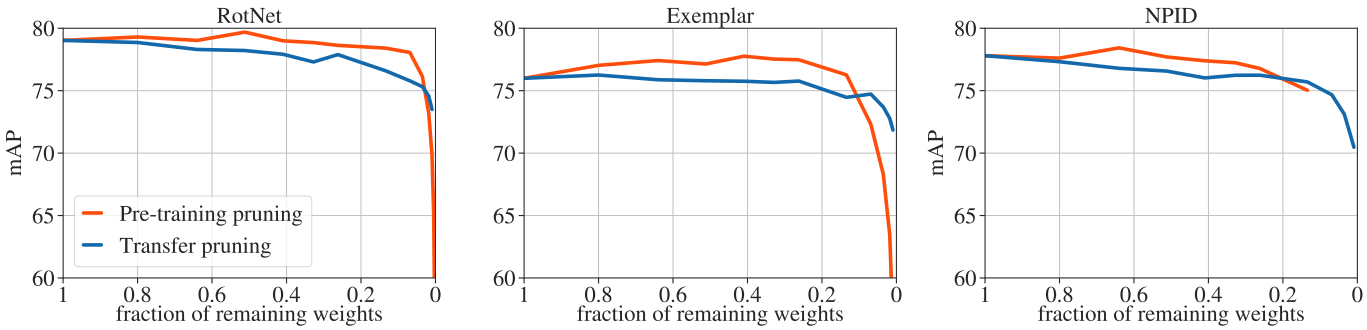


Fig. 2: Transfer learning performance of representations pruned during pre-training or during transfer on VOC07 classification task with full fine-tuning as we vary the pruning rate. For reference, finetuning unpruned supervised ImageNet features gives 90.3 while training from random initialization gives 48.4 (numbers from Goyal *et al*. [2]).

version 1.0 for all our experiments. Full training details for each of our experiments are in the supplementary material. We run each experiment with 6 (CIFAR-10) or 3 (ImageNet) random seeds, and show the mean and standard error of the accuracy.

4 Experimental Study

In our experimental study, we first evaluate the pruned representations: we investigate the effect of pruning on the subsequent self-supervised representations. We also evaluate pruning depending on whether it is performed with or without supervision. Finally, we explore the impact of adding a little supervision during pruning.

4.1 Evaluating Pruned Self-Supervised Features

In this section, we evaluate the quality of pruned self-supervised features by transferring them to different downstream tasks. We show that pruning self-supervised features preserves their transfer performance for a wide range of pruning rates.

Feature transfer performance when pruning. In this experiment, we are interested in the features transferability as we vary the amount of pruned weights in the representation function. We follow part of the benchmark proposed by Goyal *et al*. [2] for evaluating unsupervised representations. In particular, we focus on training linear classifiers for VOC07 [39], Places205 [40] and ImageNet [34] labels classification tasks on top of the final representations given by a pre-trained pruned convnet. In Figure 1, we observe that pruning up to 90% of the networks weights (i.e. there is less than 0.1 remaining weights) does not deteriorate the resulting features quality when evaluated with linear classifier on VOC07, Places205 and ImageNet datasets. Surprisingly, we observe that for RotNet and Exemplar self-supervised approaches, pruning the features even improves slightly their transfer performance. An explanation may be that by pruning, task-specific information is removed from the features which leads to a better transferability. However, this is not the case for the best performing self-supervised method NPID: the quality of the representation remains constant when pruning at moderate rates. In any case, when features are severely pruned (i.e. less than 5% of the weights remain), their quality drops significantly.

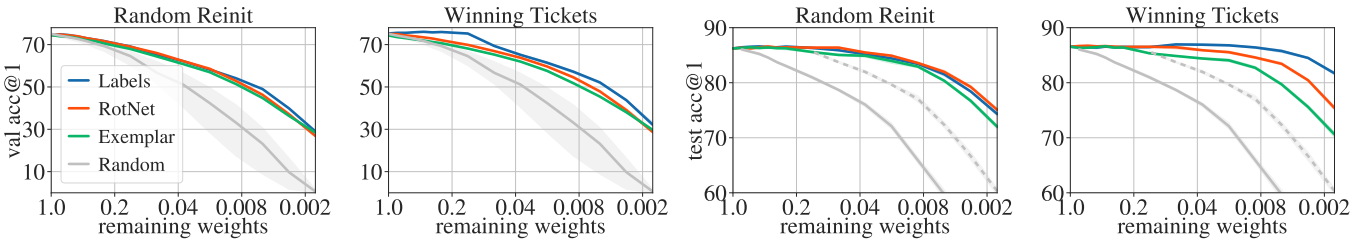
Pruning during pre-training or during transfer? One difficulty of pruning while pre-training is that the training objective used to prune the network (i.e. the pre-training task) is not directly related to the transfer task. Thus, we study in this experiment if pruning directly using the target objective might be a better strategy to obtain good performance on this same target task. In Figure 2, we evaluate pruned unsupervised features by transferring them to Pascal VOC 2007 [39] classification task. The pre-trained features are used as weight initialization, then all the network parameters are trained (i.e. finetuned) on VOC07. The features are either pruned during pre-training (i.e. on RotNet or Exemplar self-supervised tasks) or during transfer (i.e. when finetuning on Pascal VOC07 classification task directly). On Pascal VOC07, we train the models for 90 epochs on combined train and val sets, starting with a learning rate of 0.01 decayed by a factor 10 at epochs 50, 65 and 80; we report test central crop evaluation. We observe in Figure 2 that finetuning a pruned pre-trained network gives better transfer performance than finetuning while pruning. This observation has an interesting real-world impact: pruned pre-trained models can be released for users with limited computational budget without the need of additional pruning on the transfer task from their side.

