

Do We Actually Need Dense Over-Parameterization? In-Time Over-Parameterization in Sparse Training

Shiwei Liu¹ Lu Yin¹ Decebal Constantin Mocanu^{1,2} Mykola Pechenizkiy¹

Abstract

In this paper, we introduce a new perspective on training deep neural networks capable of state-of-the-art performance without the need for the expensive over-parameterization by proposing the concept of In-Time Over-Parameterization (ITOP) in sparse training. By starting from a random sparse network and continuously exploring sparse connectivities during training, we can perform an Over-Parameterization over the course of training, closing the gap in the expressibility between sparse training and dense training. We further use ITOP to understand the underlying mechanism of Dynamic Sparse Training (DST) and discover that the benefits of DST come from its ability to consider across time all possible parameters when searching for the optimal sparse connectivity. As long as sufficient parameters have been reliably explored, DST can outperform the dense neural network by a large margin. We present a series of experiments to support our conjecture and achieve the state-of-the-art sparse training performance with ResNet-50 on ImageNet. More impressively, ITOP achieves dominant performance over the overparameterization-based sparse methods at extreme sparsities. When trained with ResNet-34 on CIFAR-100, ITOP can match the performance of the dense model at an extreme sparsity of 98%.

1. Introduction

Over-Parameterization has been shown to be crucial to the dominating performance of deep neural networks in practice, despite the fact that the training objective function

¹Department of Mathematics and Computer Science, Eindhoven University of Technology, 5600 MB Eindhoven, the Netherlands

²Faculty of Electrical Engineering, Mathematics and Computer Science, University of Twente, Enschede 7522NB, The Netherlands
. Correspondence to: Shiwei Liu <s.liu3@tue.nl>.

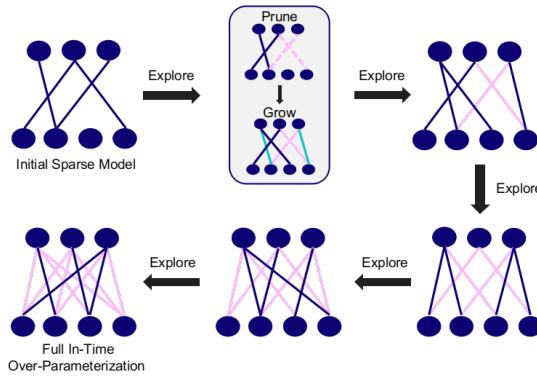


Figure 1. As the figure proceeds, we perform an Over-Parameterization in time. Blue lines refer to the currently activated connections. Pink lines are the connections that have been activated previously. While exploring In-Time Over-Parameterization, the parameter count (blue lines) of the sparse model is fixed throughout training.

is usually non-convex and non-smooth (Goodfellow et al., 2015; Brutzkus et al., 2017; Li & Liang, 2018; Safran & Shamir, 2018; Soudry & Carmon, 2016; Allen-Zhu et al., 2019; Du et al., 2019; Zou et al., 2020; Zou & Gu, 2019). Meanwhile, advanced deep models (Simonyan & Zisserman, 2014; He et al., 2016; Devlin et al., 2018; Brown et al., 2020; Dosovitskiy et al., 2021) are continuously achieving state-of-the-art results in numerous machine-learning tasks. While achieving impressive performance, the size of the state-of-the-art models is also exploding. The resources required to train and deploy those highly over-parameterized models are prohibitive.

Motivated for inference, a large body of research (Mozer & Smolensky, 1989; Han et al., 2015) attempts to discover a sparse model that can sufficiently match the performance of the corresponding dense model while substantially reduce the number of parameters. While effective, these techniques involve pre-training a highly over-parameterized model for either at least a full converged training time (full dense over-parameterization) (Janowsky, 1989; LeCun et al., 1990; Hassibi & Stork, 1993; Molchanov et al., 2017; Han et al., 2016; Gomez et al., 2019; Dai et al., 2018a) or a partial converged training time (partial dense over-parameterization) (Louizos et al., 2017; Zhu & Gupta, 2017; Gale et al., 2019; Savarese et al., 2019; Kusupati et al., 2020; You et al., 2019).

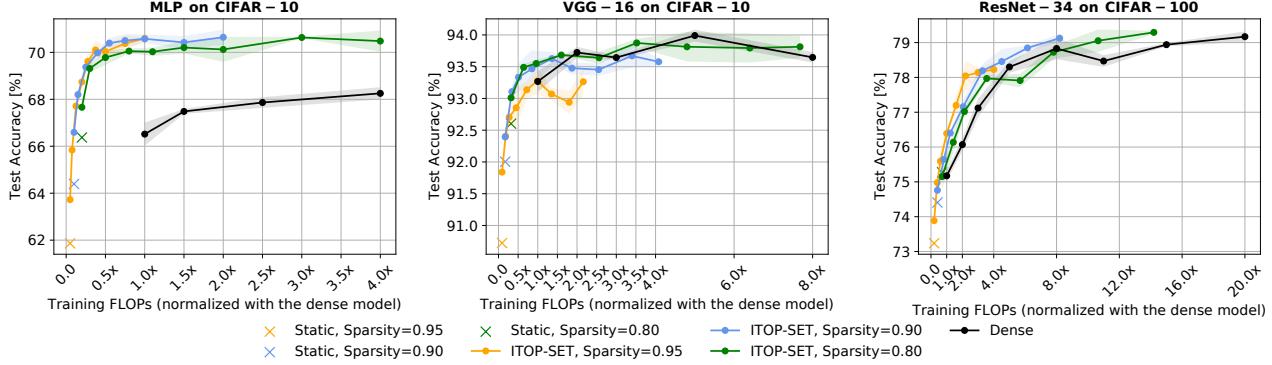


Figure 2. Training FLOPs of sparse models trained with In-Time Over-Parameterization. $\Delta T = 1500$ and batch size is 128.

Given the fact that the training costs of state-of-the-art models e.g., GPT-3 (Brown et al., 2020) and Vision Transformer (Dosovitskiy et al., 2020), have increasingly exploded, this heavily over-parameterized dependency leads to a situation where state-of-the-art models are beyond the reach of the majority of the machine learning community.

Recently, the lottery ticket hypothesis (LTH) (Frankle & Carbin, 2019) shows the possibility to train a sub-network from scratch (sparse training) to match the performance of the dense network. However, these “winning tickets” are found under the guidance of a fully dense over-parameterized process (iterative pruning a fully converged network), and solutions that are discovered through either a partial dense over-parameterization (pruning at initialization (Lee et al., 2019; Wang et al., 2020; de Jorge et al., 2020)) or no over-parameterization (randomly-initialized static sparse training (Mocanu et al., 2016; Evci et al., 2019)) typically are not able to match the accuracy achieved by their dense counterpart. A common-sense explanation would be that in comparison with dense training, sparse training, especially at extreme high sparsities, does not have the over-parameterization property, and hence suffers from a poor expressibility. One approach to address this problem is to leverage the knowledge learned from dense training, e.g., LTH (Frankle & Carbin, 2019). While effective, the computational costs and memory requirements attached to the over-parameterized dense training are prohibitive.

1.1. Our Contribution

In this paper, we propose a concept that we call *In-Time Over-Parameterization*¹ to close the gap in over-parameterization along with expressibility between sparse training and dense training, illustrated in Figure 1. Instead of inheriting weights from a dense and pre-trained model, allowing a continuous parameter exploration across the training time performs an over-parameterization in the

¹<https://github.com/Shiweiliuiiiiiii/In-Time-Over-Parameterization>

space-time manifold, which can significantly improve the expressibility of sparse training.

We find the concept of In-Time Over-Parameterization useful (1) in exploring the expressibility of sparse training, especially for extreme sparsities, (2) in reducing training and inference costs (3) in understanding the underlying mechanism of dynamic sparse training (DST) (Mocanu et al., 2018; Evci et al., 2020a), (4) in preventing overfitting and improving generalization.

Based on In-Time Over-Parameterization, we improve the state-of-the-art sparse training performance with ResNet-50 on ImageNet. We further assess the ITOP concept by applying it to the main class of sparse training methods, DST, in comparison with the overparameterization-based sparse methods including LTH, gradual magnitude pruning (GMP), and pruning at initialization (PI). Our results show that, when a sufficient and reliable parameter exploration is reached (as required by ITOP), DST consistently outperforms those overparameterization-based methods. Since ITOP eliminates the dense Over-Parameterization throughout the whole course of training, it can match the performance of the corresponding dense networks with much fewer training FLOPs, as shown in Figure 2.

2. Related Work

2.1. Dense Over-Parameterization

Sparsity-inducing techniques that depend on dense over-parameterization (dense-to-sparse training) have been extensively studied. We divide them into three categories according to their degrees of dependence on the dense over-parameterization.

