Sparse Training via Boosting Pruning Plasticity with Neuroregeneration

Anonymous Author(s)
Affiliation
Address
email

Abstract

Works on lottery ticket hypothesis (LTH) and single-shot network pruning (SNIP) have raised a lot of attention currently on post-training pruning (iterative magnitude pruning), and before-training pruning (pruning at initialization). The former method suffers from an extremely large computation cost and the latter category of methods usually struggles with insufficient performance. In comparison, during-training pruning, a class of pruning methods that simultaneously enjoys the training/inference efficiency and the comparable performance, temporarily, has been less explored. To better understand during-training pruning, we quantitatively study the effect of pruning throughout training from the perspective of **pruning plasticity** (the ability of the pruned networks to recover the original performance). Pruning plasticity can help explain several other empirical observations about neural network pruning in literature. We further find that pruning plasticity can be substantially improved by injecting a brain-inspired mechanism called **neuroregeneration**, i.e., to regenerate the same number of connections as pruned. We design a novel gradual magnitude pruning (GMP) method, named gradual pruning with zero-cost neuroregeneration (**GraNet**), and its dynamic sparse training (DST) variant (**GraNet-ST**). Both of them advance state of the art. Perhaps most impressively, the latter for the first time boosts the sparse-to-sparse training performance over various dense-to-sparse methods by a large margin with ResNet-50 on ImageNet. We will release all codes.

1 Introduction

Neural network pruning is the most common technique to reduce the parameter count, storage requirements, and computational costs of modern neural network architectures. Recently, post-training pruning [48, 29, 18, 46, 10, 53, 72, 5] and before-training pruning [31, 30, 65, 61, 6, 12] have been two fast-rising fields, boosted by lottery tickets hypothesis (LTH) [10] and single-shot network pruning (SNIP) [31]. The process of post-training pruning typically involves fully pre-training a dense network as well as many cycles of retaining (either fine-tuning [18, 17, 38] or rewinding [11, 53]). As the training costs of the state-of-the-art models, e.g., GPT-3 [4] and FixEfficientNet-L2 [62] have explored, this process can lead to a large amount of overhead cost.

Recently emerged methods for pruning at initialization significantly reduce the training cost by identifying a trainable sub-network before the main training process. While promising, the existing methods fail to match the performance achieved by the magnitude pruning after training [12].

Compared with the above-mentioned two classes of pruning, during-training pruning is a class of methods that reap the acceleration benefits of sparsity early on the training and meanwhile achieve promising performance by consulting the information obtained during training. There are some works [74, 13, 32] attempting to gradually prune the network to the desired sparsity during training,

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

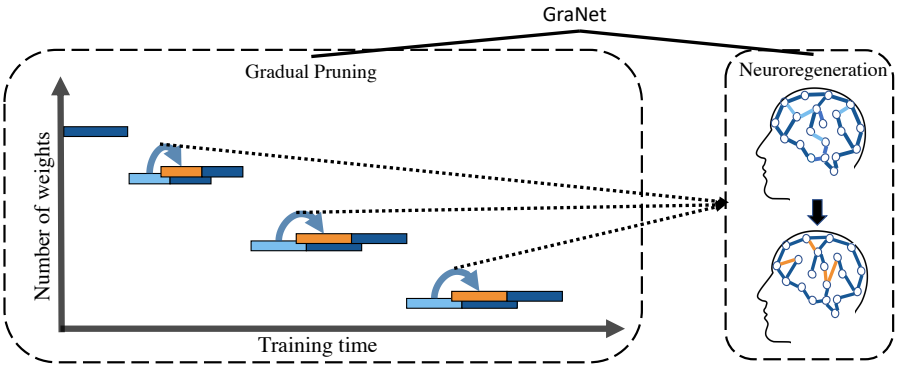

Figure 1: Schematic view of GraNet. **Left:** The gradual pruning process often trains the dense model for some iterations before pruning starts and ends with a sparse trained model. **Right:** We perform neuroregeneration after each gradual pruning step. Light blue blocks/lines refer to the "damaged" connections and orange blocks/lines refer to the regenerated new connections.

while they mainly focus on the performance improvement. Up to now, the understanding of during-training pruning has been less explored due to its more complicated dynamical process, and the performance gap still exists between pruning during training and full dense training.

To better understand the effect of pruning during the optimization process (not at inference), we study the ability of the pruned models to recover the original performance after a short continued training with the *current* learning rate, which we call *pruning plasticity* (see Section 3.1 for a more formal definition). Inspired by the *neuroregeneration* mechanism in the nervous system where new neurons and connections are synthesized to recover the damage in the nervous system [26, 40, 71], we examine if allowing the pruned network to regenerate new connections can improve pruning plasticity, and hence contribute to pruning during training. We consequently propose a parameter-efficient method to regenerate new connections during the gradual pruning process. Different from the existing works for pruning understanding which mainly focus on dense-to-sparse training [41] (training a dense model and prune it to the target sparsity), we also consider sparse-to-sparse training (training a sparse model yet adaptively re-creating the sparsity pattern) which recently has received an upsurge of interest in machine learning [43, 3, 9, 47, 8, 36, 35].

In short, we have the following main findings during the course of the study:

#1. Both pruning rate and learning rate matter for pruning plasticity. When pruned with low pruning rates (e.g., 0.2), both dense-to-sparse training and sparse-to-sparse training can easily recover from pruning. On the contrary, if too many parameters are removed at one time, almost all models suffer from accuracy drops. This finding makes a connection to the success of the iterative magnitude pruning [10, 53, 5, 6, 63], where usually a pruning process with a small pruning rate (e.g., 0.2) needs to be iteratively repeated for good performance.

Pruning plasticity also gradually decreases as the learning rate drops. When pruning happens during the training phase with large learning rates, models can easily recover from pruning (up to a certain level). However, pruning plasticity drops significantly after the second learning rate decay, leading to a situation where the pruned networks can not recover from the continued training. This finding helps to explain several observations (1) for gradual magnitude pruning (GMP), it is always optimal to end pruning before the second learning rate drop [74, 13]; (2) dynamic sparse training (DST) benefits from a monotonically decreasing pruning rate with cosine or linear update schedule [8, 9]; (3) rewinding techniques [11, 53] outperform fine-tuning as rewinding techniques retrain subnetworks with the original learning rate schedule whereas fine-tuning often retrain with small learning rates.

#2. Neuroregeneration improves pruning plasticity. Neuroregeneration [40, 71] refers to the regrowth or repair of nervous tissues, cells, or cell products. Conceptually, it involves synthesizing new neurons, glia, axons, myelin, or synapses, providing extra resources in the long term to replace those damaged by the injury, and achieving a lasting functional recovery. Such mechanism is closely related to the brain plasticity [50], and we borrow this concept to developing a computational regime.