4.2 Evaluating Self-Supervised Masks

In the previous section we evaluate pruned *representations*. In this section, we evaluate more specifically the *masks* and corresponding *weight initializations* obtained from self-supervised pruning. In particular, we show that these can be re-trained successfully on a supervised task on the same dataset. Finally, we investigate the effect of adding some label supervision during pruning.

4.2.1 *Evaluating Self-Supervised Pruning*

In this set of experiments, we evaluate the quality of the pruned masks and corresponding weight initializations obtained by pruning with a self-supervised objective.



(a) ImageNet (b) CIFAR-10

Fig. 3: We report ImageNet val (a) and CIFAR-10 test (b) top-1 accuracy when re-training subnetworks pruned with self-supervised tasks: RotNet or Exemplar. The x-axis corresponds to different rates of remaining weights. We use two different schemes for weight initialization: "Random Reinit": random re-initialization of the subnetwork [13]; "Winning Tickets": inherited from early phase of training following the lottery tickets hypothesis [12]. We also show performance when re-training subnetworks pruned with labels ("Labels") or randomly pruned subnetworks ("Random") which consist of randomly permuted masks and randomly drawn weights from the initialization distribution. On CIFAR-10, deep models are highly sparse with only approximately $\sim 15\%$ of non-zero weights. Thus, we adjust the random baseline (dashed curve) to start with the correct mask at the natural level of network sparsity (details in Section 4.3).

Experimental setting. We evaluate the pruned subnetworks by training them from their initialization on semantic label classification. The difficulty is the following: unlike the subnetworks pruned with labels, the ones pruned with self-supervised tasks have not seen the labels during pruning and are consequently fully "label-agnostic". We compare them to networks pruned with labels directly and to randomly pruned subnetworks re-trained on label classification. For the remaining of the paper, we use a logarithmic scale for the pruning rates axis in order to focus on situations where very few weights remain. Indeed, for moderate pruning rates, a random pruning baseline is competitive with supervised pruning. Moreover, we observe that pruning deep networks on CIFAR-10 is trivial for moderate rates: we detail this finding in Section 4.3.

Re-training self-supervised masks on label classification. In Figure 3, we show validation accuracy of pruned subnetworks re-trained on label classification, at different pruning ratios. We show results with ResNet-50 for ImageNet and ResNet-18 for CIFAR-10 and investigate more architectures in the supplementary material with similar conclusions. We observe in Figure 3 that subnetworks pruned without any supervision are still capable of reaching good accuracy when re-trained on label classification. When the subnetworks start from random initialization ("Random Reinit"), self-supervised pruning even matches the performance of supervised pruning. However, with winning tickets initializations ("Winning Tickets"), subnetworks pruned without labels perform significantly below supervised pruning, but remain way above the random pruning level. Note that the gap of performance between self-supervised pruning and random pruning increases when the network is severely pruned. Overall, this experiment suggests that self-supervised pruning gives pruned masks and weight initializations which are much better than random pruning when evaluated on label classification, and even on par with labels pruning when randomly re-initialized.

Winning tickets versus random initialization. For pruning with supervision ("Labels" curves in Figure 3), we observe that our results provide further empirical evidence to the lottery tickets hypothesis of Frankle *et al*. [12]. Indeed, we observe in Figure 3 that resetting the weights (winning tickets strategy) of the masks pruned with labels gives significantly better accuracy compared to random re-initialization. Interestingly, this is not the case for subnetworks pruned without labels: winning tickets initialization gives only a very slight boost (or even no boost at all) of performance compared to randomly re-initialization for subnetworks pruned with RotNet or Exemplar self-supervised tasks. Overall, these results suggest that the quality of the pruned mask m itself is similar for supervised or self-supervised pruning but the *weights* of self-supervised subnetworks are not as good starting points as the ones inherited from label classification task directly.

Layerwise Pruning. We further investigate the difference between supervised and self-supervised winning tickets on ImageNet by looking at their performance as we prune up to a given depth. In Figure 4, we verify that when pruning only the first convolutional layers this gap remains narrow, while it becomes much wider when pruning higher level layers. Indeed, it has been observed that most self-supervised approaches produce good shallow and mid-level features but poorer high level features [41]. This means that we should expect the quality of self-supervised pruning to depend on the depth of the pruning. We confirm this intuition and show the difference between self-supervised and supervised pruning increases with pruning depth.