Full dense over-parameterization. Techniques sought to inherit weights from a fully pre-trained dense model have a long history and were first introduced by Janowsky (1989) and Mozer & Smolensky (1989), autonomously evolving as the iterative pruning and retaining method. The basic idea of iterative pruning and retaining involves a three-step pro-

cess: (1) fully pre-training a dense model until converged, (2) pruning the weights or the neurons that have the lowest influence on the performance, and (3) re-training the pruned model to further improve the performance. The pruning and retraining cycle is required at least once (Liu et al., 2019), and usually many times (Han et al., 2016; Guo et al., 2016; Frankle & Carbin, 2019). The criteria used for pruning includes but are not limited to magnitude (Mozer & Smolensky, 1989; Han et al., 2016; Guo et al., 2016), Hessian (LeCun et al., 1990; Hassibi & Stork, 1993), mutual information (Dai et al., 2018a), Taylor expansion (Molchanov et al., 2016; 2019). Except for pruning, other techniques including variational dropout (Molchanov et al., 2017), targeted dropout (Gomez et al., 2019), reinforcement learning (Lin et al., 2017) also yield a sparse model from a pre-trained dense model.

Partial dense over-parameterization. Another class of methods start from a dense network and continuously sparsify the model during training. Gradual magnitude pruning (GMP) (Narang et al., 2017; Zhu & Gupta, 2017; Gale et al., 2019) was proposed to reduce the number of pruning-and-retaining rounds by pruning the dense network to the target sparsity gradually over the course of training. There are some examples Louizos et al. (2017) and Wen et al. (2016) that utilize L_0 and L_1 regularization to gradually learn the sparsity by explicitly penalizing parameters for being different from zero, respectively. Recently, Srinivas et al. (2017); Liu et al. (2020a); Savarese et al. (2019); Xiao et al. (2019); Kusupati et al. (2020); Zhou et al. (2021) moved further by introducing trainable masks to learn the desirable sparse connectivity during training. Since these techniques start from a dense model, the training cost is smaller than training a dense network, depending on the stage at which the final sparse models are learned.

One-Shot dense over-parameterization. Very recently, works on pruning at initialization (PI) (Lee et al., 2019; 2020; Wang et al., 2020; Tanaka et al., 2020; de Jorge et al., 2021) have emerged to obtain trainable sparse neural networks before the main training process based on some salience criteria. These methods fall into the category of dense over-parameterization mainly because the dense model is required to train for at least one iteration to obtain those trainable sparse networks.

2.2. In-Time Over-Parameterization

Dynamic Sparse Training. Evolving in parallel with LTH, DST is a growing class of methods to train sparse networks from scratch with a fixed parameter count throughout training (sparse-to-sparse training). This paradigm starts from a (random) sparse neural network and allows the sparse connectivity to evolve dynamically during training. It has been first introduced in Mocanu (2017) and became well-

established in Mocanu et al. (2018) by proposing the Sparse Evolutionary Training (SET) algorithm which achieves better performance than static sparse neural networks. In addition to the proper classification performance, it also helps to detect important input features (Atashgahi et al., 2020). Bellec et al. (2018) proposed Deep Rewiring to train sparse neural networks with a strict connectivity constraint by sampling sparse configurations and weights from a posterior distribution. Follow-up works further introduced weight redistribution (Mostafa & Wang, 2019; Dettmers & Zettlemoyer, 2019; Liu et al., 2021), gradient-based weight growth (Dettmers & Zettlemoyer, 2019; Evcı et al., 2020a), and extra weights update in the backward pass (Raihan & Aamodt, 2020; Jayakumar et al., 2020) to improve the sparse training performance. By relaxing the constraint of the fixed parameter count, Dai et al. (2019; 2018b) proposed a grow-and-prune strategy based on gradient-based growth and magnitude-based pruning to yield an accurate, yet very compact sparse network. More recently, Liu et al. (2020b) illustrated for the first time the true potential of using dynamic sparse training. By developing an independent framework, they can train truly sparse neural networks without masks with over one million neurons on a typical laptop.

Understanding Dynamic Sparse Training. Concurrently, some works attempt to understand Dynamic Sparse Training. Liu et al. (2020c) found that DST gradually optimizes the initial sparse topology towards a completely different one. Although there exist many low-loss sparse solutions that can achieve similar loss, they are very different in the topological space. Evcı et al. (2020b) found that sparse neural networks that are initialized by a dense initialization e.g., He et al. (2015), suffer from a poor gradient flow, whereas DST can improve the gradient flow during training significantly. Although promising, the capability of sparse training has not been fully explored and the mechanism underlying DST is not clear yet. Questions like: *Why Dynamic Sparse Training can improve the performance of sparse training? How Dynamic Sparse Training can enable sparse neural network models to match - and even to outperform - their dense counterparts?* are required to be answered.

3. In-Time Over-Parameterization

In this section, we describe in detail In-Time Over-Parameterization, a concept that we proposed to be an alternative way to train deep neural networks without the expensive over-parameterization. We refer In-Time Over-Parameterization as a variant of dense over-parameterization, which can be achieved by encouraging a continuous parameter exploration across the training time. Note that different from the over-parameterization of dense models which refers to the spatial dimensionality of the parameter space, In-Time Over-Parameterization refers to the overall dimen-

sionality explored in the space-time manifold.

3.1. In-Time Over-Parameterization Hypothesis

Based on In-Time Over-Parameterization, we propose the following hypothesis to understand Dynamic Sparse Training:

Hypothesis. *The benefits of Dynamic Sparse Training come from its ability to consider across time all possible parameters when searching for the optimal sparse neural network connectivity.* Concretely, this hypothesis can be divided into three main pillars which can explain the performance of DST:

1. Dynamic Sparse Training can significantly improve the performance of sparse training mainly due to the parameter exploration across the training time.
2. The performance of Dynamic Sparse Training is highly related to the total number of the reliably explored parameters throughout training. The reliably explored parameters refer to those newly-explored (newly-grown) weights that have been updated for long enough to exceed the pruning threshold.
3. As long as there are sufficient parameters that have been reliably explored, sparse neural network models trained by Dynamic Sparse Training can match or even outperform their dense counterparts by a large margin, even at extremely high sparsity levels.

We name our hypothesis as In-Time Over-Parameterization hypothesis for convenience.

Formally, given a dataset containing N samples $\mathbf{D} = \{(x_i, y_i)\}_{i=1}^N$ and a dense network $f(x; \theta)$ parameterized by θ . We train the dense network to minimize the loss function $\sum_{i=1}^N L(f(x_i; \theta), y_i)$. When optimizing with a certain optimizer, $f(x; \theta)$ reaches a minimum validation loss function l with a test accuracy a . Differently, sparse training starts with a sparse neural network $f(x; \theta_s)$ parameterized by a fraction of parameters θ_s . The basic mechanism of Dynamic Sparse Training is to train the sparse neural network $f(x; \theta_s)$ to minimize the loss $\sum_{i=1}^N L(f(x_i; \theta_s), y_i)$ while periodically update the sparse connectivity θ_s every ΔT iterations based on some criteria. $f(x; \theta_s^u)$ reaches a minimum validation loss l' at sparse connectivity update u with a test accuracy a' , where θ_s^u is the sparse connectivity parameters obtained at the iteration u . Let us denote R_s as the ratio of the total number of reliably explored parameters during training to the total number of parameters, or simply In-Time Over-Parameterization rate, computed as $R_s = \frac{\|\theta_s^1 \cup \theta_s^2 \cup \dots \cup \theta_s^u\|_0}{\|\theta\|_0}$, where $\|\cdot\|_0$ is the ℓ_0 -norm.

Our hypothesis states that when $\Delta T \geq T_0$, $\exists R_0$ as long as $R_s \geq R_0$, for which $a' \geq a$ (commensurate accuracy) and

$\|\theta_s^u\|_0 \ll \|\theta\|_0$ (fewer parameters in the final sparse model), where T_0 is the minimum threshold of update interval to guarantee the reliable parameter exploration, and R_0 is the threshold of In-Time Over-Parameterization rate where DST can match the performance of the dense model.

Similar to dense training, there are many factors affecting the performance of dynamic sparse training, learning rate, batch size, regularization, optimizers, sparsity distribution, etc. In this paper, we limit our study to parameter exploration, since it is the fundamental difference between dynamic sparse training and static sparse training. By “reliable”, we mainly focus on the newly-activated weights that are updated for a long time (guaranteed by $\Delta T \geq T_0$) so that they are not pruned in the next update iteration. We believe this is a good starting point, since even in this simple setting, our community does not have a satisfactory answer.

3.2. Hypothesis Evaluation

In this section, we work through the In-Time Over-Parameterization hypothesis and study the effect of In-Time Over-Parameterization on the performance of DST. We choose Sparse Evolutionary Training (SET) as our DST method as SET activates new weights in a random fashion which naturally considers all possible parameters to explore. It also helps to avoid the dense over-parameterization bias introduced by the gradient-based methods e.g., The Rigged Lottery (RigL) (Evci et al., 2020a) and Sparse Networks from Scratch (SNFS) (Dettmers & Zettlemoyer, 2019), as the latter utilize dense gradients in the backward pass to explore new weights. To work through the proposed hypothesis, we conduct a set of step-wise fashion experiments with image classification. We study Multi-layer Perceptron (MLP) on CIFAR-10, VGG-16 on CIFAR-10, ResNet-34 on CIFAR-100, and ResNet-50 on ImageNet. We use PyTorch as our library. All results are averaged from three different runs and reported with the mean and standard deviation. See Appendix A for the experimental details.

3.2.1. TYPICAL TRAINING TIME

Our first evaluation of the In-Time Over-Parameterization hypothesis is to see what happens when different over-parameterization rates R_s are reached during training within a typical training time (200 or 250 epochs). A direct way to control R_s is to vary ΔT , a hyperparameter that determines the update interval of sparse connectivities (the number of iterations between two sparse connectivity updates). We train MLP, VGG-16, and ResNet-34 with various ΔT and report the test accuracy.