We show that, while regenerating the same number of connections as pruned, the pruning plasticity is observed to improve remarkably, indicating a more neuroplastic model being developed. However,

it increases memory and computational overheads and seems to contradict the benefits of pruning-during-training. This however raises the question: *can we achieve efficient neuroregeneration during training with no extra costs*? We provide an affirmative answer in the next.

#3. Pruning plasticity with neuroregeneration can be leveraged to substantially boost sparse training performance. The above-mentioned findings of pruning plasticity can generalize to the final performance level under a full continued training to the end. Imitating the neuroregeneration behavior [40, 71], we propose a new GMP method – gradual pruning with zero-cost neuroregeneration (GraNet), and its DST variation of GraNet-ST.

In experiments, GraNet and GraNet-ST establish the new state-of-the-art performance bar for dense-to-sparse training and sparse-to-sparse training, respectively. Particularly, GraNet-ST for the first time boosts the sparse-to-sparse training performance over various dense-to-sparse methods by a large margin, with ResNet-50 on ImageNet. Besides the consistent performance improvement, we find the subnetworks that GraNet learned are more accurate than the ones learned by the existing gradual pruning method, providing explanations for the success of GraNet.

2 Related Work

Post-Training Pruning. Methods that yield a sparse neural network from a pre-trained network by pruning the unimportant weights or neurons, to the best of our knowledge, were proposed in [24] and [49]. After that, various pruning methods have emerged to provide increasingly efficient methods to identify sparse neural networks for inference. The pruning criterion includes weight magnitude [18, 10], gradient [59] Hessian [29, 19, 57], Taylor expansion [46, 45], etc. Low-rank decomposition [7, 23, 17, 69] are also used to induce structured sparsity in terms of channels or filters. Most of the above-mentioned pruning methods require many pruning and re-training cycles to achieve the desired performance.

During-Training Pruning. Instead of inheriting weights from a pre-trained model, some works attempt to discover well-performing sparse neural networks with one single training process.

Gradual Magnitude Pruning (GMP), introduced in [74] and studied further in [13], gradually sparsifies the neural network during the training process until the desired sparsity is reached. Besides, [39] and [66] are prior works that enforce the network to sparse during training via L_0 and L_1 regularization, respectively. [58, 33, 54, 68, 28] moved further by introducing trainable sparsity heuristics to learn the sparse masks and weights simultaneously. These methods are all classified as dense-to-sparse training as they start from a dense network.

Dynamic Sparse Training (DST) [43, 3, 47, 8, 9, 35, 34, 25] is another class of methods that prune models during training. The key factor of DST is that it starts from a random initialized sparse network and optimizes the sparse topology as well as the weights simultaneously during training (sparse-to-sparse training). Sparse-to-sparse training usually falls short of dense-to-sparse training in terms of the prediction accuracy. For further details, see the survey of [42, 21]

Before-Training Pruning. Motivated by SNIP [31], many works [65, 61, 6] have emerged recently to explore the possibility of obtaining a trainable sparse neural network before the main training process. [12] demonstrates that the existing methods for pruning at initialization perform equally well when the unpruned weights are randomly shuffled, which reveals that what these methods discover is the layer-wise sparsity ratio, rather than the indispensable weight values and positions. Our analysis shows that both the mask positions and weight values are crucial for GraNet.

3 Methodology for Pruning Plasticity

The primary goal of this paper is to study the effect of pruning as well as neuroregeneration on neural networks during the standard training process. Therefore, we do not consider post-training pruning and before-training pruning. Below, we introduce in detail the definition of pruning plasticity and the experimental design that we used to study pruning plasticity.

3.1 Metrics

Let us denote $W_t \in \mathbb{R}^d$ as the weights of the network and $m_t \in \{0, 1\}^d$ as the binary mask yielded from the pruning method at epoch t. Thus, the pruned network can be denoted as $W_t \odot m_t$. Let T be the total number of epochs the model should be trained. Let $\textsc{ConTrain}^k(W_t \odot m_t, a)$ refers to the function that continues to train the pruned model for k epochs with the learning rate schedule a.

Definition of Pruning plasticity. We define pruning plasticity as $t_{\text{CONTRAIN}^k(W_t \odot m_t, a_t)} - t_{\text{PRE}}$, where t_{PRE} is the test accuracy measured before pruning and $t_{\text{CONTRAIN}^k(W_t \odot m_t, a_t)}$ is the test accuracy measured after k epoch of continued training $\text{CONTRAIN}^k(W_t \odot m_t, a_t)$. Specifically, to better understand the effect of pruning on the current model status and to avoid the effect of learning rate decay, we fix the learning rate as the one when the model is pruned, i.e, a_t. This setting is also appealing to GMP [74, 13] and DST [43, 34, 13] in which most of the pruned models are continually trained with the current learning rate for some time.

Final performance gap. Nevertheless, we also investigate the effect of pruning on the final performance, that is, continually training the pruned networks to the end with the remaining learning rate schedule $\text{CONTRAIN}^{T-t}(W_t \odot m_t, a_{[t+1:T]})$. In this case, we report $t_{\text{CONTRAIN}^{T-t}(W_t \odot m_t, a_{[t+1:T]})} - t_{\text{FINAL}}$, where t_{FINAL} is the final test accuracy of the unpruned models.

3.2 Architectures and Datasets

We choose two commonly used architectures to study pruning plasticity, VGG-19 [56] with batch normalization on CIFAR-10 [27], and ResNet-20 [20] on CIFAR-10.

We share the summary of the networks, data, and hyperparameters of dense-to-sparse training in Table 1. We use standard implementations and hyperparameters available online, with the exception of the small batch size for the ResNet-50 on ImageNet due to the limited hardware resources (2× Tesla V100). All accuracies are in line with the baselines reported in the references [8, 12, 65, 9, 36]. For sparse-to-sparse training, we initialize the sparse network with uniform distribution (all layers have the same sparsity) and keep the rest of the settings the same as the dense training.

Table 1: Summary of the architectures and hyperparameters we study in this paper.

