NETWORK PRUNING THAT MATTERS: A CASE STUDY ON RETRAINING VARIANTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Network pruning and retraining is an effective method to reduce the computational expense of over-parameterized neural networks for deployment on low-resource systems. Recent state-of-the-art techniques for retraining pruned networks such as weight rewinding and learning rate rewinding have been shown to outperform the traditional fine-tuning technique in recovering the lost accuracy (Renda et al., 2020), but so far it is unclear what accounts for such performance. In this work, we conduct extensive experiments to verify and analyze the uncanny effectiveness of learning rate rewinding. We find that the reason behind the success of learning rate rewinding is the usage of a *large* learning rate. Similar phenomenon can be observed in other learning rate schedules that involve large learning rates, e.g., the 1-cycle learning rate schedule (Smith & Topin, 2019). By leveraging the *right* learning rate schedule in retraining, we demonstrate a counter-intuitive phenomenon in that randomly pruned networks could even achieve better performance than methodically pruned networks (fine-tuned with the conventional approach). Our results emphasize the cruciality of the learning rate schedule in pruned network retraining – a detail often overlooked by practioners during the implementation of network pruning.

1 INTRODUCTION

Training neural networks is an everyday task in the era of deep learning and artificial intelligence. Generally speaking, given data availability, large and cumbersome networks are often preferred as they have more capacity to exhibit good data generalization. In the literature, large networks are considered easier to train than small ones (Neyshabur et al., 2018; Arora et al., 2018; Novak et al., 2018; Brutzkus & Globerson, 2019). Thus, many breakthroughs in deep learning are strongly correlated to increasingly complex and over-parameterized networks.

However, the use of large networks exacerbate the gap between research and practice since real-world applications usually require running neural networks in low-resource environments for numerous purposes: reducing memory, latency, energy consumption, etc. To adopt those networks to resource-constrained devices, network pruning (LeCun et al., 1990; Han et al., 2015; Li et al., 2016) is often exploited to remove dispensable weights, filters and other structures from neural networks. The goal of pruning is to reduce overall computational cost and memory footprint without inducing significant drop in performance of the network.

A common approach to mitigating performance drop after pruning is retraining: we continue to train the pruned models for some more epochs. In this paper, we are interested in approaches based on learning rate schedules to control the retraining. A well-known practice is *fine-tuning*, which aims to train the pruned model with a small fixed learning rate. More advanced learning rate schedules exist, which we generally refer to as *retraining*. The retraining step is a critical part in implementing network pruning, but it has been largely overlooked and tend to vary in each implementation including differences in learning rate schedules, retraining budget, hyperparameter choices, etc.

Recently, Renda et al. (2020) proposed a state-of-the-art technique for retraining pruned networks namely *learning rate rewinding* (LRRW). Specifically, instead of fine-tuning the pruned networks with a fixed learning rate, usually the last learning rate from the original training schedule (Han et al., 2015; Liu et al., 2019), the authors suggested using the learning rate schedule from the previous t epochs (i.e. rewinding). This seemingly subtle change in learning rate schedule led to an important result: LRRW was shown to achieve comparable performance to more complex and computationally expensive

pruning algorithms while only utilizing simple norm-based pruning. Unfortunately, the authors did not provide the analysis to justify the improvement. In general, it is intriguing to understand the importance of a learning rate schedule and how it affects the final performance of a pruned model.

In this work, we study the behavior of pruned networks under different retraining settings. We found that the efficacy from retraining with learning rate rewinding is rooted in the use of a *large* learning rate, which helps pruned networks to escape from the suboptimality due to pruning. We demonstrate that the success of learning rate rewinding over fine-tuning is *not* exclusive to the learning rate schedule coupling with the original training process. Retraining with a large learning rate could outperform fine-tuning even with some modest retraining, e.g., for a few epochs, and regardless of network compression ratio.

We argue that retraining is of paramount importance to regain the performance in network pruning and should not be overlooked when comparing two pruning algorithms. This is evidenced by our extensive experiments: (1) *randomly* pruned network can outperform methodically pruned network with only (hyper-parameters free) modifications of the learning rate schedule in retraining, and (2) a simple baseline such as norm-based pruning can perform as well as as other complex pruning methods by using a large learning rate restarting retraining schedule.

The contributions of our work are as follows.

- We document a thorough experiment on learning rate schedule for the retraining step in network pruning with different pruning configurations;
- We show that learning rate matters: pruned models retrained with a *large* learning rate consistently outperform those trained by conventional fine-tuning regardless of specific learning rate schedules;
- We present a novel and counter-intuitive result achieved by solely applying large learning rate retraining: a randomly pruned network and a simple norm-based pruned network can perform as well as networks obtained from more sophisticated pruning algorithms.