4.2.2 *Semi-Supervised Pruning*

Finally in this experiment, we investigate the impact of using part of the labels for pruning. We find in previous experiments (Section 4.2.1) that pruned masks uncovered by supervision or self-supervision lead to similar performance when re-trained on labels classification, however, winning tickets initializations from labels are

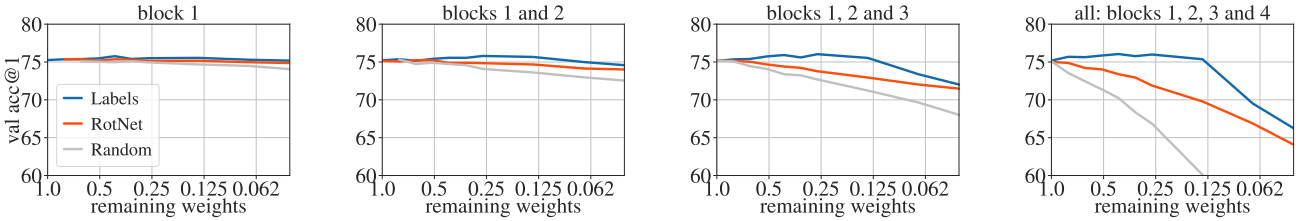
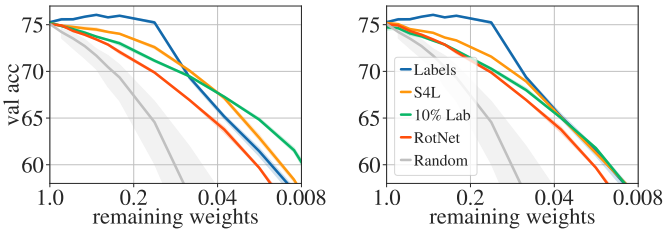


Fig. 4: ImageNet top 1 validation accuracy of winning tickets found by pruning partly or entirely (all) a network with labels classification or RotNet. We show for reference results when the corresponding network layers are randomly pruned.

globally better than ones from pretext self-supervised tasks. We show in this section that adding a little bit of label supervision for pruning improves the resulting winning tickets initializations performance on label classification, suggesting that these initializations are task-dependant to some extent.

Experimental setting. In this experiment, we consider 10% of ImageNet labels. We either sample this labeled subset per class (i.e. we preserve all the classes) or we sample a subset of classes and keep all the images for these classes (i.e. we only have 100 classes). Either way, we get approximately 130k labeled images and use the remaining images without their label. We use S^4L, the semi-supervised setting of Zhai *et al*. [42] since they show better performance on ImageNet classification compared to virtual adversarial training [43] or pseudo-labeling [44]. In particular, we use the RotNet variant of S^4L: the training loss corresponds to the sum of a label classification loss applied to labeled examples only, and a RotNet loss applied to all data samples. We compare this semi-supervised pruning setting to pruning without any labels (masks from Section 4.2.1) and to pruning with few labels only without adding a RotNet loss.We also show for reference the performance of subnetworks obtained with fully-supervised or random pruning. We follow the same methodology from Section 4.2.1 to evaluate pruning masks: we re-train the subsequent pruned masks and corresponding winning tickets weight initializations on ImageNet classification task.

Pruning with limited supervision. In Figure 5, we observe that adding 10% of labels (S^4L) for pruning leads to significantly improved performance for the resulting subnetworks compared to pruning with no labels (RotNet). We previously observed that the pruned masks are of similar quality between supervised and unsupervised pruning (see Figure 3). Hence, the improvement brought by adding labels is likely to be mainly due to better winning tickets initializations. Since both semi-supervised pruning and even supervised pruning with few data perform better than pruning with no labels, we conclude that winning tickets initializations are labels-dependant to some extent. Indeed, we surprisingly observe that simply using 10% labeled images and nothing else for pruning ("10% Lab" in Figure 5) provides better winning tickets initializations than pruning on the entire, unlabeled dataset ("RotNet"). Interestingly, we observe that this gap of performance is much wider when winning tickets are inherited from a preserved set of classes (Fig. 5a) rather than a reduced one (Fig. 5b), which suggests that winning tickets initializations are somewhat "task"-dependant. Finally, subnetworks pruned in the semi-supervised setting ("S^4L") perform better in general than both the unsupervised ("RotNet") and low-data ("10% Lab") ones which leads to think that their respective properties capture different statistics that add up to generate better winning tickets initializations.



(a) Preserved # of classes (b) Preserved # of images per class

Fig. 5: We report ImageNet validation top-1 accuracy when re-training subnetworks pruned with varying level of access to labels supervision. "Labels" corresponds to classical supervised pruning on all the labels, "S^4L" is semi-supervised pruning with S4L RotNet method, "10% Lab" is supervised pruning on a reduced dataset and "RotNet" corresponds to unsupervised pruning. We either sample the 10% labeled subset by selecting 10% images per class (a) or by selecting 10% of classes (b).

4.3 Caveat about pruning deep networks on CIFAR-10

Somewhat surprisingly, we find that prior to any pruning, a large proportion of the weights of a deep architecture trained on CIFAR-10 has converged naturally to zero during training. For example, we observe in Figure 6 that $\sim 85\%$ of the weights of a VGG-19 and $\sim 80\%$ of that of a ResNet-18 at convergence are zeroed (we show results for more architectures in the supplementary material). As a result, it is trivial to prune these networks without any loss in accuracy. Unstructured magnitude-based pruning acts here as *training* since are freezed to zero weights that were going to zero anyway [22]. Overall, while pruning on CIFAR-10 large networks originally tuned for ImageNet at rates above their natural level of sparsity ($\sim 80\%$) is still meaningful, analyzing pruning below this rate may not be conclusive.