Model	Data	#Epoch	Batch Size	LR	LR Decay, Epoch	Weight Decay	Test Accuracy
ResNet-20	CIFAR-10	160	128	0.1 ($\beta = 0.9$)	10×, [80, 120]	0.0005	92.41±0.04
VGG-19	CIFAR-10	160	128	0.1 ($\beta = 0.9$)	10×, [80, 120]	0.0005	93.85±0.05
	CIFAR-100	160	128	0.1 ($\beta = 0.9$)	10×, [80, 120]	0.0005	73.43±0.08
ResNet-50	CIFAR-10	160	128	0.1 ($\beta = 0.9$)	10×, [80, 120]	0.0005	94.75±0.01
	CIFAR-100	160	128	0.1 ($\beta = 0.9$)	10×, [80, 120]	0.0005	78.23±0.18
	ImageNet	100	64	0.1 ($\beta = 0.9$)	10×, [30, 60, 90]	0.0004	76.80±0.09

3.3 How to Prune, and How to Regenerate

We only consider unstructured sparsity in this paper. The lack of structured sparsity is a limitation of this paper. We believe unstructured sparsity is a more promising direction not only due to its outstanding performance at extreme sparsities but the increasing support for sparse operation in the practical hardware [14, 51, 73, 22]. Different from prior conventions [74, 13, 32, 2] where values of the pruned weights are kept, we set the pruned weights to zero to eliminate the historical information for all implementations in this paper.

Magnitude pruning. We prune the weights with the smallest magnitude, as it has evolved as the standard method when pruning happens during training, e.g., GMP [74, 13] and DST [43, 9, 36]. We are also aware of other pruning criteria including but not limited to Hessian [29, 19, 57], Taylor expansion [46, 45], connection sensitivity [31], Gradient Flow [65], Neural Tangent Kernel [37, 16].

One-shot pruning. To isolate the pruning effect at different training stages and to avoid the interaction between two iterations of pruning, we focus on one-shot pruning. Please note that iterative pruning can also be generalized in our setting, as our experimental design includes neural networks trained at various sparsities and each of them is further pruned with various pruning rates.

Layer-wise pruning and global pruning. We study both the layer-wise magnitude pruning and global magnitude pruning for pruning plasticity. Global magnitude pruning prunes different layers together and leads to non-uniform sparsity distributions; layer-wise pruning operates layer by layer, resulting in uniform distributions.

Gradient-based regeneration. The simplest regeneration scheme is to randomly activate new connections during training [3, 43]. However, it would take a lot of time for random regeneration to discover the important connections, especially for the very extreme sparsities. Alternatively, gradients, including those for the connections with zero weights, provide good indicators for the connection

importance. For this reason, we determine to focus on gradient-based regeneration, i.e., regenerating the same number of connections as pruned with the largest gradient magnitude.

3.4 Experimental Results

We measure pruning plasticity of the pruned models during the process of training with/without regeneration. We report the results of ResNet-20 on CIFAR-10 with global pruning in the main body of the paper. The rest of the experiments are given in Appendix A. Unless otherwise stated, results are qualitatively similar across all networks.

Pruning plasticity. We continue to train the pruned model for 30 epochs and report pruning plasticity in Figure 2. Overall, the learning rate schedule, the pruning rate, and the density of the original models all have a big impact on pruning plasticity.

Pruning plasticity decreases as the learning rate decays for all models with different density levels. The models trained with a large learning rate 0.1 can easily recover, or exceed the original performance except for the extremely large pruning rate 0.98. However, the models obtained during the later training phases can recover only with the careful pruning rate choices, e.g., 0.2 (orange lines) and 0.5 (green lines).

We next demonstrate the effect of connection regeneration on pruning plasticity in the bottom row of Figure 2, it is clear to see that connection regeneration significantly improves pruning plasticity of all the cases, especially for the models that are over-pruned (purple lines).

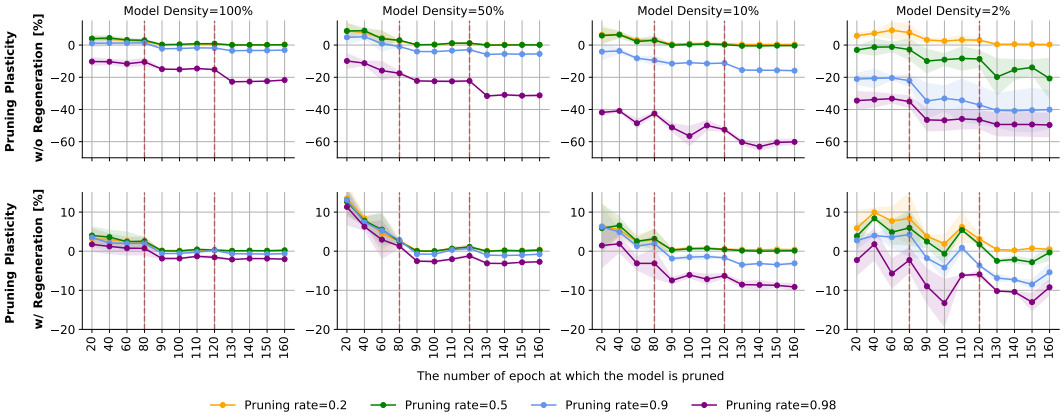

Figure 2: Pruning plasticity (see Section 3.1 for definition) under a 30-epoch continued training with and without connection regeneration for ResNet-20 on CIFAR-10. The vertical red lines refer to the points when the learning rate is decayed. The pruning method is global pruning.

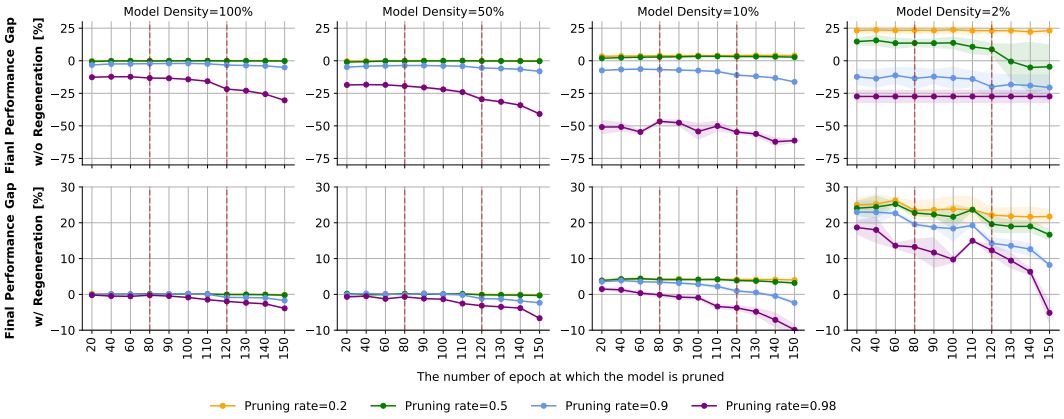

Figure 3: Final performance gap between the unpruned models and the pruned models for ResNet-20 on CIFAR-10. The vertical red lines refer to the points when the learning rate is decayed. The pruning method is global pruning.