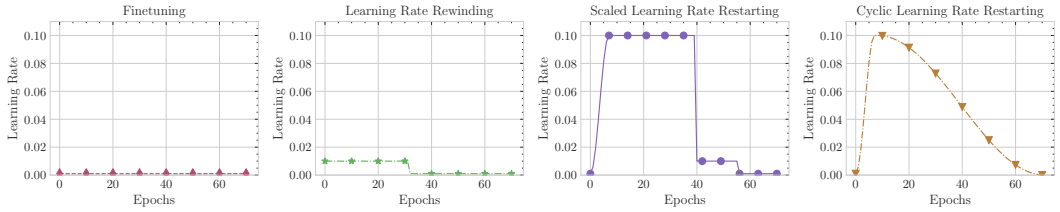
Given the significant impact of learning rate schedule in network pruning, we advocate the following practices: learning rate schedule should be considered as a critical part of retraining when designing pruning algorithms. Rigorous ablation studies with different retraining settings should be made for a fair comparison of pruning algorithms. To facilitate reproducibility, we would release our implementation upon publication.

2 Preliminary and Methodology

Pruning is a common method to produce compact and high performance neural networks from their original large and cumbersome counterparts. In most cases, pruning consists of three steps: training the original network to convergence, prune redundant weights based on some criteria, and retrain the pruned model to regain the performance loss due to pruning. Traditionally, the last step is referred to as *fine-tuning*, i.e., continue to train the pruned model with a small learning rate obtained from the last epoch of the original model. This seemly subtle step is often overlooked when designing pruning algorithms.

Particularly, we found that the implementation of previous pruning algorithms have many notable differences in their retraining step: some employed a small value of learning rate (e.g. 0.001 on ImageNet) to fine-tune the network (Molchanov et al., 2016; Li et al., 2016; Han et al., 2015) for a small number of epochs, e.g., 20 epochs in the work by Li et al. (2016); some used a larger value of learning rate (0.01) with much longer retraining budgets, e.g., 60, 100 and 120 epochs respectively on ImageNet (Zhuang et al., 2018; Gao et al., 2020; Li et al., 2020); You et al. (2019) utilized 1-cycle learning rate schedule (Smith & Topin, 2019) instead of conventional step-wise schedule. Despite such difference, the success of each pruning algorithm is only attributed to the pruning itself. This motivates us to ask the question: do details like learning rate schedule used for retraining matter?

In this section, we strive to understand the behavior of pruned models under different retraining configurations and how they impact the final performance. Specifically, we conduct experiments with different retraining schedules on simple baselines such as ℓ_1-norm filters pruning (Li et al., 2016) and magnitude-based weights pruning (Han et al., 2015). We show that the efficacy of several pruning algorithms can be boosted by simply modifying the learning rate schedule. More importantly, the



(a) FT (b) LRW (c) SLR (d) CLR

Figure 1: Learning rate with different schedules on CIFAR when retraining for 72 epochs. In (a), the learning rate is fixed to the last learning rate of original training (i.e. 0.001). In (b), the learning rate is "rewound" to previous 72 epochs (which is 0.01), and is dropped to 0.001 after 32 epochs. In (c), after warming up the learning rate, we drop its value by the factor of $10\times$ at 50% and 75% of remaining epochs. In (d), we warm up the learning rate from the lowest to the highest value (of standard training) for the first few epochs, then decay the learning rate according to cosine function.

performance gain by retraining can be remarkable: the accuracy loss can drop to zero and in some cases better accuracy than baseline models can be achieved.

To analyze the effect of retraining a pruned network, we based on learning rate rewinding (Renda et al., 2020) and experiment with different retraining settings. Although in the previous work, Renda et al. (2020) demonstrated the efficacy of learning rate rewinding across datasets and pruning criteria, there is a lack of understanding of the actual reason behind the success of this technique. Here we hypothesize that the initial pruned network is a suboptimal solution, staying in a local minima. Learning rate rewinding succeeds because it uses a large learning rate to encourage the pruned networks to converge to another, supposingly better, local minima. Our experiment setups are as follows.

Retraining techniques. To verify this conjecture empirically, we conduct experiments with different learning rate schedules including learning rate rewinding (Renda et al., 2020) while varying pruning algorithms, network architectures and datasets. In this work, we consider the following retraining techniques:

1. FINE-TUNING (FT) Fine-tuning is the most common retraining techniques (Han et al., 2015; Li et al., 2016; Liu et al., 2019). In this approach, we continue train the pruned networks for t epochs with the last (smallest) learning rate of original training.
2. LEARNING RATE REWINDING (LRW) Renda et al. (2020) propose to reuse the learning rate schedule of the original training when retraining pruned networks. Specifically, when retraining for t epochs, we reuse the learning rate schedule from the previous t epochs, i.e., rewinding.
3. SCALED LEARNING RATE RESTARTING (SLR): In this approach, we employ the learning rate schedule that is proportionally identical to the standard training. For example, the learning rate is dropped by a factor of $10\times$ at 50% and 75% of retraining epochs on CIFAR, which is akin to original training learning rate adjustment. The original learning rate schedule can be found in Appendix A.
4. CYCLIC LEARNING RATE RESTARTING (CLR): Instead of using stepwise learning rate schedule as *scaled learning rate restarting*, we leverage the 1-cycle (Smith & Topin, 2019), which is shown to give faster convergence speed than conventional approaches.