Expected results. Gradually decreasing ΔT will explore more parameters, and thus lead to increasingly higher test accuracy. However, when ΔT gets smaller than the reliable exploration threshold T_0 , the test accuracy will start

In-Time Over-Parameterization

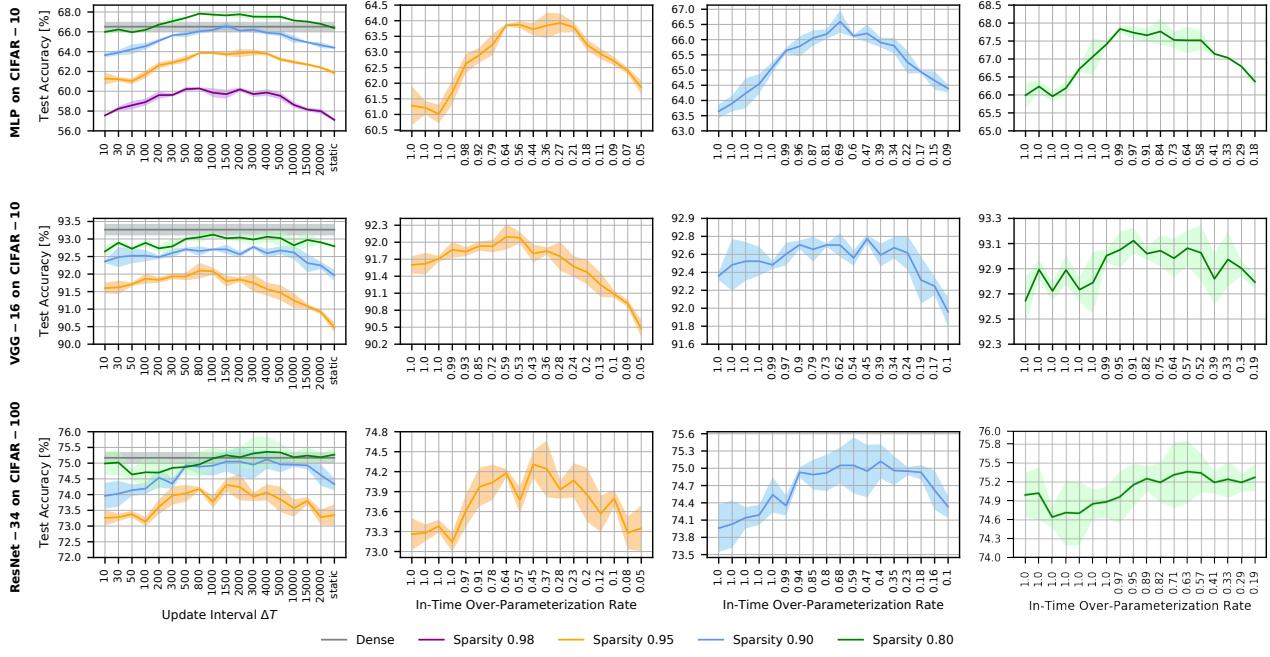


Figure 3. Effect of In-Time Over-Parameterization on sparse training MLPs (**top**), VGG-16 (**middle**), and ResNet-34 (**bottom**) with a typical training time. All sparse models are trained with SET. Each line is averaged from three different runs. “Static” refers to the static sparse training without parameter exploration.

to decrease since the new weights can not receive enough updates to exceed the pruning threshold.

Experimental results. For a better overview, we plot the performance achieved at different sparsities together in the leftmost column of Figure 3. To understand better the relationship between R_s and test accuracy, we report the final R_s associated with ΔT separately in the rest columns of Figure 3.

Overall, a similar pattern can be found existing in all lines. Starting from the static sparse training, sparse training consistently benefits from the increased R_s as ΔT decreases. However, the test accuracy starts to drop rapidly after it reaches a peak value, especially at high sparsities (yellow and blue lines). For example, even if MLPs and ResNet-34 eventually reach a 100% exploration rate with extremely small ΔT values (e.g., 10, 30), their performance is much worse than the static sparse training. This behavior is perfectly in line with our hypothesis. While small ΔT encourages sparse models to maximally explore the search space spanned over the dense model, the benefits provided by In-Time Over-Parameterization is heavily limited by the unreliable parameter exploration. Interestingly, the negative effect of the unreliable exploration on lower sparsities (green lines) is less than the one on high sparsities (yellow lines). We regard this as trivial sparsities (Frankle et al., 2020a) as the remaining models are still over-parameterized to fit the data.

3.2.2. EXTENDED TRAINING TIME

Until now, we have already learned the trade-off between test accuracy and R_s for the typical training time. A direct approach to alleviating this trade-off is to extend the training time while using large ΔT . We train MLP, VGG-16, and ResNet-34 for an extended training time with a large ΔT . We safely choose ΔT as 1500 for MLPs, 2000 for VGG-16, and 1000 for ResNet-34 according to the trade-off shown in Figure 3. In addition to the training time, the anchor points of the learning rate schedule are also scaled by the same factor.

Expected results. In this setting, we expect that, in addition to the benefits brought by the extended training time, sparse training would benefit significantly from the increased R_s .

Experimental results. The results are shown in Figure 4. Static sparse training without parameter exploration consistently achieves the lowest accuracy. However, all models at different sparsities substantially benefit from an extended training time accompanied by an increased R_s . In other words, reliably exploring the parameter space in time continuously improves the expressibility of sparse training. Importantly, after matching the performance of the dense baseline (black line), the performance of sparse training continues to improve, yielding a notable improvement over the dense baseline. Furthermore, the models with lower sparsities require less time to match their full accuracy plateau than

In-Time Over-Parameterization

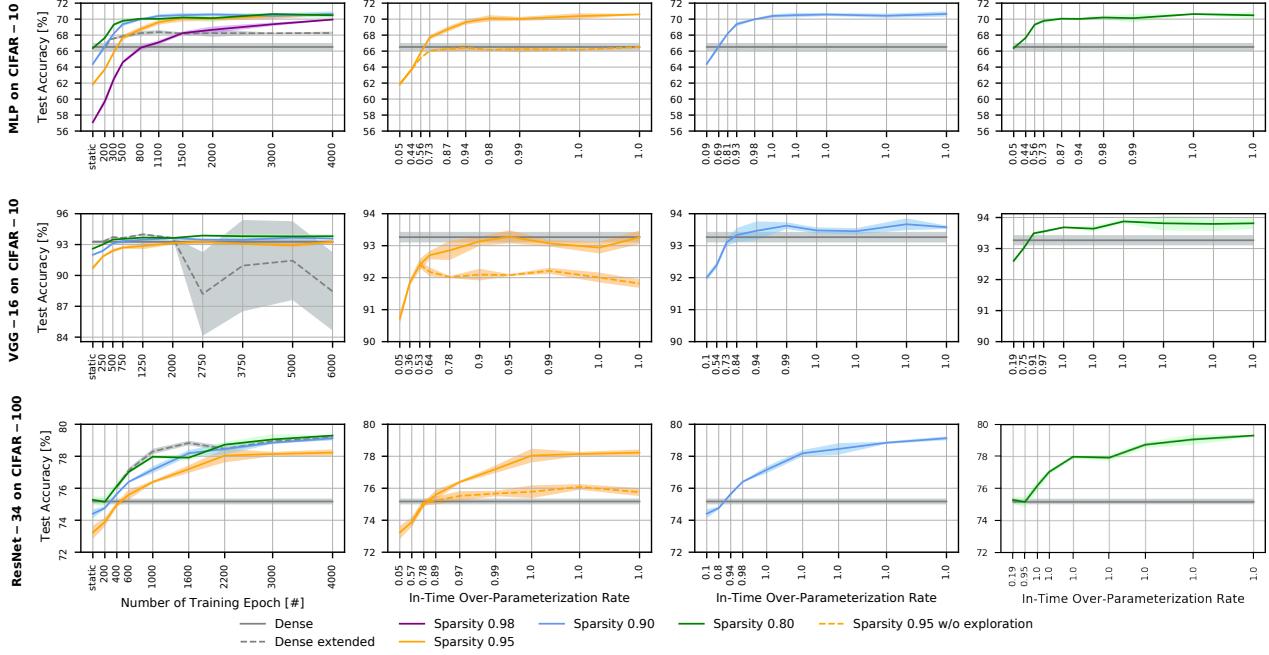


Figure 4. Effect of In-Time Over-Parameterization on sparse training MLPs (**top**), VGG-16 (**middle**) and ResNet-34 (**bottom**) with an extended training time. All sparse models are trained with SET. Each line is averaged from three different runs. ‘‘Static’’ refers to the static sparse training without parameter exploration. ‘‘Dense extended’’ refers to training a dense model for an extended time. ‘‘Sparsity 0.95 w/o exploration’’ means we train the model for the same extended time but stop exploring parameters after a typical training time (200 or 250 epochs).

higher sparsities; the cause appears to be that models with lower sparsity can explore more parameters in the same training time.

To show that the performance gains are not only caused by the longer training time, we make a controlled experiment by stopping the parameter exploration immediately after the typical training time (the sparse connectivity remains fixed after a typical training time), shown as the orange dashed lines. As we can see, even though improved, the accuracy is much lower than the accuracy achieved by In-Time Over-Parameterization.