In the random global pruning baseline (which can remove non zero weights) of Figure 3, pruning at rates below the natural sparsity of the network degrades accuracy, while pruning of weights that are already zeroed has no effect.

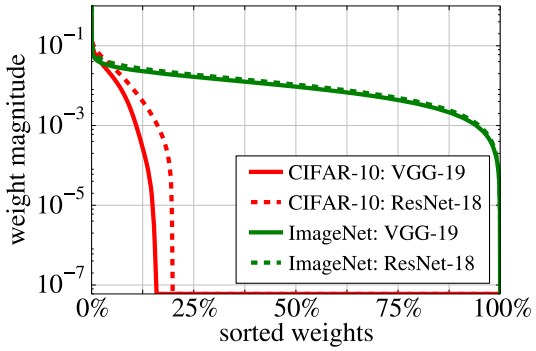


Fig. 6: Magnitude of the weights of a trained network on two different datasets: CIFAR-10 (green) and ImageNet (red). We perform thresholding at machine precision value (bottom of y-axis). On CIFAR-10, a trained VGG-19 is 84.5% sparse while a trained ResNet-18 is 80.3% sparse.

Inconveniently, this performance gap carries over to higher pruning rates (in which we are interested in) and can lead to misleading interpretations. For fair comparison, we adjust the random mask baseline in Figure 3: we remove this effect by first pruning the weights that naturally converge to zero after training. Then, we randomly mask the remaining non-zeroed weights to get different final desired pruning rates. The remaining non-masked weights are randomly initialized. This baseline therefore corrects for the natural sparsity present in CIFAR-10 networks.

5 Conclusion

Our work takes a first step into studying the pruning of networks trained with self-supervised tasks. We believe this is an emergent and important problem due to the recent rise of highly over-parametrized unsupervised pre-trained networks. In our study, we empirically provide different insights about pruning self-supervised networks. Indeed, we show that a well-established pruning method for supervised learning actually works well for self-supervised networks too, in the sense that the quality of the pruned representation is not deteriorated and the pruned masks can be re-trained to good performance on ImageNet labels. This is somewhat surprising given that labels are not seen during pruning and given that the goal of the pruning algorithm we use is to preserve the performance on the training task, which is agnostic to downstream tasks or ground-truth labels.

We also find several limitations to our study. First, we have observed pruning through the scope of unstructured magnitude-based pruning only. Future work might generalize our observations to a wider range of pruning methods, in particular structured pruning. Second, we have observed while conducting our experiments that winning tickets initializations are particularly sensitive to the late resetting parameter (see the supplementary material for a discussion about our choice of rewind parameter). The definition of "early in training" is somehow ill-defined: network weights change much more for the first epochs than for the last ones. Thus, by resetting weights early in their optimization, they contain a vast amount of information. Third, we find that pruning large modern architectures on CIFAR-10 should be done with caution as these networks tend to be sparse at convergence, making unstructured pruning at rates below 80% particularly simple.

Acknowledgment

We thank the members of Thoth and FAIR teams for their help and fruitful discussions. Julien Mairal was funded by the ERC grant number 714381 (SOLARIS project).

References

[1] M. Caron, P. Bojanowski, J. Mairal, and A. Joulin, "Unsupervised pre-training of image features on non-curated data," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2019.
[2] P. Goyal, D. Mahajan, A. Gupta, and I. Misra, "Scaling and benchmarking self-supervised visual representation learning," *Proceedings of the International Conference on Computer Vision (ICCV)*, 2019.
[3] P. Bachman, R. D. Hjelm, and W. Buchwalter, "Learning representations by maximizing mutual information across views," *arXiv preprint arXiv:1906.00910*, 2019.
[4] O. J. Hénaff, A. Razavi, C. Doersch, S. Eslami, and A. v. d. Oord, "Data-efficient image recognition with contrastive predictive coding," *arXiv preprint arXiv:1905.09272*, 2019.
[5] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.
[6] Y. LeCun, J. S. Denker, and S. A. Solla, "Optimal brain damage," in *Advances in Neural Information Processing Systems (NIPS)*, 1990.
[7] C. Louizos, M. Welling, and D. P. Kingma, "Learning sparse neural networks through l_0 regularization," in *International Conference on Learning Representations (ICLR)*, 2018.
[8] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," in *Advances in Neural Information Processing Systems (NIPS)*, 2015.
[9] M. Noroozi and P. Favaro, "Unsupervised learning of visual representations by solving jigsaw puzzles," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016.
[10] S. Gidaris, P. Singh, and N. Komodakis, "Unsupervised representation learning by predicting image rotations," in *International Conference on Learning Representations (ICLR)*, 2018.
[11] J. Frankle and M. Carbin, "The lottery ticket hypothesis: Finding sparse, trainable neural networks," in *International Conference on Learning Representations (ICLR)*, 2019.
[12] J. Frankle, G. K. Dziugaite, D. M. Roy, and M. Carbin, "Linear mode connectivity and the lottery ticket hypothesis," *arXiv preprint arXiv:1912.05671*, 2019.
[13] Z. Liu, M. Sun, T. Zhou, G. Huang, and T. Darrell, "Rethinking the value of network pruning," in *International Conference on Learning Representations (ICLR)*, 2019.
[14] A. S. Morcos, H. Yu, M. Paganini, and Y. Tian, "One ticket to win them all: generalizing lottery ticket initializations across datasets and optimizers," *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.
[15] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient convnets," in *International Conference on Learning Representations (ICLR)*, 2017.
[16] A. Prakash, J. Storer, D. Florencio, and C. Zhang, "Repr: Improved training of convolutional filters," in *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.
[17] Y. Guo, A. Yao, and Y. Chen, "Dynamic network surgery for efficient dnns," in *Advances in Neural Information Processing Systems (NIPS)*, 2016.
[18] X. Dong, S. Chen, and S. Pan, "Learning to prune deep neural networks via layer-wise optimal brain surgeon," in *Advances in Neural Information Processing Systems (NIPS)*, 2017.
[19] K. Ullrich, E. Meeds, and M. Welling, "Soft weight-sharing for neural network compression," in *International Conference on Learning Representations (ICLR)*, 2017.
[20] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016.