Note that for the last two strategies (SLR and CLR), we *warmup* the learning rate for 10% of total retraining budget. For simplicity, we always use the *largest* learning rate of the original training for *learning rate restarting*. For CLR and SLR, the learning rate is increased from the smallest learning rate of original training to the largest one according to *cosine* function. The detailed learning rate schedule of each techniques is depicted in Figure 1.

Pruning algorithms. We consider the following dimensions in our experiments. For pruning methods, we use ℓ_1-norm filters pruning (PFEC) (Li et al., 2016) and (global) magnitude-based weights pruning (MWP) (Han et al., 2015) and evaluate them on the CIFAR-10, CIFAR-100 and ImageNet dataset. We examine both variations of pruning

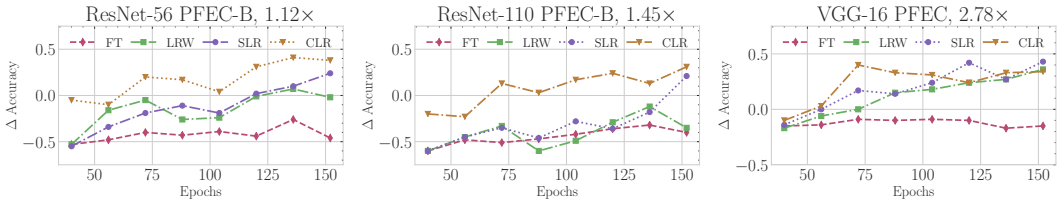
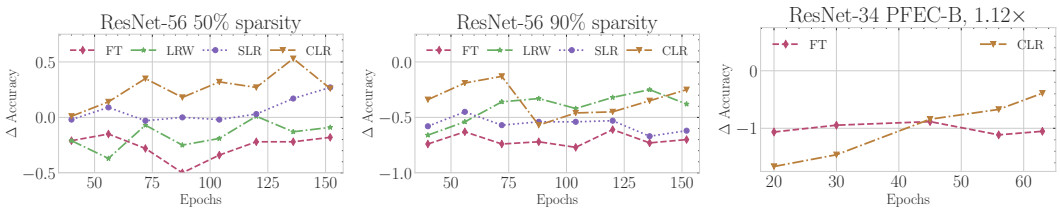


Figure 2: *One-shot structured* pruning on CIFAR-10 dataset using ℓ_1-norm pruning (Li et al., 2016) while varying retraining budgets. As can be seen, learning rate schedule matters. Schedules that employ large learning rates (LLRW, SLR, CLR) are significantly better than fine-tuning. Among them, CLR performs best in most cases.



(a) (b) (c)

Figure 3: *One-shot unstructured* pruning on CIFAR-10 dataset ((a) and (b)) and *structured* pruning on ImageNet (c).

namely one-shot pruning and iterative pruning when comparing the proposed retraining techniques. Furthermore, we also experiment the CLR schedule on HRank (Lin et al., 2020), Taylor Pruning (TP) (Molchanov et al., 2019) and Soft Filter Pruning (SFP) (He et al., 2018).

Our implementation and hyperparameters of ℓ_1-norm pruning and magnitude weight pruning are based on the public implementation of Liu et al. (2019), which is shown to obtain comparable results with the original works. For remaining algorithms, we use official implementations with hyperparameters specified according to their papers. The detailed configurations of training and fine-tuning is provided in the Appendix A.2 for interested readers.

Evaluation. For CIFAR-10 and CIFAR-100, we run each experiment three times and report "$\text{mean} \pm \text{std}$". For ImageNet, we run each experiment once. These settings are kept consistently across architectures, pruning algorithms, retraining techniques, and ablation studies unless otherwise stated.

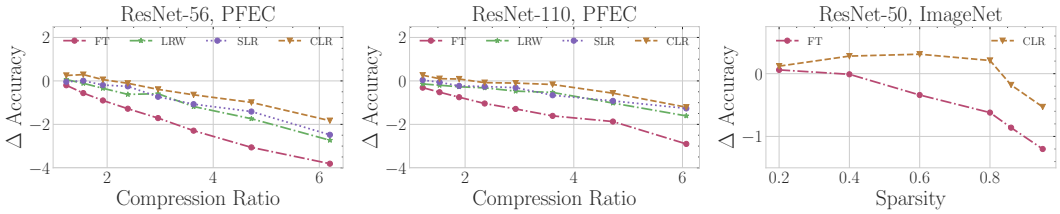
3 A CASE STUDY ON RETRAINING IN NETWORK PRUNING

3.1 RETRAINING COST AND PERFORMANCE TRADE-OFF

We first investigate the performance of retraining techniques while varying the retraining budget. Figure 2 illustrates the results of ℓ_1-norm filters pruning (PFEC) (Li et al., 2016) with different retraining techniques on CIFAR-10. We can observe that, larger values of learning rate always attain higher performance than *fine-tuning* regardless of number of retraining epochs. Furthermore, with some additional retraining budgets, e.g., 80 to 120 epochs compare to 40 epochs in original works, we can attain much higher compression ratio with almost no accuracy drop (see Figure 6 in Appendix).