We also report the performance of dense models with an extended training time as the dashed black lines. Training a dense model with an extended time leads to either inferior (MLPs and VGG-16), or equal solutions (ResNet-34). Different from the dense over-parameterization where overfitting usually occurs when the model has been overtrained for long enough, the test accuracy of dynamic sparse training is continuously increasing as R_s increases until a plateau is reached with a full In-Time Over-Parameterization. This observation highlights the advantage of In-Time Over-Parameterization to prevent overfitting over the dense over-parameterization.

4. Effect of Hyperparameter Choices

4.1. Effect of Weight Growth methods on ITOP

We next investigate the effect of gradient-based weight growth (used in RigL and SNFS) and random-based weight growth (used in SET) on In-Time Over-Parameterization. Since gradient-based methods have access to the dense over-parameterization in the backward pass (occasionally using dense gradients to activate new weights), we hypothesize that they can reach a converged accuracy without a high R_s . We make a comparison between RigL and SET for both the typical training and the extended training in Figure 5. We study them on MLPs where the model size is relatively small so that we can easily achieve a full In-Time Over-Parameterization and have a better understanding of these two methods.

Typical Training Time. It is clear that RigL also heavily suffers from the unreliable exploration. As ΔT decreases, the test accuracy of RigL presents a trend of rising, falling, and rising again. Compared with the random-based growth, RigL receives larger gains from the reliable parameter exploration and also a larger forfeit from the unreliable exploration. These differences are potentially due to that RigL grows new weights with high gradient magnitude, which leads to a faster loss decrease when the exploration is faithful, but also requires a higher ΔT to guarantee a faithful

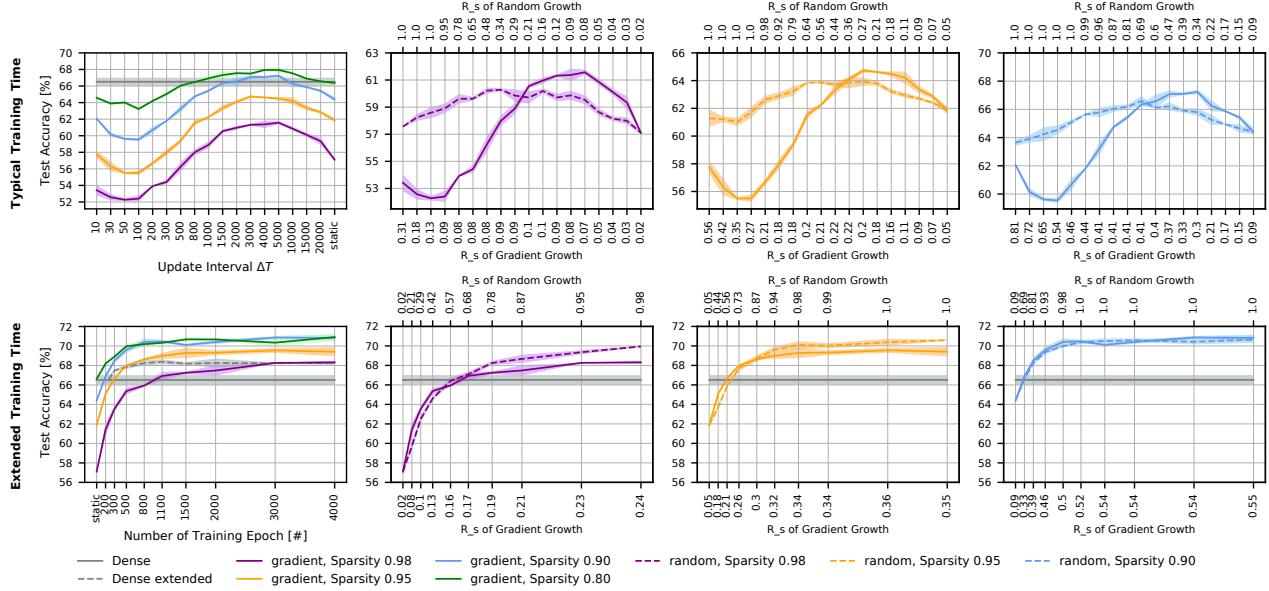


Figure 5. Comparisons between RigL and SET with MLP on CIFAR-10. We vary the update interval ΔT for the typical training time setting, and keep it fixed for the extended training time setting (1500 for SET and 4000 for RigL).

exploration as the weight with large gradients is likely to end up with high magnitude, resulting in a large pruning threshold.

Extended Training Time. For RigL, we choose $\Delta T = 4000$ to ensure the reliable exploration (the performance of RigL with a smaller $\Delta T = 1500$ is much worse as shown in Appendix D). We can see that RigL also significantly benefits from an increased R_s . Surprisingly, although RigL achieves higher accuracy than the SET with a limited training time, it ends up with lower accuracy than SET with a sufficient training time. From the perspective of R_s , we can see that the R_s of RigL is much smaller than SET, indicating that gradient weight growth drives the sparse connectivity into some similar structures and in turn limits its expressibility. On the contrary, random growth naturally considers the whole search space to explore parameters and has a larger possibility of finding better local optima. Similar results are also reported for sparse Recurrent Neural Networks (RNNs) in Liu et al. (2021). However, similar results are not shared with large-scale architectures on large datasets. For instance, RigL achieves better performance than SET with ResNet-50 on ImageNet. This result is reasonable since the dense gradients help RigL easily find the most promising weights at each sparse connectivity update. In contrast, it would take a much longer time (high R_s) for SET (random weight growth) to discover these promising weights within large-scale architectures, especially at high sparsities.

4.2. Effect of Batch Size on ITOP

Intuitively, our hypothesis uncovers ways to improve the existing DST methods within a limited training time. A direct

way to reliably explore more parameters within a typical training time is to train with a small batch size. Using a smaller batch size equally means having more updates, and therefore leads to a higher R_s . We simply demonstrate the effectiveness of this conjecture on SET with $\Delta T = 1500$ in Figure 6 (see Appendix E for RigL). With a large batch size, the parameter exploration is insufficient to achieve a high In-Time Over-Parameterization rate R_s , and the test accuracy is subsequently much lower than the dense model. As we expected, the reduction in batch size consistently increases R_s as well as the test accuracy, until the batch size gets smaller than 16. However, the performance of the dense model remarkably decreases as the batch size decreases. More interestingly, when the batch size is smaller than 16, the performance of sparse models flips and the sparsest model starts to achieve the highest accuracy. The performance drop is likely caused by the increased ‘noise scale’ of SGD where extremely small batch sizes lead to large noise scale and large accuracy drop (Smith et al. (2017)).

4.3. Effect of Pruning Rate on ITOP

The initial pruning rate (denoted as P) of parameter exploration also affects the overall number of parameters visited during training. Relatively large pruning rates encourage a large range of exploration, resulting in higher accuracy, whereas a too-large pruning rate hurts the model capacity as it prunes too many parameters. We confirm this with ResNet-18 on CIFAR-10 trained with various initial pruning rates $P \in [0.1, 0.3, 0.5, 0.7, 0.9]$ as shown in Table 1. The similar pattern as we expected is shared across all update intervals.

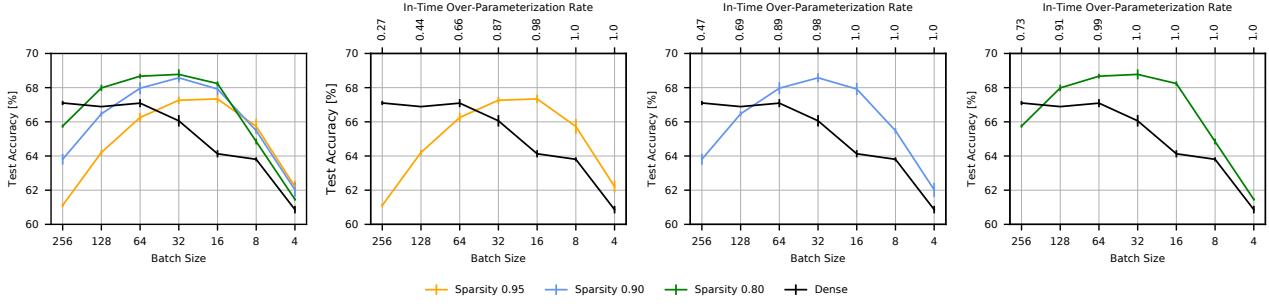


Figure 6. Test accuracy of SET with various batch sizes. The update interval ΔT is set as 1500.

Table 1. Performance of sparse ResNet-18 on CIFAR-10 with various pruning rates. The results are run three times and reported with (mean \pm std, R_s). The highest test accuracies are marked in bold.