[21] W. Chen, J. Wilson, S. Tyree, K. Weinberger, and Y. Chen, "Compressing neural networks with the hashing trick," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2015.

[22] H. Zhou, J. Lan, R. Liu, and J. Yosinski, "Deconstructing lottery tickets: Zeros, signs, and the supermask," in *"Workshop on Identifying and Understanding Deep Learning Phenomena (ICML)"*, 2019.

[23] H. Yu, S. Edunov, Y. Tian, and A. S. Morcos, "Playing the lottery with rewards and multiple languages: lottery tickets in RL and NLP," *International Conference on Learning Representations (ICLR)*, 2020.

[24] C. Doersch, A. Gupta, and A. A. Efros, "Unsupervised visual representation learning by context prediction," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2015.

[25] X. Wang and A. Gupta, "Unsupervised learning of visual representations using videos," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2015.

[26] R. Zhang, P. Isola, and A. A. Efros, "Colorful image colorization," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016.

[27] D. Pathak, R. Girshick, P. Dollár, T. Darrell, and B. Hariharan, "Learning features by watching objects move," in *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[28] Y. Tian, D. Krishnan, and P. Isola, "Contrastive multiview coding," *arXiv preprint arXiv:1906.05849*, 2019.

[29] A. Dosovitskiy, P. Fischer, J. T. Springenberg, M. Riedmiller, and T. Brox, "Discriminative unsupervised feature learning with exemplar convolutional neural networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2016.

[30] P. Bojanowski and A. Joulin, "Unsupervised learning by predicting noise," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2017.

[31] Z. Wu, Y. Xiong, S. X. Yu, and D. Lin, "Unsupervised feature learning via non-parametric instance discrimination," in *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[32] M. Caron, P. Bojanowski, A. Joulin, and M. Douze, "Deep clustering for unsupervised learning of visual features," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

[33] C. Doersch and A. Zisserman, "Multi-task self-supervised visual learning," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2017.

[34] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[35] A. Krizhevsky, "Learning multiple layers of features from tiny images," Tech. Rep., 2009.

[36] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems (NIPS)*, 2012.

[37] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[38] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," 2017.

[39] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International Journal of Computer Vision (IJCV)*, 2010.

[40] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva, "Learning deep features for scene recognition using places database," in *Advances in Neural Information Processing Systems (NIPS)*, 2014.

[41] L. Jing and Y. Tian, "Self-supervised visual feature learning with deep neural networks: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2019.

[42] X. Zhai, A. Oliver, A. Kolesnikov, and L. Beyer, "S4l: Self-supervised semi-supervised learning," *Proceedings of the International Conference on Computer Vision (ICCV)*, 2019.

[43] T. Miyato, S.-i. Maeda, M. Koyama, and S. Ishii, "Virtual adversarial training: a regularization method for supervised and semi-supervised learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2018.

[44] D.-H. Lee, "Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks," in *Workshop on Challenges in Representation Learning, (ICML)*, 2013.

[45] I. Loshchilov and F. Hutter, "Sgdr: Stochastic gradient descent with warm restarts," *arXiv preprint arXiv:1608.03983*, 2016.

Mathilde Caron is currently a second-year PhD student both at Inria in the Thoth team and at Facebook AI Research (FAIR) working on large-scale unsupervised representation learning for vision. Her supervisors are Julien Mairal, Piotr Bojanowski and Armand Joulin. Previously, she did her master thesis on clustering for unsupervised learning of visual representation at FAIR under the supervision of Piotr Bojanowski. Before that, she graduated from both Ecole polytechnique and KTH Royal Institute of Technology where she was mostly interested in applied mathematics and statistical learning.

Ari Morcos is a research scientist at Facebook AI Research (FAIR) in Menlo Park, working on using neuroscience-inspired approaches to understand and build better machine learning systems. Previously, he worked at DeepMind in London. He earned his PhD working with Chris Harvey at Harvard University. For his thesis, he developed methods to understand how neuronal circuits perform the computations necessary for complex behavior. In particular, his research focused on how parietal cortex contributes to evidence accumulation decision-making. For his undergraduate work, he attended UCSD, where he worked with Fred Gage to investigate the role of REST/NRSF in adult neurogenesis.