Figure 3 reports the accuracy of ResNet-56 pruned with magnitude-based weights pruning (Han et al., 2015) in a one-shot manner. We study the impact of different retraining schedules on low and high sparsity i.e. 50% and 90% respectively. It can be observed that under both setting learning rate restarting approaches consistently outperform fine-tuning across number of retraining epochs.

To verify the effectiveness of CLR on large-scale datasets, we conduct experiment using ℓ_1-norm filters pruning with ResNet-34 on ImageNet in Figure 3(c). We find that CLR achieve lower performance than fine-tuning for low epochs. This is expected phenomenon since (Huang et al., 2017)



(a) (b) (c)

Figure 4: *Iterative* pruning on CIFAR-10 dataset using ℓ_1-norm pruning Li et al. (2016) ((a) and (b)) and on ImageNet using magnitude-based weights pruning (Han et al., 2015) ((c)).

also found that learning rate restarting on ImageNet usually requires training for 45 epochs while it only needs 40 epochs on CIFAR-10. When retraining with number of epochs higher than 45, CLR can reach higher accuracy than fine-tuning.

Thus, the value of LRW can be a good heuristic to choose the learning rate for retraining after pruning. A concrete example is that we achieve significant gains with Taylor pruning when retraining for only 25 epochs as shown in Table 2.

3.2 Model Size and Performance Trade-off

In this section, we investigate the trade-off of between model size and performance of compact models under a fixed retraining budgets. Renda et al. (2020) found that learning rate rewinding usually saturate at half of original training, thus, we perform on retraining for 80 epochs on CIFAR-10 and 45 epochs on ImageNet.

We first experiment with iterative ℓ_1-norm pruning on CIFAR-10 and report the results in Figure 4(a,b), we can observe that SLR and CLR also perform comparable or better than LRW in this setting. We then conduct experiments with iterative unstructured pruning on ImageNet with ResNet-50 to verify the superior of CLR compared to fine-tuning in Figure 4(c), we find that MWP with CLR significantly outperform fine-tuning and even increase the accuracy of the network until 80% sparsity (of convolutional layers).

In our experiments, CLR usually reachs slightly higher accuracy than LRW and SLR. A possible explanation is that 1-cycle schedule gives better convergence speed than conventional schedules (Smith & Topin, 2019), thus, bringing much better results when employing "low" retraining budgets.

Discussion. Our empirical results suggest that practitioners should employ retraining with large learning rate schedules as an alternative technique for fine-tuning to obtain compact models with better performance after network pruning. In our results, cyclic learning rate restarting (CLR) is slightly more efficient than scaled learning rate restarting (SLR) and learning rate rewinding (LRW).

3.3 Evaluation with other pruning algorithms

We investigate whether these results hold with other pruning algorithms than simple norm-based pruning. As the performance of larger learning rate schedules such as LRW, SLR, and CLR are rather similar, we select CLR for use in this experiment. Particularly, we examine *CLR* with HRank (Lin et al., 2020), Taylor pruning (Molchanov et al., 2019) and SFP (He et al., 2018).

Soft Filters Pruning (He et al., 2018). Previous sections demonstrate that the success of large learning rate hold for a wide range of pruning algorithms, network architectures and datasets. However, we note that all examined pruning algorithms followed the *pruning-retraining* paradigm, i.e., it is necessary to retrain the pruned model. In this part, we investigate *regularizing* where we gradually, softly prune the network during the course of training. In contrast to *pruning-retraining*, the latter approach does not require further training after final pruning. Being that said, we still make a case study to verify if retraining does improve the performance of models pruned with this type of algorithms.

Table 1: Results (accuracy) for soft filters pruning (He et al., 2018). The "w/o FT" column demonstrate the performance of networks follow original work (do not retrain after final prune). "FT" column indicates results of networks when fine-tuning for 200 more epochs. "CLR-x" columns show accuracy of networks after retraining with CLR for x more epochs."%PF" indicates number of pruned filters.