Final performance gap. Compared with the current model status, people might be more interested in the effect of pruning on the final performance. We further measure the performance gap between the original test accuracy of the unpruned models and the final test accuracy of the pruned model under a full continued training $\text{CONTRAIN}^{T-t}(W_t \odot m_t, a_{[t+1:T]})$ in Figure 3.

We observe that, in this case, large learning rates do not enjoy large performance improvement, but still, the performance gap increases as the learning rate drops. It is reasonable to conjecture that the accuracy improvement of pruning plasticity with the large learning rate, 0.1, is due to the unconverged performance during the early phase of training. Besides, it is surprising to find that the final performance of extreme sparse networks (e.g., the third column and the fourth column) significantly benefits from careful pruning. Again, the ability of the pruned model to recover from pruning remarkably improves after regenerating the connections back.

4 Gradual Pruning with Zero-Cost Neuroregeneration

So far, we have known that regenerating the important connections to the pruned models during training substantially improves pruning plasticity as well as the final performance. However, naively regenerating extra connections increases the parameter count and conflicts with the motivation of gradual pruning.

Inspired by the mechanism of neuroregeneration in the nervous system, we propose a novel GMP method which we call gradual pruning with zero-cost neuroregeneration (GraNet). GraNet consults the information produced throughout training and regenerates important connections during training in a parameter-efficient fashion. See Appendix B.1 for the pseudocode of GraNet. We introduce the main components of GraNet and its DST variation GraNet-ST below.

4.1 Gradual Pruning

We follow the gradual pruning scheme used in [74] and gradually sparsifies the dense network to the target sparsity level over n pruning iterations. Let us define s_i is the initial sparsity, s_f is the target sparsity, t_0 is is the starting epoch of gradual pruning, t_f is the end epoch of gradual pruning, and Δt is the pruning frequency. The pruning rate of each pruning iteration is:

$$s_t = s_f + (s_i - s_f)\left(1 - \frac{t - t_0}{n\Delta t}\right)^3, \; t \in \{t_0, t_0 + \Delta t, ..., t_0 + n\Delta t\}\,. \tag{1}$$

We choose global pruning for our method as it generally achieves better performance than uniform pruning. We also report the performance of the uniform sparsity as used in [13] in Appendix C.3.

The conventional gradual pruning methods [74, 13] change the mask (not the weight values) to fulfill the pruning operation, so that the pruned connections have the possibility to be reactivated in the later training phases. Despite this, since the weights of the pruned connections are not updated, they have a small chance to receive sufficient updates to exceed the pruning threshold. This hinders the regeneration of the important connections.

4.2 Zero-Cost Neuroregeneration

The main difference between GraNet and the conventional GMP methods [74, 13] is the Zero-Cost Neuroregeneration. Imitating the neuroregeneration of the peripheral nervous system [40, 71] where new neurons and connections are synthesized to replace the damaged ones, we first detect and eliminate the "damaged" connections, and then regenerate the same number of new connections. By doing this, we can achieve connection regeneration without increasing the number of connections.

Concretely, we identify the "damaged" connections as the ones with the smallest weight magnitudes. Small magnitude indicates that either the weight's gradient is small or a large number of oscillations occur to the gradient direction. Therefore, these weights have a small contribution to the training loss and can be removed. Again, we use the gradient as the importance score for regeneration.

Why we call it "Zero-Cost Neuroregeneration"? In addition to not increasing the connection (parameter) count, the backward pass of our method is sparse most of the time even though our regeneration utilizes the dense gradient to identify the important connections. We perform neuroregeneration immediately after each gradual pruning step, meaning that the regeneration occurs only once every several thousand iterations. The extra overhead to calculate the dense gradient can be amortized compared with the whole training costs. Compared with the methods [32, 67] that require updating

Table 2: Test accuracy of pruned VGG-19 and ResNet-50 on CIFAR-10/100. We mark the best sparse-to-sparse training results in blue and the best dense-to-sparse training results in bold. The results reported with (mean ± std) are run with three different random seeds by us. The rest are obtained from [64] and [65]. Note that the accuracy of RigL is higher than the ones reported in [64], as we choose a large update interval following the In-Time Over-Parameterization strategy [36].

Dataset	CIFAR-10			CIFAR-100		
Pruning ratio	90%	95%	98%	90%	95%	98%
VGG-19 (Dense)	93.85±0.05	-	-	73.43±0.08	-	-
SNIP [31]	93.63	93.43	92.05	72.84	71.83	58.46
GraSP [65]	93.30	93.04	92.19	71.95	71.23	68.90
SynFlow [61]	93.35	93.45	92.24	71.77	71.72	70.94
Deep-R [3]	90.81	89.59	86.77	66.83	63.46	59.58
SET [43]	92.46	91.73	89.18	72.36	69.81	65.94
RigL [9]	93.38±0.11	93.06±0.09	91.98±0.09	73.13±0.28	72.14±0.15	69.82±0.09
GraNet-ST (ours)	**93.76±0.08**	**93.54±0.07**	**93.36±0.15**	**73.46±0.13**	**72.95±0.31**	**72.49±0.13**
STR [28]	93.73	93.27	92.21	71.93	71.14	69.89
SIS [64]	**93.99**	93.31	93.16	72.06	71.85	71.17
GMP [13]	93.59±0.10	93.58±0.07	93.52±0.03	73.10±0.12	72.30±0.15	72.07±0.37
GraNet (ours)	93.80±0.10	**93.72±0.11**	**93.63±0.08**	**73.74±0.30**	**73.10±0.04**	**72.35±0.26**
ResNet-50 (Dense)	94.75±0.01	-	-	78.23±0.18	-	-
SNIP [31]	92.65	90.86	87.21	73.14	69.25	58.43
GraSP [65]	92.47	91.32	88.77	73.28	70.29	62.12
SynFlow [61]	92.49	91.22	88.82	73.37	70.37	62.17
RigL [9]	94.45±0.43	93.86±0.25	93.26±0.22	76.50±0.33	76.03±0.34	75.06±0.27
GraNet-ST (ours)	**94.48±0.27**	**94.15±0.28**	**94.01±0.23**	**77.24±0.33**	**76.25±0.52**	**75.38±0.45**
STR [28]	92.59	91.35	88.75	73.45	70.45	62.34
SIS [64]	92.81	91.69	90.11	73.81	70.62	62.75
GMP [13]	94.34±0.09	**94.52±0.08**	94.19±0.04	76.91±0.23	76.42±0.51	75.58±0.20
GraNet (ours)	**94.49±0.08**	94.44±0.01	**94.34±0.17**	**77.29±0.45**	**76.71±0.26**	**76.10±0.20**

all the weights in the backward pass, our method is much more training efficient, as around 2/3 of the training FLOPs is owing to the backward pass [9, 70].