Sparsity	Method	P	$\Delta T = 15000$	$\Delta T = 10000$	$\Delta T = 8000$	$\Delta T = 5000$	$\Delta T = 3000$
			(93.53 \pm 0.02, 0.113)	(93.57 \pm 0.05, 0.150)	(93.44 \pm 0.12, 0.176)	(93.70 \pm 0.19, 0.247)	(93.78 \pm 0.09, 0.353)
0.95	SET	0.9	(93.54 \pm 0.09, 0.100)	(93.54 \pm 0.18, 0.130)	(93.76 \pm 0.07, 0.151)	(93.91 \pm 0.17, 0.210)	(93.63 \pm 0.01, 0.300)
	SET	0.7	(93.51 \pm 0.01, 0.086)	(93.77 \pm 0.21, 0.109)	(93.84 \pm 0.10, 0.125)	(93.93 \pm 0.09, 0.170)	(93.94 \pm 0.08, 0.241)
	SET	0.5	(93.28 \pm 0.01, 0.071)	(93.66 \pm 0.05, 0.086)	(93.80 \pm 0.01, 0.096)	(93.75 \pm 0.18, 0.126)	(93.86 \pm 0.10, 0.174)
	SET	0.3	(93.24 \pm 0.02, 0.056)	(93.35 \pm 0.18, 0.061)	(93.29 \pm 0.06, 0.065)	(93.50 \pm 0.03, 0.076)	(93.34 \pm 0.03, 0.096)
	SET	0.1	(93.24 \pm 0.02, 0.056)	(93.35 \pm 0.18, 0.061)	(93.29 \pm 0.06, 0.065)	(93.50 \pm 0.03, 0.076)	(93.34 \pm 0.03, 0.096)

4.4. Boosting the Performance of DST

Based on the above-mentioned insights, we demonstrate the state-of-the-art sparse training performance with ResNet-50 on ImageNet. More precisely, we choose an update interval ΔT of 4000, a batch size of 64, and an initial pruning rate of 0.5 so that we can achieve a high R_s within a typical training time. We briefly name the improved method as RigL-ITOP. Please see Appendix B for the implementation details. Table 2 shows that without any advanced techniques, our method boosts the accuracy of RigL over the overparameterization-based method (GMP and Lottery Ticket Rewinding (LTR)) (Frankle et al., 2020a)). More importantly, our method requires only 2× training time to match the performance of dense ResNet-50 at 80% sparsity, far less than RigL (5× training time) (Evci et al., 2020a).

Instead of using small batch size, another trick to encourage parameter exploration is sampling from the non-activated weights first when growing new weights. We demonstrate the effectiveness of this idea in Appendix F.

5. The Versatility of ITOP

Although we mainly focus on understanding DST from the ITOP point of view, ITOP can be potentially generalized to other sparsity-inducing categories. Here, we demonstrate its versatility by applying ITOP to two recently popular methods, LTH and PI. We choose SNIP (Lee et al., 2019) as the PI method, as it consistently performs well among different methods for pruning at initialization as shown by Frankle et al. (2020b). Compared with ITOP, LTH and SNIP are two overparameterization-based methods designed for a better initial subnetwork but without consulting any

information yielded during training. We choose magnitude-based weight pruning and random-based weight growth for SNIP and LTH to achieve ITOP and name the corresponding methods as SNIP-SET-ITOP and LTH-SET-ITOP. To make a fair comparison between different pruning criteria, we use global and one-shot pruning for both SNIP and LTH. We train all models for 200 epochs and report the best test accuracy in Figure 7. See Appendix C for the experimental details.

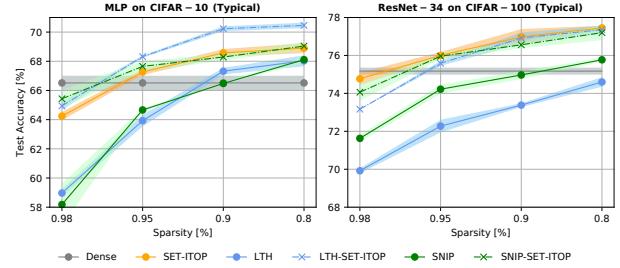


Figure 7. Effect of In-Time Over-Parameterization on SNIP and LTH. “Typical” means training a model for a typical time of 200 epochs.

With a high In-Time Over-Parameterization rate, SET-ITOP consistently outperforms the overparameterization-based methods as well as the dense training, by a large margin. For instance, SET-ITOP can easily match the performance of the corresponding dense models with at most 5% parameters. More importantly, SET-ITOP has dominant performance at the extreme sparsity (98%) over LTH and SNIP, indicating the potential of In-Time Over-Parameterization to address the poor expressibility problem of the extremely sparse neural networks.

Table 2. Performance of sparse ResNet-50 on ImageNet dataset with a typical training time. All results of other methods are obtained from Evcı et al. (2020a) except LTR which is the late-rewinding LTH version obtained from Evcı et al. (2020b). RigL-ITOP_{2×} is obtained by extending the training time by 2 times.

Methods	Top-1 Acc	R_s	Training FLOPs	Test FLOPs	Top-1 Acc	R_s	Training FLOPs	Test FLOPs
Dense	76.8 ± 0.09	1.00	$1 \times (3.2e18)$	$1 \times (8.2e9)$	76.8 ± 0.09	1.00	$1 \times (3.2e18)$	$1 \times (8.2e9)$
sparsity=0.9					sparsity=0.8			
Static	67.7 ± 0.12	0.10	$0.24 \times$	$0.24 \times$	72.1 ± 0.04	0.20	$0.42 \times$	$0.42 \times$
SET	69.6 ± 0.23	-	$0.10 \times$	$0.10 \times$	72.9 ± 0.39	-	$0.23 \times$	$0.23 \times$
SNFS	72.9 ± 0.06	-	$0.50 \times$	$0.24 \times$	75.2 ± 0.11	-	$0.61 \times$	$0.42 \times$
RigL	73.0 ± 0.04	-	$0.25 \times$	$0.24 \times$	75.1 ± 0.05	-	$0.42 \times$	$0.42 \times$
GMP	73.9	-	$0.56 \times$	$0.23 \times$	75.6	-	$0.51 \times$	$0.10 \times$
LTR	-	-	-	-	75.75 ± 0.12	-	-	-
RigL-ITOP	73.82 ± 0.08	0.83	$0.25 \times$	$0.24 \times$	75.84 ± 0.05	0.93	$0.42 \times$	$0.42 \times$
RigL-ITOP _{2×}	75.50 ± 0.09	0.89	$0.50 \times$	$0.24 \times$	76.91 ± 0.07	0.97	$0.84 \times$	$0.42 \times$

It is maybe more interesting that In-Time Over-Parameterization brings large benefits to SNIP and LTH as well. While LTH and SNIP fall short of SET-ITOP, SNIP-SET-ITOP and LTH-SET-ITOP can match or even exceed the performance of SET-ITOP with both MLP and ResNet-34. This observation confirms that ITOP is a foundational concept and can potentially improve any existing sparse training methods.

6. Generalization Improvement of ITOP

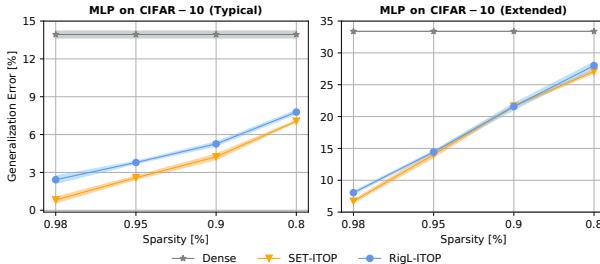


Figure 8. Generalization errors of SET-ITOP, RigL-ITOP, and the dense models.

We further observe the ability of In-Time Over-Parameterization to improve generalization. Figure 8 shows that the generalization error (the difference between the training accuracy and the test accuracy) of In-Time Over-Parameterization (SET-ITOP and RigL-ITOP) and the dense over-parameterization with MLPs on CIFAR-10. It is clear to see that models with the In-Time Over-Parameterization property generalize much better than the corresponding dense models. The generalization error gradually increases as the model gets denser. Together with the results in Figure 7, we can see that the reductions in sparsity lead to better classification performance but worse generalization.

7. Conclusion and Future Work

In this paper, we propose In-Time Over-Parameterization, a variant of dense over-parameterization in the space-time manifold, to be an alternative way to train deep neural networks without the prohibitive dense over-parameterized dependency. We demonstrate the ability of In-Time Over-Parameterization (1) to improve the expressibility of sparse training, (2) to accelerate both training and inference, (3) to understand the underlying mechanism of DST, (4) to prevent overfitting and improve generalization. In addition, we empirically found that, with a sufficient and reliable parameter exploration, randomly-initialized sparse models consistently achieve better performance over those specially-initialized static sparse models. Our paper suggests that it is more effective and efficient to allocate the limited resources to explore more the sparse connectivity space, rather than allocating all resources to find a good sparse initialization.

Our paper discovers the importance of parameter exploration for sparse training. Even though we adjust the hyperparameters of RigL and reach the state of art sparse training performance, the usage of small batch size slows down the training speed of modern architectures. It is interesting to pursue a high In-Time Over-Parameterization rate with large batch size under a typical training time. Moreover, we believe that ITOP has potentials to help people to interpret the networks' decisions (Wong et al., 2021), to improve the robustness out of distribution and uncertainty performance (Zhang et al., 2021), to detect non-spurious correlation (Sagawa et al., 2020), etc.

References

- Allen-Zhu, Z., Li, Y., and Song, Z. A convergence theory for deep learning via over-parameterization. In *International Conference on Machine Learning*, pp. 242–252. PMLR, 2019.