Model	%PF	w/o FT	FT	CLR-50	CLR-100	CLR-200
CIFAR-10 ResNet-56	30	92.74±0.28	92.85±0.21	92.57±0.19	93.00±0.02	$\mathbf{93.18 \pm 0.25}$
	40	91.95±0.29	91.94±0.26	92.19±0.16	92.23±0.12	$\mathbf{92.78 \pm 0.34}$
CIFAR-10 ResNet-110	40	92.25±0.55	92.23±0.50	92.30±0.37	92.79±0.38	$\mathbf{92.91 \pm 0.41}$
CIFAR-100 ResNet-56	30	68.92 ±0.33	69.41±0.38	70.27±0.40	70.24±0.43	$\mathbf{70.29 \pm 0.46}$
	40	67.69±0.41	67.64±0.50	68.78±0.23	$\mathbf{69.12 \pm 0.22}$	68.88±0.44
CIFAR-100 ResNet-110	30	70.93±0.58	70.94±0.58	71.77±0.60	71.74±0.63	$\mathbf{72.00 \pm 0.21}$
	40	68.41±0.26	69.04±0.41	70.22±0.14	70.05±0.27	$\mathbf{70.34 \pm 0.24}$

Table 2: Results (accuracy) for Taylor filters pruning (Molchanov et al., 2019) on ImageNet. All columns have same meaning with corresponding columns in Table 6.

Model	Unpruned	Prune + FT †	Prune + FT	Prune + CLR	FLOPs ↓ %	Params ↓ %
ResNet-50	76.13	74.50	74.49	**75.22**	45.0%	44.3%

To make a fair comparison between fine-tuning and no fine-tuning, we randomly split the conventional training set of CIFAR-10/CIFAR-100 (including 50000 images) to *train* (90% images of total set) and *val* (10% remaining images) set and we then report the result of best-validation models on "standard" *test* set (including 5000 images).

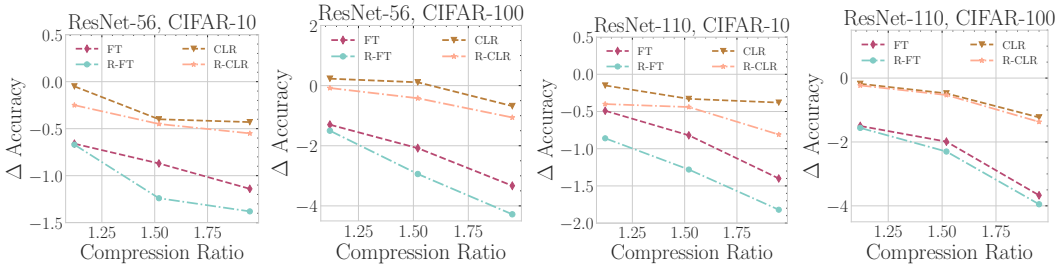
We report the performance of pruned network without fine-tuning, with fine-tuning and with learning rate restarting (CLR) in Table 1. We find that applying fine-tuning on these model usually give very fast convergences speed (roughly in first 20 epochs). However, fine-tuned models exhibit negligible gain then pruned models without fine-tuning. In contrast, CLR significantly improves the performance even with small retraining budgets.

Taylor Pruning (Molchanov et al., 2019). In this method, the authors estimated the contributions of each filter to the final performance by utilizing the first- and second-order Taylor expansion and performing iterative pruning. In the original implementation, the authors fine tuned the network with a small learning rate (0.001) for 25 epochs on ImageNet. In our implementation, we opt to evaluate the accuracy of network retrained with learning rate restarting (CLR) with a relatively small budget. Therefore, we keep the retraining budget the same and only increase the learning rate to 0.01.

Table 2 shows the accuracy of ResNet-50 on ImageNet when pruning 72% of network weights. It can be seen that for a small retraining budget, employing learning rate restarting also gives significant gain in performance.

HRank (Lin et al., 2020). Inspired by the discovery that average rank of feature maps of a filter is consistent, the authors suggested to iteratively remove low-rank feature maps that contain less information. Particularly, we iteratively remove filters in each layer and then retrain the network for 30 epochs each time. We examine the performance of pruned models with HRank on CIFAR while employing CLR at each fine-tuning step. Due to the heavily expensive computational cost for retraining, we only run each model once for HRank. In the original version of this method, the authors used the learning rate of 0.01 and drop by 10× at epochs 5 and 10. In our implementation, we restart the learning rate to 0.1 and apply the CLR schedule.

The quantitative results are shown in Table 6 (Appendix). In our experiments, CLR also attains comparable or better results than proposed retraining methods in original work.



(a) ResNet-56 CIFAR-10 (b) ResNet-56 CIFAR-100 (c) ResNet-110 CIFAR-10 (d) ResNet-110 CIFAR-100

Figure 5: *One-shot structured* pruning on CIFAR-10 dataset using PFEC (Li et al., 2016) and randomly filters pruning with different retraining schemes.

Table 3: Results of networks when applying random pruning and methodically pruning algorithms. "FT" and "CLR" columns present accuracy of pruned networks with original fine-tuning scheme and *CLR* respectively. "R-FT" and "R-CLR" columns demonstrate random pruning with original fine-tuning and CLR. Top- and second-ranked results are highlighted in **bold blue** and blue.