Let us denote r as the ratio of the number of the regenerated connections to the total number of connections; W is the network weight. We first remove r proportion of "damaged" weights with the smallest magnitude by:

$$W' = \text{TopK}\left(|W|,\ 1-r\right). \tag{2}$$

Here $\text{TopK}(v, k)$ returns the weight tensor retaining the top k-proportion of elements from v. Immediately after that, we regenerate r proportion of new connections based on the gradient magnitude:

$$W = W' + \text{TopK}\left(|\mathbf{g}_{i \notin W'}|,\ r\right), \tag{3}$$

where $|\mathbf{g}_{i \notin W'}|$ are the gradient magnitude of the zero weights. We perform Zero-Cost Neuroregeneration layer by layer from the beginning of the training to the end.

Note that our method can easily generalize to the DST scenario by setting the initial sparsity level $s_i > 0$ in Eq. (1). We name it GraNet-ST. Different from the existing DST methods in which the sparsity of the forward pass is fixed, GraNet-ST starts from a denser yet still sparse model and gradually prunes the sparse model to the desired sparsity. For simplicity, we set $s_i = 0.5$, $t_0 = 0$, t_f as the epoch when performing the first learning rate decay for all models. Both GraNet and GraNet-ST sparsify all layers including the first layer and the last layer.

4.3 Experimental Results

We conduct various experiments to evaluate the effectiveness of our methods. We mainly compare GraNet with the dense-to-sparse methods and compare GraNet-ST with the sparse-to-sparse methods. The results of Rigged Lottery (RigL) and GMP with CIFAR-10/100 were reproduced by our implementation with PyTorch so that the only difference between GraNet and GMP is the Zero-Cost Neuroregeneration. For each model, we divide the results into three groups from top to bottom:

pruning at initialization, dynamic sparse training and dense-to-sparse methods. See Appendix B for more implementation details used in the experiments.

CIFAR-10/100. The results of CIFAR-10/100 are shared in Table 2. We can observe that performance differences among different methods on CIFAR-10 are generally small, but still, GraNet consistently improves the performance over GMP except for the sparsity 95%, and achieves the highest accuracy in 4 out of 6 cases. In terms of the more complex data CIFAR-100, the performance differences between the during-training pruning methods and before-training pruning methods are much larger. GraNet again consistently outperforms GMP with all sparsities, highlighting the benefits of Zero-Cost Neuroregeneration. It is maybe more interesting that GraNet even outperforms the post-training method, subdifferential inclusion for sparsity (SIS), by a large margin.

For DST, our proposed GraNet-ST has a dominant performance over other DST methods. Especially at the very extreme sparsity 0.98, our method outperforms the state-of-the-art DST method RigL by 1.38% and 2.67% with VGG-19 on CIFAR-10 and CIFAR-100, respectively.

Table 3: Test accuracy of pruned ResNet-50 on ImageNet dataset. The best results of DST methods are marked as blue and the best results of pruning during training methods are marked in bold. The training/inference FLOPs are normalized with the FLOPs of a dense model.

Method	Top-1 Accuracy	FLOPs (Train)	FLOPs (Test)	TOP-1 Accuracy	FLOPs (Train)	FLOPs (Train)
Dense	76.8±0.09	1x (3.2e18)	1x (8.2e9)			
		S = 0.8			S = 0.9	
Static (ERK)	72.1±0.04	0.42×	0.42×	67.7±0.12	0.24×	0.24×
Small-Dense	72.1±0.06	0.23×	0.23×	67.2±0.12	0.10×	0.10×
SNIP [31]	72.0±0.06	0.23×	0.23×	67.2±0.12	0.10×	0.10×
SET [43]	72.9±0.39	0.23×	0.23×	69.6±0.23	0.10×	0.10×
DSR[47]	73.3	0.40×	0.40×	71.6	0.30×	0.30×
RigL (ERK) [9]	75.1±0.05	0.42×	0.42×	73.0±0.04	0.25×	0.24×
SNFS (ERK) [8]	75.2±0.11	0.61×	0.42×	72.9±0.06	0.50×	0.24×
GraNet-ST (ERK) (ours)	**76.0**	0.37×	0.35×	**74.5**	0.25×	0.20×
STR [28]	**76.1**	n/a	0.17×	74.0	n/a	0.08×
DPF [32]	75.1	0.71×	0.23×	n/a	n/a	n/a
GMP [13]	75.6	0.56×	0.23×	73.9	0.51×	0.10×
GraNet (ours)	75.9	0.41×	0.28×	**74.2**	0.32×	0.16×

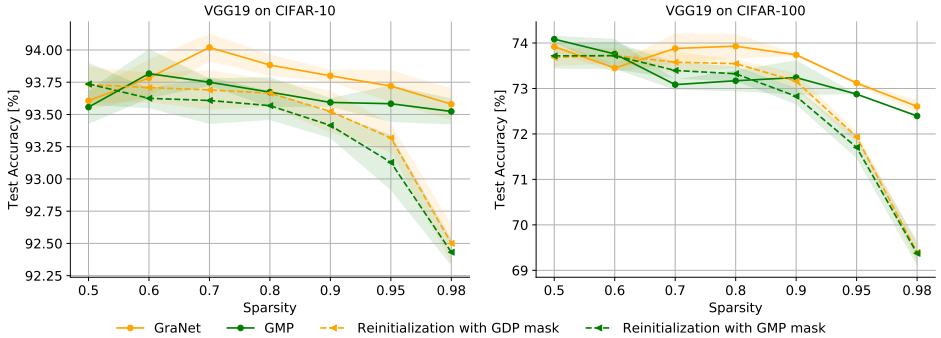

Figure 4: Reinitialization ablation on subnetowrks discovered by GMP and GraNet.

ImageNet. Due to the small data size, the experiments with CIFAR-10/100 may not be sufficient to draw a solid conclusion. We further evaluate our method with ResNet-50 on ImageNet in Table 3. We only run this experiment once due to the limited resources. We set $t_0 = 5$ and $t_f = 55$ for GraNet on ImageNet. Again, GraNet outperforms GMP consistently and achieves the highest accuracy among all the dense-to-sparse methods at sparsity of 0.9.