- Atashgahi, Z., Sokar, G., van der Lee, T., Mocanu, E., Mocanu, D. C., Veldhuis, R., and Pechenizkiy, M. Quick and robust feature selection: the strength of energy-efficient sparse training for autoencoders. *arXiv preprint arXiv:2012.00560*, 2020.
- Bellec, G., Kappel, D., Maass, W., and Legenstein, R. Deep rewiring: Training very sparse deep networks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=BJ_wN01C-.
- Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.
- Brutzkus, A., Globerson, A., Malach, E., and Shalev-Shwartz, S. Sgd learns over-parameterized networks that provably generalize on linearly separable data. *arXiv preprint arXiv:1710.10174*, 2017.
- Dai, B., Zhu, C., Guo, B., and Wipf, D. Compressing neural networks using the variational information bottleneck. In *International Conference on Machine Learning*, pp. 1135–1144. PMLR, 2018a.
- Dai, X., Yin, H., and Jha, N. K. Grow and prune compact, fast, and accurate lstms. *arXiv preprint arXiv:1805.11797*, 2018b.
- Dai, X., Yin, H., and Jha, N. K. Nest: A neural network synthesis tool based on a grow-and-prune paradigm. *IEEE Transactions on Computers*, 68(10):1487–1497, 2019.
- de Jorge, P., Sanyal, A., Behl, H. S., Torr, P. H., Rogez, G., and Dokania, P. K. Progressive skeletonization: Trimming more fat from a network at initialization. *arXiv preprint arXiv:2006.09081*, 2020.
- de Jorge, P., Sanyal, A., Behl, H., Torr, P., Rogez, G., and Dokania, P. K. Progressive skeletonization: Trimming more fat from a network at initialization. In *International Conference on Learning Representations*, 2021. URL <https://openreview.net/forum?id=9GsFOUyUPI>.
- Dettmers, T. and Zettlemoyer, L. Sparse networks from scratch: Faster training without losing performance. *arXiv preprint arXiv:1907.04840*, 2019.
- Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.
- Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.
- Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021. URL <https://openreview.net/forum?id=YicbFdNTTy>.
- Du, S., Lee, J., Li, H., Wang, L., and Zhai, X. Gradient descent finds global minima of deep neural networks. In *International Conference on Machine Learning*, pp. 1675–1685. PMLR, 2019.
- Evci, U., Pedregosa, F., Gomez, A., and Elsen, E. The difficulty of training sparse neural networks. *arXiv preprint arXiv:1906.10732*, 2019.
- Evci, U., Gale, T., Menick, J., Castro, P. S., and Elsen, E. Rigging the lottery: Making all tickets winners. In *International Conference on Machine Learning*, 2020a.
- Evci, U., Ioannou, Y. A., Keskin, C., and Dauphin, Y. Gradient flow in sparse neural networks and how lottery tickets win. *arXiv preprint arXiv:2010.03533*, 2020b.
- Frankle, J. and Carbin, M. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2019. URL <https://openreview.net/forum?id=rJ1-b3RcF7>.
- Frankle, J., Dziugaite, G. K., Roy, D., and Carbin, M. Linear mode connectivity and the lottery ticket hypothesis. In *International Conference on Machine Learning*, pp. 3259–3269. PMLR, 2020a.
- Frankle, J., Dziugaite, G. K., Roy, D. M., and Carbin, M. Pruning neural networks at initialization: Why are we missing the mark? *arXiv preprint arXiv:2009.08576*, 2020b.
- Gale, T., Elsen, E., and Hooker, S. The state of sparsity in deep neural networks. *arXiv preprint arXiv:1902.09574*, 2019.
- Gomez, A. N., Zhang, I., Kamalakara, S. R., Madaan, D., Swersky, K., Gal, Y., and Hinton, G. E. Learning sparse networks using targeted dropout. *arXiv preprint arXiv:1905.13678*, 2019.
- Goodfellow, I. J., Vinyals, O., and Saxe, A. M. Qualitatively characterizing neural network optimization problems. In *International Conference on Learning Representations*, 2015.

- Guo, Y., Yao, A., and Chen, Y. Dynamic network surgery for efficient dnns. In *Advances in neural information processing systems*, pp. 1379–1387, 2016.
- Han, S., Pool, J., Tran, J., and Dally, W. Learning both weights and connections for efficient neural network. In *Advances in neural information processing systems*, pp. 1135–1143, 2015.
- Han, S., Mao, H., and Dally, W. J. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. In *International Conference on Learning Representations*, 2016.
- Hassibi, B. and Stork, D. G. Second order derivatives for network pruning: Optimal brain surgeon. In *Advances in neural information processing systems*, pp. 164–171, 1993.
- He, K., Zhang, X., Ren, S., and Sun, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, 2015.
- He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.
- Janowsky, S. A. Pruning versus clipping in neural networks. *Physical Review A*, 39(12):6600, 1989.
- Jayakumar, S., Pascanu, R., Rae, J., Osindero, S., and Elsen, E. Top-kast: Top-k always sparse training. *Advances in Neural Information Processing Systems*, 33, 2020.
- Kusupati, A., Ramanujan, V., Somani, R., Wortzman, M., Jain, P., Kakade, S., and Farhadi, A. Soft threshold weight reparameterization for learnable sparsity. In *International Conference on Machine Learning*, 2020.
- LeCun, Y., Denker, J. S., and Solla, S. A. Optimal brain damage. In *Advances in neural information processing systems*, pp. 598–605, 1990.
- Lee, N., Ajanthan, T., and Torr, P. SNIP: SINGLE-SHOT NETWORK PRUNING BASED ON CONNECTION SENSITIVITY. In *International Conference on Learning Representations*, 2019. URL <https://openreview.net/forum?id=B1VZqjAcYX>.
- Lee, N., Ajanthan, T., Gould, S., and Torr, P. H. S. A signal propagation perspective for pruning neural networks at initialization. In *International Conference on Learning Representations*, 2020. URL <https://openreview.net/forum?id=HJeTo2VFwH>.
- Li, Y. and Liang, Y. Learning overparameterized neural networks via stochastic gradient descent on structured data. In *Advances in Neural Information Processing Systems*, pp. 8157–8166, 2018.
- Lin, J., Rao, Y., Lu, J., and Zhou, J. Runtime neural pruning. In *Advances in neural information processing systems*, pp. 2181–2191, 2017.
- Liu, J., Xu, Z., Shi, R., Cheung, R. C. C., and So, H. K. Dynamic sparse training: Find efficient sparse network from scratch with trainable masked layers. In *International Conference on Learning Representations*, 2020a. URL <https://openreview.net/forum?id=SJ1bGJrtDB>.
- Liu, S., Mocanu, D. C., Matavalam, A. R. R., Pei, Y., and Pechenizkiy, M. Sparse evolutionary deep learning with over one million artificial neurons on commodity hardware. *Neural Computing and Applications*, 2020b.
- Liu, S., van der Lee, T., Yaman, A., Atashgahi, Z., Ferrar, D., Sokar, G., Pechenizkiy, M., and Mocanu, D. Topological insights into sparse neural networks. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 2020c.
- Liu, S., Mocanu, D. C., Pei, Y., and Pechenizkiy, M. Selfish sparse rnn training. *arXiv preprint arXiv:2101.09048*, 2021.
- Liu, Z., Sun, M., Zhou, T., Huang, G., and Darrell, T. Rethinking the value of network pruning. In *International Conference on Learning Representations*, 2019.
- Louizos, C., Welling, M., and Kingma, D. P. Learning sparse neural networks through l_0 regularization. *arXiv preprint arXiv:1712.01312*, 2017.
- Mocanu, D. C. *Network computations in artificial intelligence*. PhD thesis, Technische Universiteit Eindhoven, June 2017.
- Mocanu, D. C., Mocanu, E., Nguyen, P. H., Gibescu, M., and Liotta, A. A topological insight into restricted boltzmann machines. *Machine Learning*, 104(2-3):243–270, 2016.
- Mocanu, D. C., Mocanu, E., Stone, P., Nguyen, P. H., Gibescu, M., and Liotta, A. Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science. *Nature communications*, 9(1):1–12, 2018.
- Molchanov, D., Ashukha, A., and Vetrov, D. Variational dropout sparsifies deep neural networks. In *International Conference on Machine Learning*, 2017.