Method	Model	Params ↓	R-FT	FT	R-CLR	CLR
HRank CIFAR-10	ResNet-56	42.4	92.66	92.97	93.11	**93.16**
	ResNet-110	38.7	93.71	93.84	**93.93**	93.90
	DenseNet-40	30.5	93.86	94.10	93.77	94.10
	VGG-16	82.10	92.01	91.97	92.50	**92.53**
MWP (one-shot) CIFAR-10	ResNet-56	49.10	-	93.88 ± 0.09	93.33 ± 0.25	$\mathbf{94.01 \pm 0.12}$
		88.39	-	92.20 ± 0.12	93.16 ± 0.21	$\mathbf{93.19 \pm 0.15}$
PFEC ImageNet	ResNet-18	13.28	69.05	69.16	69.22	**69.42**
	ResNet-34	10.68	72.33	72.46	72.52	**72.54**
		31.47	71.55	71.82	71.98	**72.11**
		36.99	70.77	70.94	71.34	**71.37**

4 Random Pruning with Learning Rate Restarting

Section 3 suggests that learning rate schedule can significantly improve the performance of several pruning algorithms. We notice that there are notable differents between settings of implementation of network pruning beside retraining such as the training scheme of baseline models. In this section, we show that the difference between implementation could easily lead to misleading results and unfair comparisons between pruning algorithms.

Firstly, let us investigate the simple ℓ_1-norm filters pruning on small datasets. Figure 5 illustrates the performance of networks pruned with PFEC and random pruning on CIFAR-10 and CIFAR-100 when retraining with 40 epochs – the same setting used by Li et al. (2016). We can see that randomly pruned networks consistently achieve *superior* performance than methodically pruned networks (fine-tuned with standard learning rate schedule) in terms of accuracy. However, random pruning obtain lower accuracy than PFEC when using identical retraining techniques.

To understand if this result generalizes to sophisticated pruning algoirithms and large-scale datasets, we also conduct experiments with HRank and ImageNet in Table 3. Particularly, we retrain these networks with *largest* learning rate during training, we use retrain budgets of 40 and 45 epochs for ImageNet and CIFAR-10/CIFAR-100 – the smallest number of epochs for networks to converge after restarting (Huang et al., 2017). We can observe that random pruning still can outperform methodical pruning on many occasions even with *same* retraining budgets and only alter the learning rate schedule.

Furthermore, we show that by naively neglecting retraining configurations (i.e. budget), we can also reach state-of-the-art results on CIFAR-10 with PFEC Li et al. (2016). Specifically, we prune two

Table 4: Results of ResNet-56 and ResNet-110 on CIFAR-10. The performance of other pruning algorithms are taken directly from original papers.

Model	Method	Unpruned	Params ↓	FLOPs ↓	Prune	Acc. ↓
ResNet-56	PFEC (Li et al., 2016)	93.04	14.1	27.6	93.06	-0.02
	NISP (Yu et al., 2018)	93.26	42.4	35.5	93.01	0.25
	HRank (Lin et al., 2020)	93.26	42.4	50.0	93.17	0.09
	GAL (Lin et al., 2019)	93.26	-	37.6	92.98	0.28
	PFEC + CLR (ours)	93.21	48.7	-	93.29	-0.07
ResNet-110	PFEC (Li et al., 2016)	93.53	32.6	38.7	93.30	-0.23
	GAL-0.5 (Lin et al., 2019)	93.50	44.8	48.4	92.74	0.76
	HRank (Lin et al., 2020)	93.50	59.2	58.2	93.36	0.14
	PFEC+CLR (ours)	93.56	64.2	-	93.69 ± 0.17	-0.13

"standard" models namely ResNet-56 and ResNet-110 and retrain with CLR for 152 epochs, and then we compare with more sophisticated pruning algorithms in Table 4.

These results suggest that retraining techniques, e.g., learning rate schedule and retraining budget, play a pivotal role to final performance. Thus, in order to perform fair comparison of different methods, one should be cautious of this seemingly subtle detail. While it is unclear to set up a fair comparison between pruning algorithms that belong to different categories, e.g., retraining vs. regularization, iterative vs. oneshot pruning, we advocate standardizing retraining configurations for each algorithm group.

5 Discussion and Conclusion

In this work, we conducted extensive experiments to show that learning rates do matter in achieving good performance of pruned neural networks. We concluded that compared to traditional fine-tuning, learning rate rewinding is an efficient way to retrain pruned networks to recover performance drop due to pruning. The success of learning rate rewinding is accounted by the use of large learning rates in the retraining. We believe that these findings help raise awareness of proper use of learning rate schedule when desiging pruning algorithms, standardizing empirical experiments and allowing fair comparisons. Our takeaway message is:

Pruning algorithms should be compared in the same retraining configurations.

Our work is not without limitations. So far we investigated with hyperparameters identical to those of the original implementations of the pruning algorithms, and only experiment with different learning rate schedules. We also limited our experiments to the image classification task in computer vision. Considering more datasets and other domains is a great research avenue for future works.