For GraNet-ST, we choose $t_0 = 0$ and $t_f = 30$ and the initial sparsity s_i is chosen as 0.5. Surprisingly, GraNet-ST significantly boosts the sparse-to-sparse training performance, even over the dense-to-sparse training. Concretely, GraNet-ST outperforms RigL by 0.9% and 1.5% at sparsity 0.8 and 0.9,

respectively. To the best of our knowledge, this is the first time in the literature that sparse-to-sparse training reaches a test accuracy of 76% with ResNet-50 on ImageNet at sparsity 0.8. It is reasonable for GraNet-ST to achieve better accuracy than RigL, since the denser models at the beginning help GraNet-ST explore more parameter space. According to the In-Time Over-Parameterization hypothesis [36], the performance of DST methods is highly correlated with the total number of parameters that the sparse model has visited.

We further report the training/inference FLOPs required by all pruning methods. Compared with other dense-to-sparse methods, the final networks learned by GraNet require more FLOPs to inference, whereas the overall training FLOPs required by GraNet are smaller than others. Even though starting from a dense model, GraNet-ST requires less training and inference FLOPs than the state-of-the-art DST methods. The sparsity budgets for 0.9 sparse ResNet50 on ImageNet-1K learned by our methods are reported in Appendix D. We also report how FLOPs of the pruned ResNet-50 evolve during the course of training in Appendix E.

4.4 Ablation Study of Random Reinitialization

Next, we ask whether what GraNet learned are the specific weights or the sparse structure together with the weights. We randomly reinitialize the pruned network with the same mask and retrain it. The results are given in Figure 4. The performance of the reinitialized networks falls significantly short of the performance achieved by GraNet, indicating that what was learned by GraNet is the sparse connectivity together with the weight values. Besides, we find that the retraining performance of GraNet is higher than GMP. This further confirms that Zero-Cost Neuroregeneration helps the gradual pruning find more accurate mask positions.

4.5 Effect of Retraining on GraNet

In this section, we study if retaining techniques can further improve the performance of the subnetworks discovered by GraNet. The authors of Lottery Ticket Hypothesis (LTH) [10] introduced a retraining technique, even if they did not evaluate it as such, where the subnetworks discovered by iterative magnitude pruning can be retrained in isolation to full accuracy with the original initializations. Later on, learning rate rewinding (LRR) [53] was proposed further to improve the retraining performance by only rewinding the learning rate. Since GraNet also utilizes magnitude pruning to discover subnetworks, it is natural to test if these retraining techniques can bring benefits to GraNet. As shown in Table 4, both retraining techniques do not bring benefits to GraNet. Instead of retaining the subnetworks, we find that simply extending the training time significantly boosts the performance of GraNet with slightly more computation overheads.

Table 4: Effects of LTH and LRR on the subnetworks learned by GraNet. Methods with $_{2\times}$ refer to extending the training steps by 2 times. The results are reported with top-1 test accuracy [%].

Dataset	CIFAR-10			CIFAR-100		
Pruning ratio	90%	95%	98%	90%	95%	98%
VGG-19 (Dense)	93.85±0.05	-	-	73.43±0.08	-	-
GraNet	93.80±0.10	93.72±0.11	93.63±0.08	73.74±0.30	73.10±0.04	72.35±0.26
+ Lottery Ticket Hypothesis	93.63±0.04	93.29±0.05	92.46±0.08	72.97±0.25	71.76±0.22	69.28±0.36
+ Learning Rate Rewinding	93.84±0.14	93.72±0.06	93.53±0.04	73.71±0.08	73.24±0.24	72.50±0.26
GraNet$_{2\times}$	**94.17±0.03**	**93.98±0.07**	**93.94±0.11**	**74.80±0.29**	**73.65±0.32**	**73.63±0.05**
ResNet-50 (Dense)	94.75±0.01	-	-	78.23±0.18	-	-
GraNet (ours)	94.49±0.08	94.44±0.01	94.34±0.17	77.29±0.45	76.71±0.26	76.10±0.20
+ Lottery Ticket Hypothesis	93.96±0.10	93.70±0.15	92.94±0.14	75.74±0.19	74.31±0.10	71.99±0.08
+ Learning Rate Rewinding	94.55±0.13	94.39±0.13	94.20±0.25	77.40±0.14	76.90±0.19	75.75±0.25
GraNet$_{2\times}$	**95.09±0.15**	**94.84±0.11**	**94.69±0.24**	**78.18±0.20**	**78.17±0.20**	**77.15±0.29**

5 Conclusion, and Reflection of Broader Impacts

In this paper, we quantitatively study pruning during training from the perspective of pruning plasticity. Inspired by the findings associated with pruning plasticity and the mechanism of neuroregeneration in the nervous system, we further proposed two pruning methods allowing the cost-free connection regeneration. Both of them advance the state of the art in their own application scenarios.

Our paper re-emphasizes the great potential of during-training pruning in reducing the training/inference resources required by ML models without sacrificing accuracy. It has a significant environmental impact on reducing the energy cost of the ML models and CO2 emissions [1, 52, 15, 55, 60].

References

[1] Z. Atashgahi, G. Sokar, T. van der Lee, E. Mocanu, D. C. Mocanu, R. Veldhuis, and M. Pechenizkiy. Quick and robust feature selection: the strength of energy-efficient sparse training for autoencoders. *arXiv preprint arXiv:2012.00560*, 2020.

[2] B. R. Bartoldson, A. S. Morcos, A. Barbu, and G. Erlebacher. The generalization-stability tradeoff in neural network pruning. *arXiv preprint arXiv:1906.03728*, 2019.

[3] G. Bellec, D. Kappel, W. Maass, and R. Legenstein. Deep rewiring: Training very sparse deep networks. In *International Conference on Learning Representations*, 2018.

[4] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

[5] T. Chen, J. Frankle, S. Chang, S. Liu, Y. Zhang, M. Carbin, and Z. Wang. The lottery tickets hypothesis for supervised and self-supervised pre-training in computer vision models. *arXiv preprint arXiv:2012.06908*, 2020.

[6] P. de Jorge, A. Sanyal, H. S. Behl, P. H. Torr, G. Rogez, and P. K. Dokania. Progressive skeletonization: Trimming more fat from a network at initialization. *arXiv preprint arXiv:2006.09081*, 2020.

[7] E. Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus. Exploiting linear structure within convolutional networks for efficient evaluation. *arXiv preprint arXiv:1404.0736*, 2014.

[8] T. Dettmers and L. Zettlemoyer. Sparse networks from scratch: Faster training without losing performance. *arXiv preprint arXiv:1907.04840*, 2019.

[9] U. Evci, T. Gale, J. Menick, P. S. Castro, and E. Elsen. Rigging the lottery: Making all tickets winners. In *International Conference on Machine Learning*, pages 2943–2952. PMLR, 2020.

[10] J. Frankle and M. Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2019.