- Molchanov, P., Tyree, S., Karras, T., Aila, T., and Kautz, J. Pruning convolutional neural networks for resource efficient inference. *arXiv preprint arXiv:1611.06440*, 2016.
- Molchanov, P., Mallya, A., Tyree, S., Frosio, I., and Kautz, J. Importance estimation for neural network pruning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 11264–11272, 2019.
- Mostafa, H. and Wang, X. Parameter efficient training of deep convolutional neural networks by dynamic sparse reparameterization. In *International Conference on Machine Learning*, 2019.
- Mozer, M. C. and Smolensky, P. Using relevance to reduce network size automatically. *Connection Science*, 1(1): 3–16, 1989.
- Narang, S., Elsen, E., Diamos, G., and Sengupta, S. Exploring sparsity in recurrent neural networks. In *International Conference on Learning Representations*, 2017.
- Raihan, M. A. and Aamodt, T. M. Sparse weight activation training. *arXiv preprint arXiv:2001.01969*, 2020.
- Safran, I. and Shamir, O. Spurious local minima are common in two-layer relu neural networks. In *International Conference on Machine Learning*, pp. 4433–4441. PMLR, 2018.
- Sagawa, S., Raghunathan, A., Koh, P. W., and Liang, P. An investigation of why overparameterization exacerbates spurious correlations. In *International Conference on Machine Learning*, pp. 8346–8356. PMLR, 2020.
- Savarese, P., Silva, H., and Maire, M. Winning the lottery with continuous sparsification. *arXiv preprint arXiv:1912.04427*, 2019.
- Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.
- Smith, S. L., Kindermans, P.-J., Ying, C., and Le, Q. V. Don’t decay the learning rate, increase the batch size. *arXiv preprint arXiv:1711.00489*, 2017.
- Soudry, D. and Carmon, Y. No bad local minima: Data independent training error guarantees for multilayer neural networks. *arXiv preprint arXiv:1605.08361*, 2016.
- Srinivas, S., Subramanya, A., and Venkatesh Babu, R. Training sparse neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 138–145, 2017.
- Tanaka, H., Kunin, D., Yamins, D. L., and Ganguli, S. Pruning neural networks without any data by iteratively conserving synaptic flow. *arXiv preprint arXiv:2006.05467*, 2020.
- Wang, C., Zhang, G., and Grosse, R. Picking winning tickets before training by preserving gradient flow. In *International Conference on Learning Representations*, 2020. URL <https://openreview.net/forum?id=SkgsACVKPH>.
- Wen, W., Wu, C., Wang, Y., Chen, Y., and Li, H. Learning structured sparsity in deep neural networks. In *Advances in neural information processing systems*, pp. 2074–2082, 2016.
- Wong, E., Santurkar, S., and Madry, A. Leveraging sparse linear layers for debuggable deep networks. *arXiv preprint arXiv:2105.04857*, 2021.
- Xiao, X., Wang, Z., and Rajasekaran, S. Autoprune: Automatic network pruning by regularizing auxiliary parameters. In *Advances in Neural Information Processing Systems*, pp. 13681–13691, 2019.
- You, H., Li, C., Xu, P., Fu, Y., Wang, Y., Chen, X., Baraniuk, R. G., Wang, Z., and Lin, Y. Drawing early-bird tickets: Towards more efficient training of deep networks. *arXiv preprint arXiv:1909.11957*, 2019.
- Zhang, D., Ahuja, K., Xu, Y., Wang, Y., and Courville, A. Can subnetwork structure be the key to out-of-distribution generalization?, 2021.
- Zhou, A., Ma, Y., Zhu, J., Liu, J., Zhang, Z., Yuan, K., Sun, W., and Li, H. Learning n: M fine-grained structured sparse neural networks from scratch. *arXiv preprint arXiv:2102.04010*, 2021.
- Zhu, M. and Gupta, S. To prune, or not to prune: exploring the efficacy of pruning for model compression. *arXiv preprint arXiv:1710.01878*, 2017.
- Zou, D. and Gu, Q. An improved analysis of training over-parameterized deep neural networks. In *Advances in Neural Information Processing Systems*, pp. 2055–2064, 2019.
- Zou, D., Cao, Y., Zhou, D., and Gu, Q. Gradient descent optimizes over-parameterized deep relu networks. *Machine Learning*, 109(3):467–492, 2020.

Appendices

A. Experimental Settings of Hypothesis Evaluation

In this Appendix, we describe the experimental settings of the hypothesis evaluation in Section 3.2.

A.1. Models

We use MLP on CIFAR-10, VGG-16 on CIFAR-10, ResNet-34 on CIFAR-100 to work through our hypothesis. We describe these models in detail as follows:

MLP. MLP is a clean three-layer MLP with ReLU activation for CIFAR-10. The number of neurons of each layer is 1024, 512, 10, respectively. No other regularization such as dropout or batch normalization is used further.

VGG-16. VGG-16 is a modified CIFAR-10 version of the original VGG model introduced by [Lee et al. \(2019\)](#). The size of the fully-connected layer is reduced to 512 and the dropout layers are replaced with batch normalization to avoid any other sparsification.

ResNet-34. ResNet-34 is the CIFAR-100 version of ResNet with 34 layers introduced by [He et al. \(2016\)](#).

A.2. Algorithm

We choose Sparse Evolutionary Training (SET) ([Mocanu et al., 2018](#)) as the DST method to evaluate our hypothesis. SET helps to avoid the dense over-parameterization bias introduced by the gradient-based methods e.g., RigL and SNFS, as the latter utilize dense gradients in the backward pass to explore new weights. SET starts from a random sparse topology (*Erdős-Rényi*), and optimize the sparse connectivity towards a scale-free topology during training.

This algorithm contains three key steps:

1. Initializing a sparse neural network with *Erdős-Rényi* random graph at a sparsity of S.
2. Training the sparse neural network for ΔT iterations.
3. Removing weights according to the standard magnitude pruning and growing new weights in a random fashion.

Steps 2 and 3 will be repeated iteratively until the end of the training. By doing this, SET maintains a fixed parameter count throughout training.

A.3. Training

Table 3. Experiment hyperparameters of the hypothesis evaluation in Section 3.2. The hyperparameters include Learning Rate (LR), Batch Size (BS), Typical Training Epochs (TT Epochs), Learning Rate Drop (LR Drop), Weight Decay (WD), Sparse Initialization (Sparse Init), Update Interval of the Extended Training (ΔT), Pruning Rate Schedule (Sched), Initial Pruning Rate (P), etc.

Model	Data	Methods	LR	BS	TT Epochs	LR Drop	WD	Sparse Init	ΔT	Sched	P
MLP	CIFAR-10	RigL	0.01	128	200	10x	5e-4	ER	4000	Cosine	0.5
MLP	CIFAR-10	SET	0.01	128	200	10x	5e-4	ER	1500	Cosine	0.5
VGG-16	CIFAR-10	SET	0.1	128	250	10x	5e-4	ERK	2000	Cosine	0.5
ResNet-34	CIFAR-100	SET	0.1	128	200	10x	1e-4	ERK	1000	Cosine	0.5

We basically follow the experimental settings from [Dettmers & Zettlemoyer \(2019\)](#).

For models trained for a typical time, we train them with various update interval ΔT reported in Figure 3. We use a set of 10% training data as the validation set and train on the remaining training data. Weight growth is guided by random sampling and weight pruning is guided by magnitude. We do not specifically finetune the starting point and the finishing point of the parameter exploration. The exploring operation is performed throughout training. The initial sparse connectivity is sampled by the *Erdős-Rényi* distribution introduced in [Mocanu et al. \(2018\)](#). We set the initial pruning rate as 0.5 and

gradually decay it to 0 with a cosine annealing, as introduced in Dettmers & Zettlemoyer (2019). The remaining training hyperparameters are set as follows:

MLP. We train sparse MLPs for 200 epochs by momentum SGD with a learning rate of 0.01 and a momentum coefficient of 0.9. We use a small learning rate 0.01 rather than 0.1, as the dense MLP doesn't converge with a learning rate of 0.1. We decay the learning rate by a factor of 10 every 24000 iterations. We set the batch size as 128. The weight decay is set as 5.0e-4.

VGG-16. We strictly follow the experimental settings from Dettmers & Zettlemoyer (2019) for VGG-16. All sparse models are trained with momentum SGD for 250 epochs with a learning rate of 0.1, decayed by 10 every 30000 mini-batches. We use a batch size of 128 and weight decay to 5.0e-4.

ResNet-34. We train sparse ResNet-34 for 200 epochs with momentum SGD with a learning rate of 0.1, decayed by 10 at the 100 and 150 epoch. We use a batch size of 128 and weight decay to 1.0e-4.

For models trained for an extended training time, we simply extend the training time and the anchor epochs of the learning rate schedule, while using a large ΔT . The update interval ΔT is chosen according to the trade-off shown in Figure 3. Besides the learning steps, the anchor epochs of the learning rate schedule and the pruning rate schedule are also scaled by the same factor. For each training time, the accuracy are averaged over 3 seeds with mean and standard deviation. More detailed training hyperparameters are shared in Table 3.

B. Implementation Details of RigL-ITOP in Section 4.2

Table 4. Experiment hyperparameters in Section 4.2 and Section 5. The hyperparameters include Learning Rate (LR), Batch Size (typical training time / extended training time) (BS), Training Epochs (typical training time / extended training time) (Epochs), Learning Rate Drop (LR Drop), Weight Decay (WD), Sparse Initialization (Sparse Init), Update Interval (ΔT), Pruning Rate Schedule (Sched), Initial Pruning Rate (P), etc.

Model	Data	Methods	LR	BS	Epochs	LR Drop	WD	Sparse Init	ΔT	Sched	P
MLP	CIFAR-10	SET-ITOP	0.01	32 / 128	200 / 4000	10x	5e-4	ER	1500	Cosine	0.5
MLP	CIFAR-10	RigL-ITOP	0.01	32 / 128	200 / 4000	10x	5e-4	ER	4000	Cosine	0.5
ResNet-34	CIFAR-100	SET-ITOP	0.1	32 / 128	200 / 4000	10x	1e-4	ERK	1500	Cosine	0.5
ResNet-34	CIFAR-100	RigL-ITOP	0.1	32 / 128	200 / 4000	10x	1e-4	ERK	4000	Cosine	0.5
ResNet-50	ImageNet	RigL-ITOP	0.1	64 / -	100 / -	10x	1e-4	ERK	4000	Cosine	0.5

In this Appendix, we describe our replication of RigL (Evci et al., 2020a) and the hyperparameters we used for RigL-ITOP.