Piotr Bojanowski is a research scientist at Facebook AI Research, working on machine learning applied to computer vision and natural language processing. His main research interest revolve around large-scale unsupervised learning. Before joining Facebook, in 2016, he got a PhD in Computer Science at the Willow team (INRIA Paris) under the supervision of Jean Ponce, Cordelia Schmid, Ivan Laptev and Josef Sivic. He graduated from Ecole polytechnique in 2013 and received a Masters Degree in Mathematics, Machine Learning and Computer Vision (MVA).

Julien Mairal (SM16) received the Graduate degree from the Ecole Polytechnique, Palaiseau, France, in 2005, and the Ph.D. degree from Ecole Normale Superieure, Cachan, France, in 2010. He was a Postdoctoral Researcher at the Statistics Department, UC Berkeley. In 2012, he joined Inria, Grenoble, France, where he is currently a Research Scientist. His research interests include machine learning, computer vision, mathematical optimization, and statistical image and signal processing. In 2016, he received a Starting Grant from the European Research Council and in 2017, he received the IEEE PAMI young research award. He was awarded the Cor Baayen prize in 2013, the IEEE PAMI young research award in 2017 and the test-of-time award at ICML 2019.

Armand Joulin is a research manager at Facebook AI Research. Prior to this position, he was a postdoctoral fellow at Stanford University working with Fei-Fei Li and Daphne Koller. He did his PhD in Inria and Ecole Normale Superieure, under the supervision of Francis Bach and Jean Ponce. He did his undergrad at Ecole Polytechnique. His subjects of interest are machine learning, computer vision and natural language processing.

6 Supplementary Material

6.1 Late resetting parameter

We follow [12] and use late resetting (or *rewind*) for the winning tickets generation process. Indeed, before re-training a winning ticket, we reset its weights to their value "early in training" of the full over-parameterized network. In our work, we set the late resetting parameter to 1 epoch on CIFAR-10. However, when dataset size, total number of epochs, mini-batch sizes or learning rate vary, it becomes more complicated to choose a rewind criterion that guarantees a fair comparison between all settings. A choice can be to rewind at a point where "the same amount of information" has been processed. Thus, in our work, we choose to set the rewind parameter to $3 \times 1,280,000$ samples for all our experiments on ImageNet, which corresponds to 3 epochs on full ImageNet. We describe in Table 1 to what this rewind parameter corresponds to in terms of number of epoch, number of data samples seen, number of gradient update and percentage of total training for our different experiments. Moreover, we show in Figure 7 the performance of winning tickets generated using 10% of ImageNet with different values of rewind. Each of the considered value corresponds to keeping one of the criteria (number of epoch, number of data samples seen, number of gradient update or percentage of total training) fixed compared to the rewind parameter on full ImageNet (first row of Table 1).

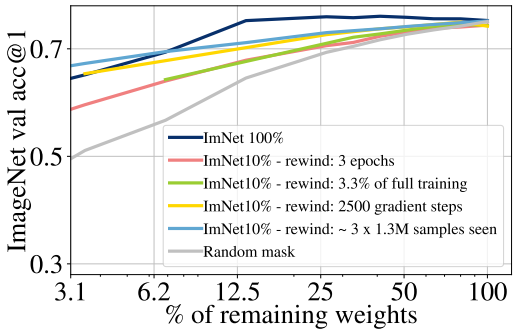


Fig. 7: ImageNet top 1 validation accuracy of winning tickets initialization found with a subset of 10% of ImageNet dataset. We show the influence of different values for the late resetting parameter.

	# epochs	# samples seen	# grad updates	% training
ImNet 100%	3	$\sim 3 \times 1,280,000$	2500	3.3%
ImNet 10%	30	$\sim 3 \times 1,280,000$	5000	15%

TABLE 1

6.2 More architectures for evaluating self-supervised masks

Re-training self-supervised masks on label classification. More results for this experiment can be found in Figure 8.

Layerwise winning tickets. We show more results about the layerwise winning tickets experiment. We generate winning tickets by pruning only the n first convolutional layers of a network, and leaving the remaining of the network unpruned. On AlexNet, we consider 4 situations. From left to right in Figure 9), we prune the first convolutiona layer; up to the third convolutiona layer; up to the fifth convolutional layer; or the whole network. In Figure 9, we verify that for the convolutional layers the gap of performance between labels winning tickets initializations and self-supervised ones remains narrow, while it becomes much wider when pruning the MLP. Note that, even if this effect is particularly visible with an AlexNet, it is happening with most self-supervisedly trained network as can be seen in the main paper.

6.3 Sparse trained networks on CIFAR-10

In this appendix, in Figure 10, we provide results on more architectures about the proportion of weights zeroed during training on CIFAR-10 compared to ImageNet. Note that we do not consider the batch-norm layers parameters, nor the parameters of the last fully-connected layer (since we do not prune it in our setting). For all the considered VGGs we use the modified version of [14], replacing the final MLP by a fully connected layer.

6.4 Hyperparameters and model details

We detail in this appendix the different hyperparameters used in our experiments. We use Adam optimizer on CIFAR-10. On ImageNet, unless specified otherwise, we perform standard data augmentation consisting in croppings of random sizes and aspect ratios and horizontal flips [36]). On CIFAR-10, we use horizontal flips and croppings of fixed size on a 2-padded input image.