[11] J. Frankle, G. K. Dziugaite, D. M. Roy, and M. Carbin. The lottery ticket hypothesis at scale. *arXiv preprint arXiv:1903.01611*, 2019.

[12] J. Frankle, G. K. Dziugaite, D. M. Roy, and M. Carbin. Pruning neural networks at initialization: Why are we missing the mark? *arXiv preprint arXiv:2009.08576*, 2020.

[13] T. Gale, E. Elsen, and S. Hooker. The state of sparsity in deep neural networks. *arXiv preprint arXiv:1902.09574*, 2019.

[14] T. Gale, M. Zaharia, C. Young, and E. Elsen. Sparse gpu kernels for deep learning. *arXiv preprint arXiv:2006.10901*, 2020.

[15] E. García-Martín, C. F. Rodrigues, G. Riley, and H. Grahn. Estimation of energy consumption in machine learning. *Journal of Parallel and Distributed Computing*, 134:75–88, 2019.

[16] T. Gebhart, U. Saxena, and P. Schrater. A unified paths perspective for pruning at initialization. *arXiv preprint arXiv:2101.10552*, 2021.

[17] J. Guo, Y. Li, W. Lin, Y. Chen, and J. Li. Network decoupling: From regular to depthwise separable convolutions. *arXiv preprint arXiv:1808.05517*, 2018.

[18] S. Han, J. Pool, J. Tran, and W. Dally. Learning both weights and connections for efficient neural network. In *Advances in neural information processing systems*, pages 1135–1143, 2015.

[19] B. Hassibi and D. G. Stork. *Second order derivatives for network pruning: Optimal brain surgeon*. Morgan Kaufmann, 1993.

[20] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[21] T. Hoefler, D. Alistarh, T. Ben-Nun, N. Dryden, and A. Peste. Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks. *arXiv preprint arXiv:2102.00554*, 2021.

[22] I. Hubara, B. Chmiel, M. Island, R. Banner, S. Naor, and D. Soudry. Accelerated sparse neural training: A provable and efficient method to find n: M transposable masks. *arXiv preprint arXiv:2102.08124*, 2021.

[23] M. Jaderberg, A. Vedaldi, and A. Zisserman. Speeding up convolutional neural networks with low rank expansions. *arXiv preprint arXiv:1405.3866*, 2014.

[24] S. A. Janowsky. Pruning versus clipping in neural networks. *Physical Review A*, 39(12):6600, 1989.

[25] S. Jayakumar, R. Pascanu, J. Rae, S. Osindero, and E. Elsen. Top-kast: Top-k always sparse training. *Advances in Neural Information Processing Systems*, 33:20744–20754, 2020.

[26] E. R. Kandel, J. H. Schwartz, T. M. Jessell, S. Siegelbaum, A. J. Hudspeth, and S. Mack. *Principles of neural science*, volume 4. McGraw-hill New York, 2000.

[27] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. *Master's thesis, Department of Computer Science, University of Toronto*, 2009.

[28] A. Kusupati, V. Ramanujan, R. Somani, M. Wortsman, P. Jain, S. Kakade, and A. Farhadi. Soft threshold weight reparameterization for learnable sparsity. In *International Conference on Machine Learning*, pages 5544–5555. PMLR, 2020.

[29] Y. LeCun, J. S. Denker, and S. A. Solla. Optimal brain damage. In *Advances in neural information processing systems*, pages 598–605, 1990.

[30] N. Lee, T. Ajanthan, S. Gould, and P. H. Torr. A signal propagation perspective for pruning neural networks at initialization. *arXiv preprint arXiv:1906.06307*, 2019.

[31] N. Lee, T. Ajanthan, and P. H. Torr. Snip: Single-shot network pruning based on connection sensitivity. *International Conference on Learning Representations*, 2018.

[32] T. Lin, S. U. Stich, L. Barba, D. Dmitriev, and M. Jaggi. Dynamic model pruning with feedback. In *International Conference on Learning Representations*, 2020.

[33] J. Liu, Z. Xu, R. Shi, R. C. C. Cheung, and H. K. So. Dynamic sparse training: Find efficient sparse network from scratch with trainable masked layers. In *International Conference on Learning Representations*, 2020.

[34] S. Liu, D. C. Mocanu, A. R. R. Matavalam, Y. Pei, and M. Pechenizkiy. Sparse evolutionary deep learning with over one million artificial neurons on commodity hardware. *Neural Computing and Applications*, 33(7):2589–2604, 2021.

[35] S. Liu, D. C. Mocanu, Y. Pei, and M. Pechenizkiy. Selfish sparse rnn training. *arXiv preprint arXiv:2101.09048*, 2021.

[36] S. Liu, L. Yin, D. C. Mocanu, and M. Pechenizkiy. Do we actually need dense over-parameterization? in-time over-parameterization in sparse training. *arXiv preprint arXiv:2102.02887*, 2021.

[37] T. Liu and F. Zenke. Finding trainable sparse networks through neural tangent transfer. In *International Conference on Machine Learning*, pages 6336–6347. PMLR, 2020.

[38] Z. Liu, M. Sun, T. Zhou, G. Huang, and T. Darrell. Rethinking the value of network pruning. *International Conference on Learning Representations*, 2019.

[39] C. Louizos, M. Welling, and D. P. Kingma. Learning sparse neural networks through l_0 regularization. *International Conference on Learning Representations*, 2018.

[40] M. Mahar and V. Cavalli. Intrinsic mechanisms of neuronal axon regeneration. *Nature Reviews Neuroscience*, 19(6):323–337, 2018.

[41] D. Mittal, S. Bhardwaj, M. M. Khapra, and B. Ravindran. Recovering from random pruning: On the plasticity of deep convolutional neural networks. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 848–857. IEEE, 2018.

[42] D. C. Mocanu, E. Mocanu, T. Pinto, S. Curci, P. H. Nguyen, M. Gibescu, D. Ernst, and Z. A. Vale. Sparse training theory for scalable and efficient agents. *arXiv preprint arXiv:2103.01636*, 2021.

[43] D. C. Mocanu, E. Mocanu, P. Stone, P. H. Nguyen, M. Gibescu, and A. Liotta. Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science. *Nature communications*, 9(1):2383, 2018.

[44] D. Molchanov, A. Ashukha, and D. Vetrov. Variational dropout sparsifies deep neural networks. In *International Conference on Machine Learning*, pages 2498–2507. PMLR, 2017.

[45] P. Molchanov, A. Mallya, S. Tyree, I. Frosio, and J. Kautz. Importance estimation for neural network pruning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11264–11272, 2019.

[46] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz. Pruning convolutional neural networks for resource efficient inference. *International Conference on Learning Representations*, 2016.