RigL is a state-of-the-art DST method growing new weights that are expected to receive gradient with high magnitude in the next iteration. Besides, it shows the proposed sparse distribution *Erdős-Rényi-Kernel* (ERK) improves the sparse performance over the *Erdős-Rényi* (ER). Since RigL is originally implemented with TensorFlow, we replicate it with PyTorch based on the implementation from Dettmers & Zettlemoyer (2019). We note that RigL tunes the starting epoch and the ending point of the mask update. To encourage more exploration, we do not follow this strategy and explore sparse connectivities throughout training. We train sparse ResNet-50 for 100 epochs, the same as Dettmers & Zettlemoyer (2019); Evci et al. (2020a). The learning rate is linearly increased to 0.1 with a warm-up in the first 5 epochs and decreased by a factor of 10 at epochs 30, 60, and 90. To reach a high and reliable In-Time Over-Parameterization rate, we use a small batch size of 64 and an update interval of 4000. Batch sizes lower than 64 lead to worse test accuracy. ImageNet experiments were run on 2x NVIDIA Tesla V100. With more fine-tuning, the results of RigL-ITOP (e.g. extended training time) can likely be improved, but we lack the resources to do it. We share the hyperparameters of RigL-ITOP in Table 4.

C. Implementation Details in Section 5

In this Appendix, we describe the hyperparameters of SET-ITOP used in 5 in Table 4. The replication details of LTH and SNIP are given below.

LTH. Lottery Ticket Hypothesis (LTH) (Frankle & Carbin, 2019) shows that there exist sub-networks that can match the accuracy of the dense network when trained with their original initializations. We follow the PyTorch implementation

provide by Liu et al. (2019) on GitHub¹ to replicate LTH.

Given the fact that the iterative pruning process of LTH would lead to much larger training resource costs than SNIP and static sparse training, we use one-shot pruning for LTH. For the typical training time setting, we first train a dense model for 200 epochs, after which we use global and one-shot magnitude pruning to prune the model to the target sparsity and retrain the pruned model with its original initializations for 200 epochs.

SNIP. Single-shot network pruning (SNIP) proposed in Lee et al. (2019), is a method that attempts to prune at initialization before the main training based on the connection sensitivity score $s_i = |\frac{\partial L}{\partial w_i} w_i|$. The weights with the smallest score are pruned. We replicate SNIP based on the PyTorch implementation on GitHub¹. Same as Lee et al. (2019), we use a mini-batch of data to calculate the important scores and obtain the sparse model in a one-shot fashion before the initialization. After that, we train the sparse model without any sparse exploration for 200 epochs.

D. Extended Training Performance of RigL with $\Delta T = 1500$

According to the results from Figure 5, we can see the $\Delta T = 4000$ is a good choice for the update interval of RigL. What if we choose a small update interval, e.g., $\Delta T = 1500$? Here we compare the extended training performance of RigL with two different update intervals 1500 and 4000. The results are shown in Figure 9. It is clear to see models trained with $\Delta T = 1500$ fall short of models trained with $\Delta T = 4000$, which indicates small update intervals is not sufficient for newly weights to catch up the existing weights in terms of magnitude. More importantly, although expected to perform sparse exploration more frequently, models trained with $\Delta T = 1500$ end up with a lower R_s than the ones trained with $\Delta T = 4000$. These results highlight the importance of the sufficient training time for the new weights.

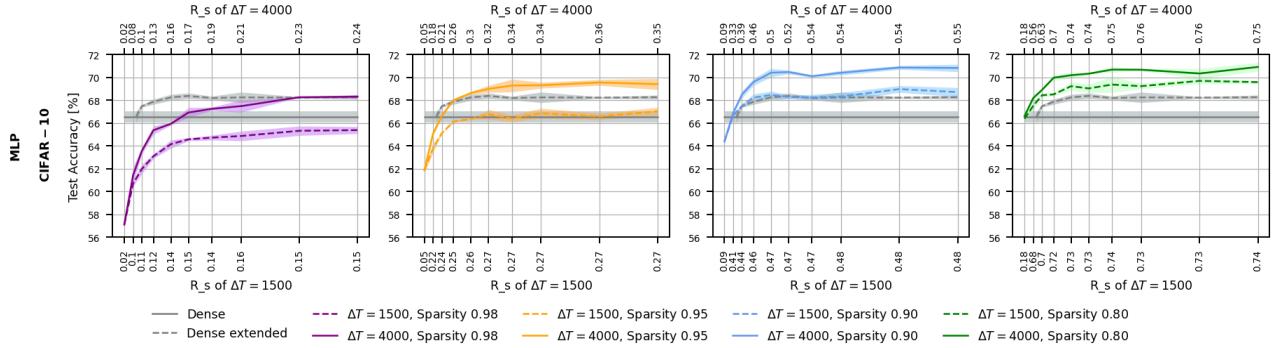


Figure 9. Extended training performance of RigL with update interval $\Delta T = 1500$ and $\Delta T = 4000$.

E. Test Accuracy of RigL with Various Batch Sizes

In this Appendix, we evaluate the performance of RigL with different batch sizes. We choose MLP as our model and the update interval $\Delta T = 4000$. The results are shown in Figure 10. Similar with SET, the performance of RigL also increases as the batch size decrease from 256 to 32. After that, the performance starts to drop due to the noisy input caused by the extreme small batch sizes. The In-Time Over-Parameterization rate (R_s) of RigL is again bounded up to some values. We also provide the comparison between RigL (solid lines) and SET (dashed lines) in this setting. We find a similar pattern with the extended training time, that is, RigL outperforms SET when R_s is small but falls short of SET when sufficient parameters have been reliably explored.

F. Regrowing from the Non-Activated Weights First

One direct way to increase the In-Time Over-Parameterization rate during a typical training time is to sample from the non-activated weights first when performing weight growing. We evaluate this idea with SET by regrowing the non-activated weights first and report the results as SET+ with $(\text{mean} \pm \text{std}, R_s)$ in Table 5. We training sparse ResNet-18 on CIFAR-10

¹<https://github.com/Eric-mingjie/rethinking-network-pruning>

¹<https://github.com/mil-ad/snip>

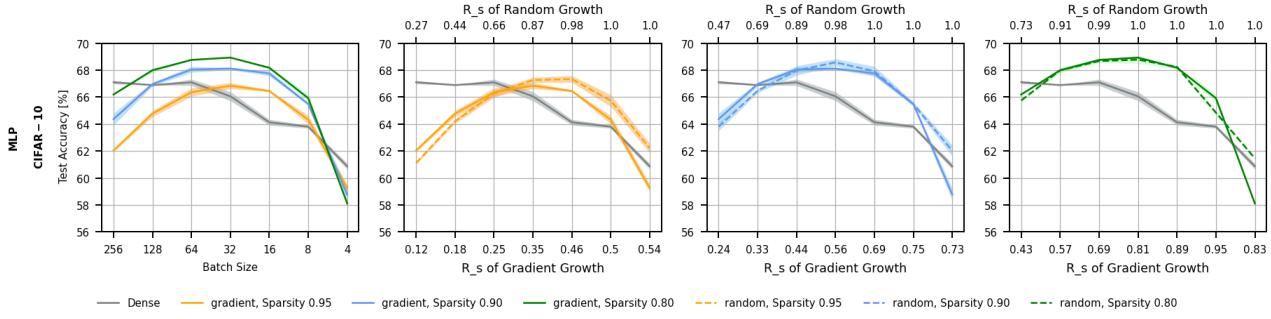


Figure 10. Test accuracy of RigL with various batch sizes. The update interval ΔT is set as 4000.

for 250 epochs with a learning rate of 0.1 decayed by 10x at 125, 187 epochs, a batch size of 128, a pruning rate of 0.5.

When the parameter exploration is insufficient (small R_s), SET+ consistently achieves higher accuracy and higher R_s than SET. While effective, the R_s increase achieved by this modification is relatively limited. This observation highlights an important direction for future work to achieve high R_s within a typical training time.

Table 5. Performance of sparse ResNet-18 on CIFAR-10 with various pruning rates. The results are run three times and reported with (mean \pm std, R_s). The highest test accuracies are marked in bold.

Sparsity	Method	P	$\Delta T = 15000$	$\Delta T = 10000$	$\Delta T = 8000$	$\Delta T = 5000$	$\Delta T = 3000$
0.9	SET	0.5	(94.30 \pm 0.16, 0.162)	(94.47 \pm 0.14, 0.201)	(94.25 \pm 0.10, 0.228)	(94.36 \pm 0.08, 0.302)	(94.54 \pm 0.05, 0.411)
	SET+	0.5	(94.43 \pm 0.14, 0.169)	(94.59 \pm 0.11, 0.215)	(94.54 \pm 0.28, 0.247)	(94.38 \pm 0.07, 0.342)	(94.53 \pm 0.03, 0.492)
0.95	SET	0.5	(93.57 \pm 0.16, 0.086)	(93.46 \pm 0.04, 0.108)	(93.67 \pm 0.04, 0.124)	(93.60 \pm 0.04, 0.170)	(93.61 \pm 0.09, 0.241)
	SET+	0.5	(93.66 \pm 0.14, 0.088)	(93.70 \pm 0.03, 0.114)	(93.66 \pm 0.15, 0.133)	(93.78 \pm 0.04, 0.186)	(94.00 \pm 0.08, 0.272)