References

Sanjeev Arora, Nadav Cohen, and Elad Hazan. On the optimization of deep networks: Implicit acceleration by overparameterization. *arXiv preprint arXiv:1802.06509*, 2018.

Alon Brutzkus and Amir Globerson. Why do larger models generalize better? a theoretical perspective via the xor problem. In *International Conference on Machine Learning*, pp. 822–830. PMLR, 2019.

Shangqian Gao, Feihu Huang, Jian Pei, and Heng Huang. Discrete model compression with resource constraint for deep neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1899–1908, 2020.

Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In *Advances in neural information processing systems*, pp. 1135–1143, 2015.

Yang He, Guoliang Kang, Xuanyi Dong, Yanwei Fu, and Yi Yang. Soft filter pruning for accelerating deep convolutional neural networks. *arXiv preprint arXiv:1808.06866*, 2018.

Gao Huang, Yixuan Li, Geoff Pleiss, Zhuang Liu, John E Hopcroft, and Kilian Q Weinberger. Snapshot ensembles: Train 1, get m for free. *arXiv preprint arXiv:1704.00109*, 2017.

Yann LeCun, John S Denker, and Sara A Solla. Optimal brain damage. In *Advances in neural information processing systems*, pp. 598–605, 1990.

Bailin Li, Bowen Wu, Jiang Su, Guangrun Wang, and Liang Lin. Eagleeye: Fast sub-net evaluation for efficient neural network pruning. *arXiv preprint arXiv:2007.02491*, 2020.

Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710*, 2016.

Mingbao Lin, Rongrong Ji, Yan Wang, Yichen Zhang, Baochang Zhang, Yonghong Tian, and Ling Shao. Hrank: Filter pruning using high-rank feature map. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1529–1538, 2020.

Shaohui Lin, Rongrong Ji, Chenqian Yan, Baochang Zhang, Liujuan Cao, Qixiang Ye, Feiyue Huang, and David Doermann. Towards optimal structured cnn pruning via generative adversarial learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2790–2799, 2019.

Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. Rethinking the value of network pruning. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=rJlnB3C5Ym`.

Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. Pruning convolutional neural networks for resource efficient inference. *arXiv preprint arXiv:1611.06440*, 2016.

Pavlo Molchanov, Arun Mallya, Stephen Tyree, Iuri Frosio, and Jan Kautz. Importance estimation for neural network pruning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 11264–11272, 2019.

Behnam Neyshabur, Zhiyuan Li, Srinadh Bhojanapalli, Yann LeCun, and Nathan Srebro. Towards understanding the role of over-parametrization in generalization of neural networks. *arXiv preprint arXiv:1805.12076*, 2018.

Roman Novak, Yasaman Bahri, Daniel A Abolafia, Jeffrey Pennington, and Jascha Sohl-Dickstein. Sensitivity and generalization in neural networks: an empirical study. *arXiv preprint arXiv:1802.08760*, 2018.

Alex Renda, Jonathan Frankle, and Michael Carbin. Comparing rewinding and fine-tuning in neural network pruning. *arXiv preprint arXiv:2003.02389*, 2020.

Leslie N Smith and Nicholay Topin. Super-convergence: Very fast training of neural networks using large learning rates. In *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications*, volume 11006, pp. 1100612. International Society for Optics and Photonics, 2019.

Zhonghui You, Kun Yan, Jinmian Ye, Meng Ma, and Ping Wang. Gate decorator: Global filter pruning method for accelerating deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pp. 2133–2144, 2019.

Ruichi Yu, Ang Li, Chun-Fu Chen, Jui-Hsin Lai, Vlad I Morariu, Xintong Han, Mingfei Gao, Ching-Yung Lin, and Larry S Davis. Nisp: Pruning networks using neuron importance score propagation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 9194–9203, 2018.

Zhuangwei Zhuang, Mingkui Tan, Bohan Zhuang, Jing Liu, Yong Guo, Qingyao Wu, Junzhou Huang, and Jinhui Zhu. Discrimination-aware channel pruning for deep neural networks. In *Advances in Neural Information Processing Systems*, pp. 875–886, 2018.

A APPENDIX

A.1 NETWORK PRUNING FORMULATION

Let us start with a formal definition of the network pruning problem. Let $\boldsymbol{\theta} \in \mathbb{R}^D$ be the parameters of a (large) neural network that needs to be pruned, $\mathcal{D}$ be the dataset for training. The parameters $\boldsymbol{\theta}$ is updated to minimize the loss function $\mathcal{L}(\boldsymbol{\theta}; \mathcal{D})$. Define *sparsity mask* $\mathbf{m} \in \{0, 1\}^D$ indicating if a weight should be kept or removed. Similar to Molchanov et al. (2016), we can formulate the pruning algorithms as finding the optimal sparsity mask such that:

$$\mathbf{m}^* = \underset{\mathbf{m}}{\operatorname{argmin}} \;\; \mathcal{L}(\boldsymbol{\theta}^* \odot \mathbf{m}; \mathcal{D}) - \mathcal{L}(\boldsymbol{\theta}^*; \mathcal{D}) \tag{1}$$