- **ImageNet Labels - full dataset - AlexNet**: we train for 90 epochs with a total batch-size of 4096 distributed over 8 GPUs (512 samples per GPU), learning rate of 0.4, weight decay of 0.0001. We decay the learning rate by a factor 10 at epochs 30 and 60.
- **ImageNet Labels - full dataset - ResNet-50**: we train for 90 epochs with a total batch-size of 1536 distributed over 16 GPUs (96 samples per GPU), learning rate of 0.1, weight decay of 0.0001. We decay the learning rate by a factor 10 at epochs 50, 65 and 80.
- **ImageNet Labels - 10% dataset**: we train for 200 epochs with a total batch-size of 768 distributed over 8 GPUs (96 samples per GPU), learning rate of 0.1 warmed up during the first 5 epochs, weight decay of 0.001. We decay the learning rate by a factor 10 at epochs 140, 160 and 180.
- **ImageNet Labels - 10% of classes**: we train for 200 epochs with a total batch-size of 768 distributed over 8 GPUs (96 samples per GPU), learning rate of 0.3 warmed up during the first 5 epochs, weight decay of 0.0003. We decay the learning rate by a factor 10 at epochs 140, 160 and 180.
- **ImageNet Semi-supervised RotNet**: We reproduce the semi-supervised method of [42] and follow precisely their hyperparameter. We train for 200 epochs with a total batch-size of 2048 distributed over 32

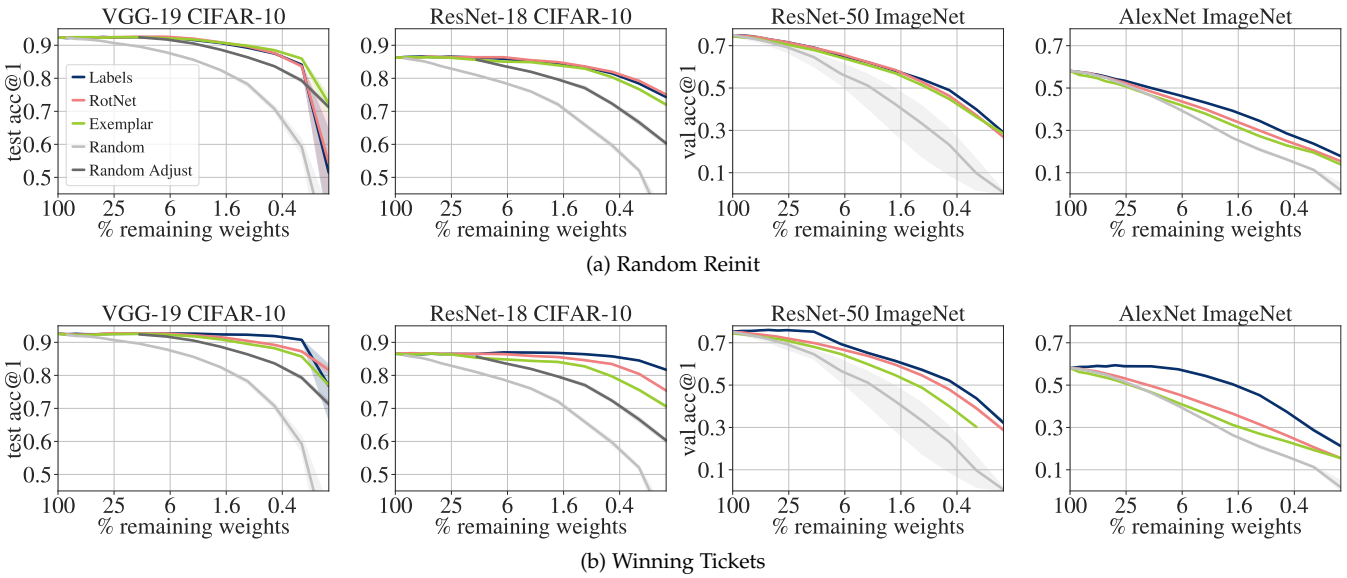


Fig. 8: We report CIFAR-10 test (left columns) and ImageNet val (right columns) top-1 accuracy for subnetworks pruned without labels, i.e. with self-supervised tasks: RotNet or Exemplar. The x-axis corresponds to different pruning ratios. We use two different schemes for weight initialization: Random Reinit (a): random re-initialization of the subnetwork [13]; Winning Tickets (b): inherited from early phase of training like the lottery tickets hypothesis [12]. Architectures are VGG-19 and ResNet-18 for CIFAR-10 and ResNet-50 and AlexNet for ImageNet. We compare the performance with standard supervised pruning (dark blue) and random (grey) subnetworks which consist of randomly permuted masks and randomly drawn weights from the initialization distribution. On CIFAR-10, deep models are highly sparse with only $\sim 15\%$ of non-zero weights. Thus, we adjust the random baseline to start with the correct mask at the natural level of network sparsity.