[47] H. Mostafa and X. Wang. Parameter efficient training of deep convolutional neural networks by dynamic sparse reparameterization. *International Conference on Machine Learning*, 2019.

[48] M. C. Mozer and P. Smolensky. Skeletonization: A technique for trimming the fat from a network via relevance assessment. In *Advances in neural information processing systems*, pages 107–115, 1989.

[49] M. C. Mozer and P. Smolensky. Using relevance to reduce network size automatically. *Connection Science*, 1(1):3–16, 1989.

[50] P. G. Nagappan, H. Chen, and D.-Y. Wang. Neuroregeneration and plasticity: a review of the physiological mechanisms for achieving functional recovery postinjury. *Military Medical Research*, 7(1):1–16, 2020.

[51] Nvidia. Nvidia a100 tensor core gpu architecture. *https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/nvidia-ampere-architecture-whitepaper.pdf*, 2020.

[52] D. Patterson, J. Gonzalez, Q. Le, C. Liang, L.-M. Munguia, D. Rothchild, D. So, M. Texier, and J. Dean. Carbon emissions and large neural network training. *arXiv preprint arXiv:2104.10350*, 2021.

[53] A. Renda, J. Frankle, and M. Carbin. Comparing rewinding and fine-tuning in neural network pruning. *arXiv preprint arXiv:2003.02389*, 2020.

[54] P. Savarese, H. Silva, and M. Maire. Winning the lottery with continuous sparsification. *arXiv preprint arXiv:1912.04427*, 2019.

[55] R. Schwartz, J. Dodge, N. A. Smith, and O. Etzioni. Green ai. *arXiv preprint arXiv:1907.10597*, 2019.

[56] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *International Conference on Learning Representations*, 2014.

[57] S. P. Singh and D. Alistarh. Woodfisher: Efficient second-order approximation for neural network compression. *Advances in Neural Information Processing Systems*, 33, 2020.

[58] S. Srinivas, A. Subramanya, and R. Venkatesh Babu. Training sparse neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 138–145, 2017.

[59] N. Strom. Scalable distributed dnn training using commodity gpu cloud computing. In *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[60] E. Strubell, A. Ganesh, and A. McCallum. Energy and policy considerations for deep learning in nlp. *arXiv preprint arXiv:1906.02243*, 2019.

[61] H. Tanaka, D. Kunin, D. L. Yamins, and S. Ganguli. Pruning neural networks without any data by iteratively conserving synaptic flow. *arXiv preprint arXiv:2006.05467*, 2020.

[62] H. Touvron, A. Vedaldi, M. Douze, and H. Jégou. Fixing the train-test resolution discrepancy: Fixefficientnet. *arXiv preprint arXiv:2003.08237*, 2020.

[63] S. Verdenius, M. Stol, and P. Forré. Pruning via iterative ranking of sensitivity statistics. *arXiv preprint arXiv:2006.00896*, 2020.

[64] S. Verma and J.-C. Pesquet. Sparsifying networks via subdifferential inclusion, 2021.

[65] C. Wang, G. Zhang, and R. Grosse. Picking winning tickets before training by preserving gradient flow. In *International Conference on Learning Representations*, 2020.

[66] W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li. Learning structured sparsity in deep neural networks. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

[67] M. Wortsman, A. Farhadi, and M. Rastegari. Discovering neural wirings. *arXiv preprint arXiv:1906.00586*, 2019.

[68] X. XIAO, Z. Wang, and S. Rajasekaran. Autoprune: Automatic network pruning by regularizing auxiliary parameters. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[69] Y. Xu, Y. Li, S. Zhang, W. Wen, B. Wang, Y. Qi, Y. Chen, W. Lin, and H. Xiong. Trained rank pruning for efficient deep neural networks. *arXiv preprint arXiv:1812.02402*, 2018.

[70] D. Yang, A. Ghasemazar, X. Ren, M. Golub, G. Lemieux, and M. Lis. Procrustes: a dataflow and accelerator for sparse deep neural network training. In *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 711–724. IEEE, 2020.

[71] G. Yiu and Z. He. Glial inhibition of cns axon regeneration. *Nature Reviews Neuroscience*, 7(8):617–627, 2006.

[72] H. You, C. Li, P. Xu, Y. Fu, Y. Wang, X. Chen, R. G. Baraniuk, Z. Wang, and Y. Lin. Drawing early-bird tickets: Towards more efficient training of deep networks. *arXiv preprint arXiv:1909.11957*, 2019.

[73] A. Zhou, Y. Ma, J. Zhu, J. Liu, Z. Zhang, K. Yuan, W. Sun, and H. Li. Learning n: M fine-grained structured sparse neural networks from scratch. *arXiv preprint arXiv:2102.04010*, 2021.

[74] M. Zhu and S. Gupta. To prune, or not to prune: exploring the efficacy of pruning for model compression. *arXiv preprint arXiv:1710.01878*, 2017.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] See Section 1. We sufficiently describe the background, the research gap, the method we used to bridge the gap, and our findings in abstract as well as the introduction. For convenience, we summarize our main contributions with bullet points in the introduction.

(b) Did you describe the limitations of your work? [Yes] See Section 3.3. The main limitation of this work is that we only focus on unstructured sparsity and no structured sparsity is considered in our paper. The main reason is that, compared with structured sparsity, unstructured sparsity is capable of matching the performance of the dense model at very high sparsities (e.g., 10%, 5%) [43, 36, 13]. Once this great potential is supported by future hardware, it can have a significant positive impact on our planet by saving a huge amount of energy and reducing overall total carbon emissions. Furthermore, an increasing number of companies and researchers start considering including unstructured sparsity support in their hardware [14, 51, 73, 22]. Our paper provides an algorithm for such advances to consider.

(c) Did you discuss any potential negative societal impacts of your work? [Yes] Since we propose an algorithm to improve the performance of during-training pruning, it might be unfavorable for after-training pruning which is relatively more costly.

(d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]

2. If you are including theoretical results...

(a) Did you state the full set of assumptions of all theoretical results? [No] We definitely agree that the theoretical results of our method e.g., generalization bound, would be more beneficial to our community. We would like to study it as future work.

(b) Did you include complete proofs of all theoretical results? [No]

3. If you ran experiments...

(a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See Table 1 in Section3.2 as well as Appendix B and Appendix C.

(b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Appendix B and Appendix C

(c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]

(d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Section 3.2. All experiments are run run with NVIDIA Tesla V100.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...

(a) If your work uses existing assets, did you cite the creators? [Yes]

(b) Did you mention the license of the assets? [Yes]

(c) Did you include any new assets either in the supplemental material or as a URL? [Yes]

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes]

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]