where $\odot$ denotes the Hadamard (element-wise) product, $\boldsymbol{\theta}^*$ represents the (locally) optimal solution of $\boldsymbol{\theta}$. An alternative approach is minimizing the different between final output of two networks (Lin et al., 2019):

$$\mathbf{m}^* = \underset{\mathbf{m}}{\operatorname{argmin}} \;\; \|\mathcal{F}(\boldsymbol{\theta}^* \odot \mathbf{m}; \mathcal{D}) - \mathcal{F}(\boldsymbol{\theta}^*; \mathcal{D})\|_2^2 \tag{2}$$

where $\mathcal{F} : \mathbb{R}^D \rightarrow \mathbb{R}^{D'}$ is the function mapping input feature (i.e. images) to the final output (before softmax).

Theoretically speaking, as the pruning formulation only takes the loss function into account, and there is a caveat that pruning can result in a set of weights that belong to a bad local minima, making the network generalize poorly. In practice, it is common to observe such issues from the performance drop of the pruned model. In such cases, a fine-tuning process can be applied to further optimize the weights, mitigating the accuracy drop. However, as what fine-tuning means, the weights might be only slightly adjusted without any guarantee about the final network performance.

A.2 TRAINING CONFIGURATION

The implementation of ℓ_1-norm filters pruning (PFEC) Li et al. (2016) and magnitude-based weights pruning (MWP) Han et al. (2015) are adopted from the work of Liu et al. (2019). The implementations of other algorithms are taken from the authors' official repositories.

For simplicity, we adopt Pytorch's pretrained models for ImageNet. The unpruned models (used for ℓ_1-norm filters pruning) are trained with below configurations.

Table 5: Training configuration for unpruned models. To train CIFAR-10, we use Nesterov SGD with $\beta = 0.9$, batch size 64, weight decay 0.0001 for 160 epochs. To train ImageNet, we use Nesterov SGD with $\beta = 0.9$, batch size 32, weight decay 0.0001 for 90 epochs.

Dataset	Network	#Params.	Learning rate (t = training epoch)	Test accuracy
CIFAR-10	ResNet-56	0.85M	$\alpha = \begin{cases} 0.1 & t \in [0, 80) \\ 0.01 & t \in [80, 120) \\ 0.001 & t \in [120, 160) \end{cases}$	$93.46 \pm 0.21\%$
	ResNet-110	1.73M		
ImageNet	ResNet-18	11.69M	$\alpha = \begin{cases} 0.1 & t \in [0, 30) \\ 0.01 & t \in [30, 60) \\ 0.001 & t \in [60, 90) \end{cases}$	69.76% top-1
	ResNet-34	21.8M		73.30% top-1
	ResNet-50	25.5M		76.15% top-1

A.3 ADDITIONAL RESULTS

In addition results in Figure 2, we conduct experiments to examine impact of different retraining techniques to networks pruned with low and high compression ratios. The results are shown in Figure 6.

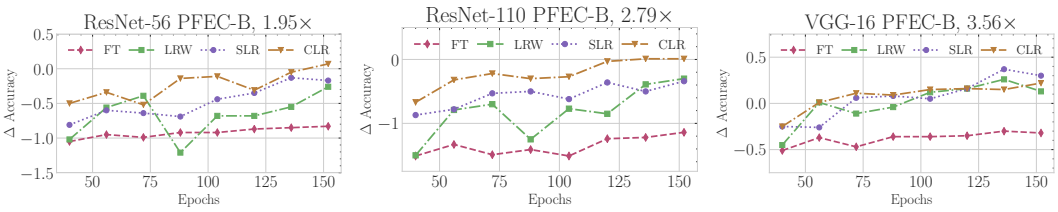


Figure 6: Results from pruning with high compression ratios for *one-shot structured* pruning on CIFAR-10 dataset using ℓ_1-norm pruning (Li et al., 2016). With a proper learning rate schedule, it is possible to achieve almost no accuracy drop while having much more compact models and using slightly more training budgets.

Table 6: Results (accuracy) for HRank ilters pruning (Lin et al., 2020) on CIFAR-10. "Pruned + FT†" is the model pruned from the large model with *original* fine-tuning scheme that are reported. "Prune + FT" and "Prune + CLR" are results of our runs with original retraining and CLR respectively. Configurations of Model and Pruned Model are both from the original paper.

Model	Unpruned	Prune + FT †	Prune + FT	Prune + CLR	FLOPs ↓ %	Params ↓ %
VGG-16 (BN)	93.93	92.34	91.97	**92.53**	65.3%	82.10%
ResNet-56	93.26	93.17	92.97	**93.16**	50.0%	42.4%
ResNet-110	93.50	94.23	93.84	**93.90**	37.9%	38.7%
DenseNet-40	94.81	94.24	94.10	94.10	40.8%	30.5%