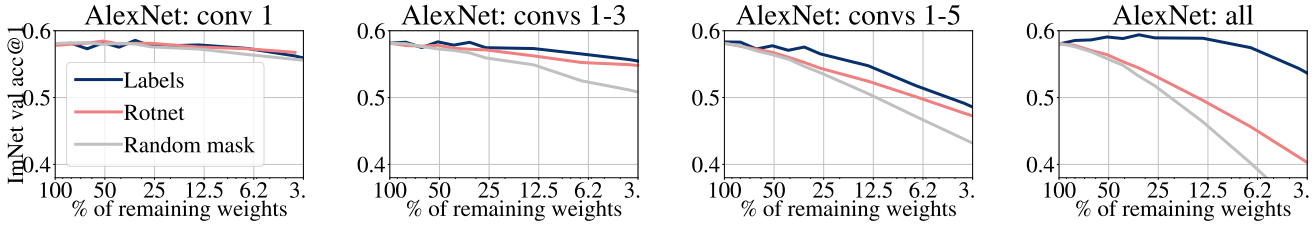


Fig. 9: ImageNet top 1 validation accuracy of winning tickets generated by pruning partly or entirely (all) a network with 2 generation tasks: labels classification or RotNet for AlexNet. We also show for reference results when the network layers are randomly pruned.

GPUs (64 samples per GPU), learning rate of 0.1 warmed up during the first 5 epochs, weight decay of 0.0003. We decay the learning rate by a factor 10 at epochs 140, 160 and 180.

- **ImageNet RotNet - ResNet-50**: we train for 90 epochs with a total batch-size of 1536 distributed over 16 GPUs (96 samples per GPU), learning rate of 1 warmed up during the first 5 epochs, weight decay of 0.00001. We decay the learning rate by a factor 10 at epochs 50, 65 and 80
- **ImageNet Exemplar - ResNet-50**: we train for 40 epochs with a total batch-size of 1536 distributed over 32 GPUs (48 samples per GPU), learning rate of 0.3, weight decay of 0.00001. We decay the learning rate by a factor 10 at epochs 20 and 30. We follow the Exemplar implementation based on triplet margin loss from [33]. As the goal of the task is to learn invariance to data transformation, we use data color augmentation and random small rotations on top of standard data augmentation scheme.
- **ImageNet NPID - ResNet-50**: we train for 150 epochs with a total batch-size of 1024 distributed over 8 GPUs (128 samples per GPU), learning rate of 0.03, weight decay of 0.0001. We use 16384 negatives and a cosine annealing learning rate schedule [45].
- **ImageNet RotNet - AlexNet**: we train for 90 epochs with a total batch-size of 8192 distributed over 16 GPUs (512 samples per GPU), learning rate of 0.5 warmed up during the first 2 epochs, weight decay of 0.00001. We decay the learning rate by a factor 10 at epochs 30 and 60.
- **ImageNet Exemplar - AlexNet**: we train for 30 epochs with a total batch-size of 4096 distributed over 16 GPUs (256 samples per GPU), learning rate of 0.1, weight decay of 0.0001. We decay the learning rate by a factor 10 at epoch 20. We follow the Exemplar implementation based on triplet margin loss from [33]. We use data color augmentation and

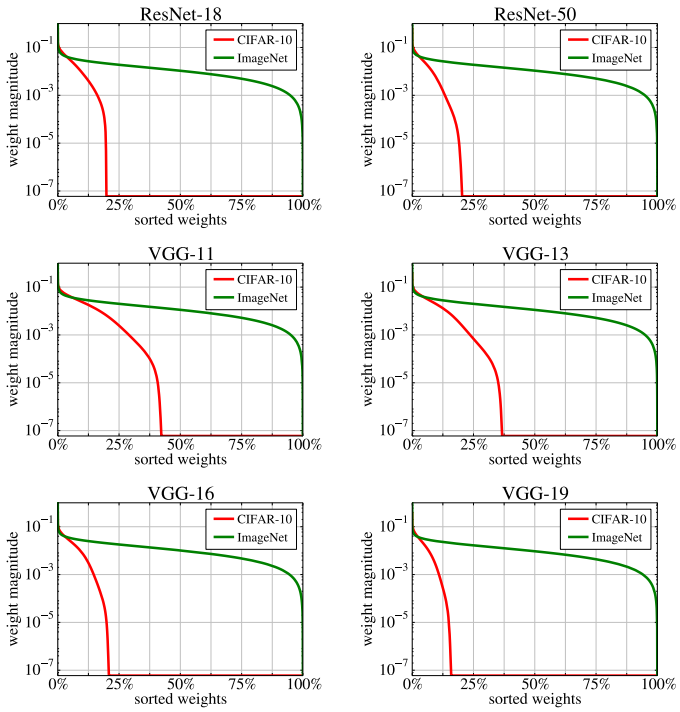


Fig. 10: Magnitude of the weights of a trained network on two different datasets: CIFAR-10 (green) and ImageNet (red) with different datasets. We perform thresholding at machine precision value (bottom of y-axis).

random small rotations on top of standard data augmentation scheme.

- **CIFAR-10 Labels & RotNet - VGG-19 & ResNet-18**: we train for 160 epochs with a total batch-size of 512 on 1 GPU, learning rate of 0.001, weight decay of 0.0001. We decay the learning rate by a factor 10 at epochs 80 and 120.
- **CIFAR-10 Exemplar - VGG-19 & ResNet-18**: we train for 180 epochs with a total batch-size of 512 on 1 GPU, learning rate of 0.0003, weight decay of 0.0001. We decay the learning rate by a factor 10 at epochs 120. As the goal of the task is to learn invariance to data transformation, we use data color augmentation and random small rotations on top of standard data augmentation scheme.